News Vertical Search using User-Generated Content

Richard McCreadie

School of Computing Science

University of Glasgow

A thesis submitted for the degree of

Doctor of Philosophy

September 2012

Abstract

The thesis investigates how content produced by end-users on the World Wide Web — referred to as user-generated content — can enhance the news vertical aspect of a universal Web search engine, such that news-related queries can be satisfied more accurately, comprehensively and in a more timely manner. We propose a news search framework to describe the news vertical aspect of a universal web search engine. This framework is comprised of four components, each providing a different piece of functionality. The Top Events Identification component identifies the most important events that are happening at any given moment using discussion in user-generated content streams. The News Query Classification component classifies incoming queries as news-related or not in real-time. The Ranking News-Related Content component finds and ranks relevant content for news-related user queries from multiple streams of news and user-generated content. Finally, the News-Related Content Integration component merges the previously ranked content for the user query into the Web search ranking. In this thesis, we argue that user-generated content can be leveraged in one or more of these components to better satisfy news-related user queries. Potential enhancements include the faster identification of news queries relating to breaking news events, more accurate classification of news-related queries, increased coverage of the events searched for by the user or increased freshness in the results returned.

Approaches to tackle each of the four components of the news search framework are proposed, which aim to leverage user-generated content. Together, these approaches form the news vertical component of a universal Web search engine. Each approach proposed for a component is thoroughly evaluated using one or more datasets developed for that component. Conclusions are derived concerning whether the use of user-generated content enhances the component in question using an appropriate measure, namely: effectiveness when ranking events by their current importance/newsworthiness for the Top Events Identification component; classification accuracy over different types of query for the News Query Classification component; relevance of the documents returned for the Ranking News-Related Content component; and end-user preference for rankings integrating user-generated content in comparison to the unaltered Web search ranking for the News-Related Content Integration component. Analysis of the proposed approaches themselves, the effective settings for the deployment of those approaches and insights into their behaviour are also discussed.

In particular, the evaluation of the Top Events Identification component examines how effectively events — represented by newswire articles — can be ranked by their importance using two different streams of user-generated content, namely blog posts and Twitter tweets. Evaluation of the proposed approaches for this component indicates that blog posts are an effective source of evidence to use when ranking events and that these approaches achieve state-of-the-art effectiveness. Using the same approaches instead driven by a stream of tweets, provide a story ranking performance that is significantly more effective than random, but is not consistent across all of the datasets and approaches tested. Insights are provided into the reasons for this with regard to the transient nature of discussion in Twitter.

Through the evaluation of the News Query Classification component, we show that the use of timely features extracted from different news and user-generated content sources can increase the accuracy of news query classification over relying upon newswire provider streams alone. Evidence also suggests that the usefulness of the user-generated content sources varies as news events mature, with some sources becoming more influential over time as new content is published, leading to an upward trend in classification accuracy.

The Ranking News-Related Content component evaluation investigates how to effectively rank content from the blogosphere and Twitter for news-related user queries. Of the approaches tested, we show that learning to rank approaches using features specific to blog posts/tweets lead to state-of-the-art ranking effectiveness under real-time constraints.

Finally this thesis demonstrates that the majority of end-users prefer rankings integrated with user-generated content for news-related queries to rankings containing only Web search results or integrated with only newswire articles. Of the user-generated content sources tested, the most popular source is shown to be Twitter, particularly for queries relating to breaking events.

The central contributions of this thesis are the introduction of a news search framework, the approaches to tackle each of the four components of the framework that integrate user-generated content and their subsequent evaluation in a simulated real-time setting. This thesis draws insights from a broad range of experiments spanning the entire search process for news-related queries. The experiments reported in this thesis demonstrate the potential and scope for enhancements that can be brought about by the leverage of user-generated content for real-time news search and related applications.

Acknowledgements

This thesis was make possible by the great support that I have received from a wide variety of people during the course of my PhD.

I first would like to thank my parents, who constantly encouraged me and helped keep the more practical aspects of my life intact, especially when deadlines were approaching.

I must thank both my supervisor, Iadh Ounis, and co-supervisor Craig Macdonald. Their supervision taught me how to integrate and build upon prior works from many areas to tackle new tasks and design sound experiments to explore them. Moreover, without their meticulous attention to detail, this thesis would not have been possible.

I would also like to thank the TerrierTeam research group with whom I shared an office for four years (and counting), most notably Rodrygo Santos who provided feedback relating to some of the experiments in this thesis.

Finally, I wish to thank Charlie Clarke for his mentoring during the SIGIR Doctoral Consortium in the early stage of this thesis, which helped to shape it.

Contents

Abstract **i**

Acknowledgements **iii**

1 Introduction **1**

1.1 Introduction 1

1.2 Motivation 2

1.3 Thesis Statement 6

1.4 Contributions 6

1.5 Origins of the Material 8

1.6 Thesis Outline 9

2 Information Retrieval **12**

2.1 Introduction 12

2.2 Indexing 14

2.2.1 Tokenisation 15

2.2.2 Stopword Removal 15

2.2.3 Stemming 16

2.2.4 Index Data Structures 16

2.2.5 Large Scale Indexing and MapReduce 19

2.2.6 Document Boilerplate 23

2.3 Document Retrieval 25

2.3.1 Term Frequency - Inverse Document Frequency (TF-IDF) 26

2.3.2 The Best Match Weighting Model 27

2.3.3 Divergence From Randomness 28

2.3.4 Language Modelling 29

2.3.5 Query Expansion and Pseudo Relevance Feedback 30
2.4 Evaluation 32
2.4.1 Evaluation Measures 32
2.4.2 Cranfield Paradigm 34
2.4.3 TREC 35
2.4.4 Relevance Assessment 35
2.5 Machine Learning for IR 36
2.5.1 Classification 36
2.5.2 Learning to Rank 38
2.5.3 Machine Learning Libraries/Toolkits 39
2.6 Topic Detection and Tracking (TDT) 39
2.6.1 Topic Tracking 40
2.7 Ranking Aggregates 41
2.7.1 Aggregate Ranking Approaches 41
2.8 Resource Selection/Vertical Search 43
2.8.1 Source Selection via Expected Relevancy 45
2.8.2 Source Selection via Vertical Type 46
2.8.3 Merging of Document Rankings 47
2.9 Conclusions 48

3 Search and User Generated Content 50
3.1 Introduction 50
3.2 The Blogosphere 51
3.2.1 Blog Search Queries 52
3.2.2 Blog Post Search 53
3.2.3 Top News Stories Identification 54
3.2.4 News and Blogs 56
3.2.5 Summary 57
3.3 Digg 57
3.3.1 Digg Users 58
3.3.2 Story Promotion 58
3.3.3 Summary 59
3.4 Twitter 59

3.4.1 Twitter Characteristics . . . 60
3.4.2 Twitter Search . . . 61
3.4.3 TREC 2011 Microblog Track . . . 62
3.4.4 Event Detection in Twitter . . . 63
3.4.5 Summary . . . 64
3.5 Query-Logs . . . 64
3.5.1 Query Log Analysis . . . 65
3.5.2 Query Types and Vertical Selection . . . 66
3.5.3 Summary . . . 68
3.6 Wikipedia . . . 68
3.6.1 News Events and Wikipedia . . . 69
3.6.2 Summary . . . 69
3.7 Conclusions . . . 70

4 A Framework for Real-time News Search **72**
4.1 Introduction . . . 72
4.2 Real-Time News Queries . . . 73
4.3 A Taxonomy of News Queries . . . 74
4.4 Framework Overview . . . 76
4.5 Top Events Identification . . . 79
4.6 News Query Classification . . . 83
4.7 Ranking News-Related Content . . . 86
4.7.1 Blog Posts . . . 87
4.7.2 Real-time Tweet Search . . . 88
4.8 News-Related Content Integration . . . 89
4.9 Conclusions . . . 91

5 Evaluation Datasets and Crowdsourcing **92**
5.1 Introduction . . . 92
5.2 Datasets Overview . . . 94
5.2.1 Top Event Identification Datasets . . . 97
5.2.2 News Query Classification Datasets . . . 99
5.2.3 Ranking News-Related Content Datasets . . . 102
5.2.4 News-Related Content Integration Datasets . . . 103

5.2.5 Summary 104
5.3 Crowdsourcing 104
5.3.1 Work Validation Strategies 106
5.3.2 Agreement Measures 107
5.3.3 Structured Crowdsourced Evaluation 108
5.4 Crowdsourced Task: Identifying Top Events 108
5.4.1 Task Description 109
5.4.2 Interface Design for News Article Assessment 109
5.4.3 Validation of Worker Assessments 110
5.4.4 Crowdsourcing Configuration 112
5.4.5 Assessment Accuracy 112
5.5 Crowdsourcing Task: Classifying User Queries 113
5.5.1 Task Description 113
5.5.2 Interface Design for News Query Classification 114
5.5.3 Validation of Worker Labelling 116
5.5.4 Crowdsourcing Configuration 116
5.5.5 Label Quality 117
5.6 Crowdsourcing Task: Ranking Blog Posts for a News Story 118
5.6.1 Task Description 118
5.6.2 Interface Design for Blog Post Ranking 119
5.6.3 Validation of Worker Assessments 120
5.6.4 Crowdsourcing Configuration 122
5.6.5 Assessment Quality 122
5.7 Crowdsourced Task: Document Relevance Assessment 123
5.7.1 Task Description 123
5.7.2 Interface Design for Document Relevance Assessment 124
5.7.3 Validation of Worker Assessments 126
5.7.4 Crowdsourcing Configuration 126
5.7.5 Assessment Quality 126
5.8 Conclusions 127

6 Identifying Top News **129**
6.1 Introduction 129
6.2 Unlearned Story Ranking Approaches 131
6.2.1 Voting for Important Stories (Votes) 131
6.2.2 Relevance Weighted Aggregation (RWA) 132
6.2.3 Story Length Normalisation (RWAN) 133
6.2.4 Temporal Promotion (GaussBoost) 134
6.2.5 Maximum Burst Magnitude (MaxBurst) 136
6.3 Learning To Rank Stories (LTRS) 137
6.4 Experimental Methodology 140
6.4.1 Blog Stream Methodology 140
6.4.2 Tweet Stream Methodology 143
6.5 Research Questions 144
6.6 Evaluation: News Story Ranking with Blogs 146
6.6.1 Votes, RWA and RWAN 147
6.6.2 GaussBoost 148
6.6.3 Learning to Rank Stories Effectiveness 149
6.6.4 Influential Story Ranking Features 151
6.6.5 Event Promotion in Blogs 152
6.6.6 Conclusions 153
6.7 Evaluation: News Story Ranking with Tweets 153
6.7.1 Votes, RWA and RWAN 154
6.7.2 GaussBoost 156
6.7.3 MaxBurst 158
6.7.4 Learning to Rank Stories Effectiveness 159
6.7.5 Influential Story Ranking Features 162
6.7.6 Event Promotion in Tweets 163
6.7.7 Conclusions 164
6.8 Conclusions 165

7 News Query Classification 168

7.1 Introduction 168

7.2 Feature Aggregation from News Streams (FANS) 169

7.3 Experimental Methodology 172

7.4 News Query Classification Features 174

7.5 Research Questions 177

7.6 Evaluating News Query Classification on the May 2006 Dataset 178

7.6.1 Integrating User-generated Content 178

7.6.2 Feature Importance 180

7.6.3 Timeliness 183

7.7 Evaluating News Query Classification on the April 2012 Dataset 184

7.7.1 Integrating User-generated Content 185

7.7.2 Feature Importance 187

7.7.3 Timeliness 191

7.8 Conclusions 192

8 Ranking News Content 195

8.1 Introduction 195

8.2 Learning to Rank User-Generated Content 196

8.2.1 Learning to Rank Blog Posts 197

8.2.2 Learning to Rank Tweets 201

8.3 Experimental Methodology 205

8.3.1 Blog Post Ranking Methodology 205

8.3.2 Tweet Ranking Methodology 207

8.4 Research Questions 207

8.5 Evaluation: Blog Post Ranking 208

8.5.1 Learning to Rank Performance 208

8.5.2 Ranking Effectiveness Over Time 209

8.5.3 Blog Post Ranking Features 210

8.5.4 Conclusions 212

8.6 Evaluation: Tweet Ranking 213

8.6.1 Learning to Rank Performance 213

8.6.2 Tweet Ranking Features 215

8.6.3 Conclusions . . . 216
8.7 Conclusions . . . 217

9 News-Related Content Integration 219
9.1 Introduction . . . 219
9.2 Resource Selection for News-Related Content Integration . . . 220
9.3 Comparative Ranking Evaluation and Presentation . . . 222
9.4 Research Questions . . . 226
9.5 Experimental Methodology . . . 226
9.5.1 News-Related Queries and Content Sources . . . 226
9.5.2 Document Rankings . . . 227
9.5.3 Experiment structure . . . 229
9.5.4 Worker Instructions . . . 230
9.5.5 Crowdsourcing Configuration and Statistics . . . 230
9.6 Evaluation: Integrating News and User-generated Content . . . 231
9.6.1 Integrating Top Events . . . 231
9.6.2 Integrating News and User-Generated Content vs the Web Search Ranking . . . 233
9.6.3 Integrating User-Generated Content vs News Content . . . 235
9.6.4 Integrating User-Generated Content for Breaking and Recent Events . . . 237
9.7 Conclusions . . . 238

10 Conclusions and Future Work 241
10.1 Contributions and Conclusions . . . 241
10.1.1 Contributions . . . 241
10.1.2 Conclusions . . . 244
10.1.3 Thesis Conclusions . . . 248
10.2 Directions for Future Work . . . 249
10.3 Closing Remarks . . . 250

A Parameter Analysis for Top Events Identification 252
A.1 Parameter Analysis on Blogs . . . 252
A.1.1 News Story Representations . . . 252
A.1.2 Ranking Depth . . . 254

A.2 Parameter Analysis on Twitter 254
A.2.1 Background Window 256
A.2.2 Recent Window 257
A.2.3 News Story Representations 258
A.2.4 Ranking Depth 261

B Worker Instructions for Crowdsourcing 264
B.1 Story Ranking Task Worker Instructions 265
B.2 News Query Classification Worker Instructions 266
B.3 Blog Post Ranking Task Worker Instructions 267
B.4 Document Assessment Task Worker Instructions 268

C Features for News Query Classification 269
C.1 Query-only Features 269
C.2 Frequency Features 270
C.3 Retrieval Features 273
C.4 Burstiness Features 274
C.5 Importance Features 276
C.6 Stream-Specific Features 277

D News Query Classification Learners 279

References 304

List of Figures

1.1 Example search results for the query ‘olympics’ to the Google Search engine on the 22/07/12. 3

2.1 Pseudo-code for the word-counting operation, expressed as map and reduce functions. 21
2.2 Pseudo-code for the Terrier indexing strategy Map function. 23
2.3 Pseudo-code for the Terrier indexing strategy Reduce function. 24

4.1 Top ranked results from the Google Web search engine for the query ‘olympic swimming’ on 01/08/12 at two points in time. 74
4.2 Overview of the proposed News Search Framework. 77
4.3 Top news identification component within our news search framework. 80
4.4 Top Bing search results for the query ‘cnn’ on 27/06/2012. 81
4.5 News query classification component within our news search framework. 83
4.6 Top news-related results returned for the query ‘find silver in london’ from the Google Web search engine on 01/08/12. 84
4.7 Google Trends result for the query ‘ash cloud’ (made on 30/01/2012). 85
4.8 Ranking News-Related Content component within our news search framework. 86
4.9 News-Related Content Integration component within our news search framework. . . . 89

5.1 Illustration of available document corpora from different years. 94
5.2 An example of news story importance scoring on the nytimes.com website - 10/01/2012. 100
5.3 A screenshot of the external judging interface shown to workers within the instructions. 110
5.4 Displayed summary of three workers judgements for a single task instance. 111
5.5 Task instance possibly completed by a bot. 111
5.6 News query classification instances used in this thesis. Not all of the news story summaries are shown in the Headline+Summary interface to save space. 115

5.7 A screenshot of the external judging interface shown to workers within the instructions. 119
5.8 An example blog post rendering both when cleaned and as HTML. The HTML rendering contains the same content as the cleaned version, however, due to a missing CSS template one would need to scroll down to get to it. 121
5.9 The basic interface with which our workers label each query. 125

6.1 Example of Gaussian curves with varying values of l (the Gaussian curve width). Note that for illustration clarity, weights have been normalised so that $\Delta t = 0$ will always be assigned a weight of 1. 134
6.2 An example of 12 story ranking features generated from two story representations, two tweet rankers, two extracted tweet-features and two aggregation models. 139
6.3 Example tweet that would be promoted by the Mentions feature. 163
6.4 The average age of selected tweets as the retrieval depth is increased. 164

7.1 A hypothetical distribution of documents published over time that are about a newsworthy event in three streams. 170
7.2 Illustration of feature generation for news query classification. 172
7.3 News query classification F_1 performance for three temporal settings using each news and user-generated content stream. 182
7.4 News query classification effectiveness when using different combinations of stream features query-only features (66/33 split). 188
7.5 News query classification F_1 when using features derived from each stream over time. 193

9.1 Illustration of the top events and news and user-generated content integration into the Web search ranking ($Integration_{News}(R_W, I)$ and $Integration_{RE}(Q, R_{1..n}, R_W, S)$). 222
9.2 Comparison of our framework presentation of search results to the Bing Web search engine. 224
9.3 Illustration of the ranking interface produced for the query 'north korea' on 13/4/12 with our framework, where the enhanced ranking integrates Digg posts. The first two rankings are the ranking pair to be assessed. The third (red) bordered ranking a validation ranking designed to catch workers that are randomly clicking. 225
9.4 The number of topics where the majority of users preferred the Web search ranking or the enhanced ranking that integrates each source. 234

9.5 Topic comparison between rankings enhanced with news articles and those enhanced with other user-generated content sources. 236

List of Tables

2.1 A document-posting list 17
2.2 An example inverted-index 17
2.3 An example document index 18
2.4 IR corpora in order of increasing size 20
2.5 Document ranking/classification confusion table for calculating precision and recall. . 33

4.1 Classifications for the 2268 queries sampled from the MSN query log. 76

5.1 Statistics of all corpora representing document streams used in this thesis. 95
5.2 A listing of the datasets and corpora used to evaluate each component of our news search framework. For each dataset, whether they were produced for the Text REtrieval Conference (TREC) and whether they use crowdsourcing for relevance assessment generation is also noted. Where the crowdsourced assessments column is marked with a ✔*, that indicates that we collaborated with TREC to produce the assessments using crowdsourcing. 96
5.3 Top Events Identification datasets used in this thesis. Datasets that were produced by TREC or that contain crowdsourced relevance assessments are denoted with a ✔ in the associated column. Where crowdsourced assessments are marked with a ✔* the dataset assessments were crowdsourced by ourselves on behalf of TREC. 98
5.4 News Query Classification datasets used in this thesis. Datasets that were produced by TREC or that contain crowdsourced relevance assessments are denoted with a ✔ in the associated column. 100
5.5 Ranking News-Related Content datasets used in this thesis. Datasets that were produced by TREC or that contain crowdsourced relevance assessments are denoted with a ✔ in the associated column. Where crowdsourced assessments is marked with a ✔* the dataset assessments were crowdsourced by ourselves on behalf of TREC. 102

5.6 Summary of the $NRCI_{Apr2012}$ News-Related Content Integration dataset used in this thesis. Datasets that were produced by TREC or that contain crowdsourced relevance assessments are denoted with a ✔ in the associated column. 104
5.7 Summary of the tasks crowdsourced for each dataset. 105
5.8 Judgement distribution and agreement on a per category basis. 112
5.9 Statistics for the MSN query log from 2006, as well as the sampled NQC^{test}_{2006} and NQC^{full}_{2006}. 114
5.10 Statistics for the validation sets created for the NQC^{test}_{2006} and NQC^{full}_{2006}. 116
5.11 Quality measures for news query classification on the approximately 100 query NQC^{test}_{2006} with varying interfaces, in addition to the over 1000 query NQC^{full}_{2006} using the *Link-Supported* interface. 118
5.12 Average amount paid per hour to workers and work composition for each batch of HITs. 122
5.13 Accuracy of blog post assessments in comparison to the track organiser judgements. . . 123
5.14 Statistics of the documents to be assessed from each of the five news and user-generated content sources. 124
5.15 Statistics of the document relevance assessments produced by MTurk workers. 127

6.1 A sample assignment of votes for two articles (a_1 and a_2) over 3 days (d_1 to d_3). . . . 132
6.2 Blog stream LTRS feature instances used. 143
6.3 Tweet stream LTRS feature instances used. 145
6.4 News story ranking performance of Votes, RWA and RWAN in comparison to the best participating system to the TREC top news stories identification task during 2009 and 2010. 146
6.5 News story ranking performance of Votes$_{GaussBoost}$ and RWA$_{GaussBoost}$ when the Gaussian curve width l is varied. 150
6.6 Comparison between our learning to rank approach when trained under both Cross-Corpus and Per-Corpus training with the best TREC systems in terms of overall story ranking performance under the Blogs$_{09}$ and Blogs$_{10}$ story ranking topics. ▲ denotes a statistically significant increase over the best individual feature (t-test $p < 0.05$). 151
6.7 Strongest 5 positive and negative features on the Blogs$_{09}$ and Blogs$_{10}$ datasets. Boldened feature weights indicate features with a high impact on the story ranking. The document feature component is not reported, as only document relevance is used in this experiment. 152

6.8 News story ranking performance of Votes, RWA and RWAN in comparison to the random baseline. ▲ and ▼ indicate statistically significant increases/decreases over Votes using the Tweet representation under *t-test* $p < 0.05$. ▲▲ and ▼▼ indicate statistically significant increases/decreases over Votes using the Tweet representation under *t-test* $p < 0.01$. 155

6.9 News story ranking performance of Votes$_{GaussBoost}$ and RWA$_{GaussBoost}$ in comparison to Votes and RWA. The most effective NDCG and Success@1 performance observed under each approach and dataset is highlighted. ▲ and ▼ indicate statistically significant increases/decreases over the baseline run without $GaussBoost$ under *t-test* $p < 0.05$. ▲▲ and ▼▼ indicate statistically significant increases/decreases under *t-test* $p < 0.01$. 157

6.10 News story ranking performance of Votes$_{MaxBurst}$, RWA$_{MaxBurst}$ and RWAN$_{MaxBurst}$. The most effective NDCG and Success@1 performance observed under each approach and dataset is highlighted. ▲ and ▼ indicate statistically significant increases/decreases over the base news story ranking approach without $MaxBurst$ under *t-test* $p < 0.05$. ▲▲ and ▼▼ indicate statistically significant increases/decreases under *t-test* $p < 0.01$. 158

6.11 News story ranking performance of Votes$_{MaxBurst+GaussBoost}$ and RWA$_{MaxBurst+GaussBoost}$ in comparison to Votes and RWA. The most effective NDCG and Success@1 performance observed under each approach and dataset is highlighted. ▲ and ▼ indicate statistically significant increases/decreases over the base news story ranking approach without $MaxBurst$ and $GaussBoost$ under *t-test* $p < 0.05$. ▲▲ and ▼▼ indicate statistically significant increases/decreases under *t-test* $p < 0.01$. 160

6.12 Learned story ranking performance when training across story ranking datasets when using AFS. The most effective NDCG and Success@1 performance observed under each approach and dataset is highlighted. ▲ and ▼ indicate statistically significant increases/decreases over the random baseline under *t-test* $p < 0.05$. ▲▲ and ▼▼ indicate statistically significant increases/decreases under *t-test* $p < 0.01$. 161

6.13 Top 5 features (comprised of their four individual elements) selected by AFS on the Twitter$_{12-TR}$ dataset. 162

7.1 News Streams available within each of the two news query classification datasets. . . . 173

7.2 All query-only and stream features extracted from the $NQC_{May2006}$ and $NQC_{Apr2012}$ datasets, excluding stream-specifc features. 176

7.3 All 10 corpus specific stream features extracted $NQC'_{2012}s$ Digg and Twitter streams. . 177
7.4 News query classification performance using multiple combinations of news and user-generated streams. Statistical significance (t-test $p < 0.05$) over the BBC+Guardian+Telegraph news baseline is denoted ▲. 179
7.5 News query classification performance for each of the three query feature categories described in Appendix C.1. 180
7.6 News query classification performance using different feature set/streams combinations. Statistically significant (t-test $p < 0.05$) decreases in performance over using all features are denoted ▼. 181
7.7 News query classification performance in terms of precision, recall and F_1 for three temporal settings using all streams. Statistically significant (t-test $p < 0.05$) decreases in performance over classification at the original query time are denoted ▼. 183
7.8 Percentage of the four query types that were classified correctly at the three points in time. 184
7.9 News query classification performance using multiple combinations of news and user-generated corpora. 187
7.10 Positive and Negative query features selected on the 66% training set and their weights (66/33 split). 189
7.11 News query classification performance using different feature set/streams combinations. The best performing feature combinations that use a single stream are highlighted in bold. Cases where combining streams results in increased performance over any single stream is also highlighted in bold. 191
7.12 News query classification performance in terms of precision, recall and F_1 for three temporal settings using each news and user-generated content stream. The best performing feature combinations that use a single stream are highlighted in bold. Cases where combining streams results in increased performance over any single stream is also highlighted in bold. Statistically significant increases (t-test p<0.05) over the same features extracted from the query time are denoted ▲. 192

8.1 Blog post ranking features. 199
8.2 Tweet learning to rank features. 204

8.3 Blog post ranking performance of the best TREC runs in comparison to our proposed learning to rank approach. Note that $LTR^{C8.2.1}_{BlogPosts}$ was the best TREC system, hence we do not list it twice. Cases where our learning to rank approach achieves statistically significant increases (t-test p<0.05) over the DPH sample are denoted ▲. 209
8.4 The most influential features for blog post ranking for time t 210
8.5 The most influential features for blog post ranking for time $t + 7days$ 213
8.6 Tweet ranking effectiveness of LTR using a variety of learning algorithms. ▲ denotes a statistically significant (t-test p<0.05) increase over tweet ranking sample (*Sample*). . . 214
8.7 The most influential features for tweet ranking. 216

9.1 Statistics of the 219 news-related queries used during our evaluation. 227
9.2 News-related content integration evaluation experiment statistics. 230
9.3 Number of end users that prefer the Web search ranking with the integrated top events ranking in comparison to the Web search ranking unaltered. 232
9.4 Number of preferences for both the Web and Enhanced rankings. 233
9.5 Proportion of query topics where integrating user-generated content was more effective than integrating newswire articles. 238

A.1 News story ranking performance of Votes and RWA when using different story representations a. 253
A.2 News story ranking performance of Votes and RWA when retrieval depth is varied. . . 255
A.3 News story ranking performance of Votes when the background window size of available tweets is varied. The most effective NDCG and Success@1 performance observed under each approach and dataset is highlighted. ▲ and ▼ indicate statistically significant increases/decreases over the same approach with a default window size of 10 days under *t-test* $p < 0.05$. ▲▲ and ▼▼ indicate statistically significant increases/decreases under *t-test* $p < 0.01$. 257
A.4 News story ranking performance of Votes when the recent time window size r is varied. The background window size w is set to 10. The most effective NDCG and Success@1 performance observed under each approach and dataset is highlighted. ▲ and ▼ indicate statistically significant increases/decreases over the same approach with a default window size of 10 days under *t-test* $p < 0.05$. ▲▲ and ▼▼ indicate statistically significant increases/decreases under *t-test* $p < 0.01$. 259

A.5 News story ranking performance of Votes when the recent time window size is varied. The background window size w is set to 2 days. The most effective NDCG and Success@1 performance observed under each approach and dataset is highlighted. ▲ and ▼ indicate statistically significant increases/decreases over the same approach with a default window size of 10 days under *t-test* $p < 0.05$. ▲▲ and ▼▼ indicate statistically significant increases/decreases under *t-test* $p < 0.01$. 260

A.6 News story ranking performance of Votes, RWA and RWAN with and without query expansion on Wikipedia. The most effective NDCG and Success@1 performance observed under each approach and dataset is highlighted. ▲ and ▼ indicate statistically significant increases/decreases over the same approach with a default window size of 10 days under *t-test* $p < 0.05$. ▲▲ and ▼▼ indicate statistically significant increases/decreases under *t-test* $p < 0.01$. 261

A.7 News story ranking performance of Votes, RWA and RWAN as ranking depth is varied. The most effective NDCG and Success@1 performance observed under each approach and dataset is highlighted. ▲ and ▼ indicate statistically significant increases/decreases over the same approach with a default window size of 10 days under *t-test* $p < 0.05$. ▲▲ and ▼▼ indicate statistically significant increases/decreases under *t-test* $p < 0.01$. . 263

D.1 News Query classification effectiveness when using all features under a variety of learning algorithms. Statistical significance over Logistic Regression (Multinomial) (paired t-test p<0.05) denoted ▲. 280

Chapter 1

Introduction

1.1 Introduction

Major universal Web search engines serve around a billion of user queries each day (Norman, 2011). It has been reported that up to 11% of these Web search queries are related to current news (Bar-Ilan *et al.*, 2009). For these queries, users may either be searching for information about a specific news story, or looking to see what the top news stories of the moment are. We refer to these types of queries as *news-related queries*.

User-generated content refers to documents published on the World Wide Web (Web) by the general public, rather than by paid individuals, corporations or companies. Social media websites, such as Twitter[1], are prime examples of user-generated content sources. On Twitter, users publicly post messages to other users that subscribe to them. Other examples of widely used user-generated content are blogs and Wikipedia[2] pages.

The online news landscape has been greatly affected by the emergence of user-generated content. In particular, the role of the public in the online news space has undergone a dramatic shift from a static news consumer base into a dynamic news machine, where news stories are summarised, discussed and commented upon in real-time (Bandari *et al.*, 2012; O'Reilly & Milstein, 2009). Indeed, news today is spread by a variety of diverse media (Kwak *et al.*, 2010; Scott, 2007), driven by user interaction and user-generated content, e.g. news reporting in Twitter (O'Brien, 2009).

This shift has had consequences for universal search engines when tackling news-related queries. In particular, users can now be searching for information about events only seconds after those events have occurred (Hansell, 2007). These queries are challenging to satisfy, since the search engine may not yet be aware of the event in question, making the identification of such queries as news-related difficult.

[1]`http://twitter.org`
[2]`http://wikipedia.org`

Moreover, if the query is identified as news-related, no relevant newswire articles or Web search results are likely to have yet been published. Instead, the only relevant content about a breaking event may exist within user-generated content sources.

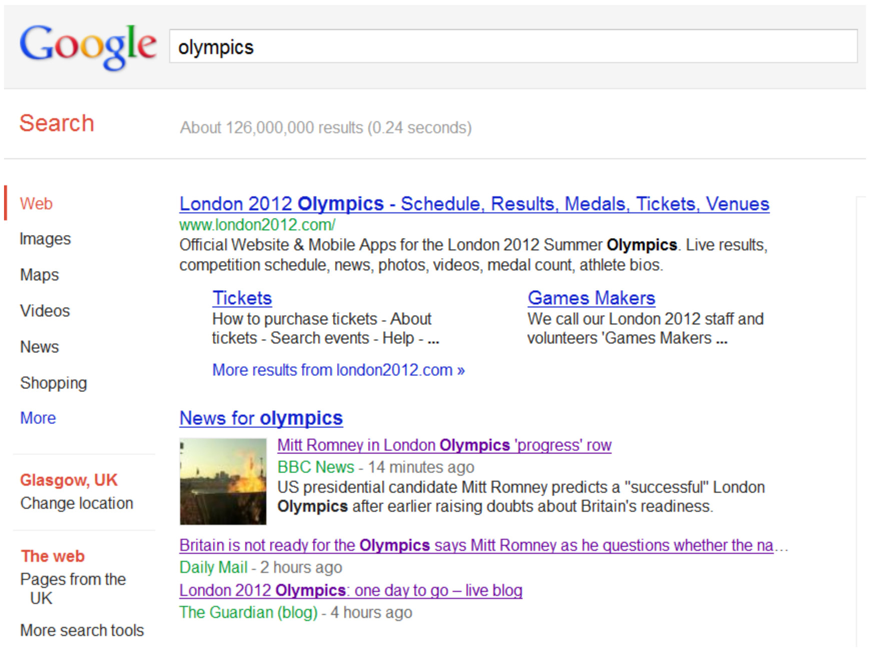
In this thesis, we investigate how a universal Web search engine can be enhanced with user-generated content, such that news-related user queries can be satisfied more accurately, fully and in a more timely manner. The aim is to provide end-users with a final search result ranking that provides increased relevance and coverage of events for news-related queries. In particular, we focus on improving the final search result ranking for those news-related queries that cannot be easily satisfied by newswire articles, either because the event that the user is searching for has just broken and hence no news articles have yet been published, or because newswire article coverage is insufficiently detailed or out-of-date.

1.2 Motivation

A universal Web search engine enables users to search large collections of heterogeneous documents published on the Web. For example, news websites, forums, blogs and public wiki sites are all indexed by Web search engines. However, modern universal Web search engines do not treat all their documents in the same manner. In particular, they define groups of documents referred to as *verticals*, where each vertical can be searched independently (Arguello *et al.*, 2009). The motivation for vertical search is to provide increased search effectiveness over specific types of documents. For example, an image vertical allows the searching of images, while a shopping vertical allows the searching of products. Universal Web search engines search one or more of their verticals when the user submits a query. They return an aggregate of the results retrieved from verticals that they predict are potentially relevant to the user's query. In this thesis, we consider a universal Web search engine that supports two verticals in particular, namely a Web vertical and a news vertical. The Web vertical ranks Web documents to produce an initial ranking for the user query, referred to as the *Web search ranking*. The news vertical integrates specifically news-related documents into the Web search ranking for queries that the search engine identifies are news-related.

For illustration, Figure 1.1 shows the search results returned by the Google[1] search engine for the query 'olympics' submitted on the 22nd of July 2012. As can be seen, Google provides search from multiple verticals, each listed down the left hand side of the Web search results. Meanwhile, within the results themselves, we see that the London Olympics homepage has been retrieved at the top rank. This result comes from their Web vertical, which represents search over the Web as a whole. On the other

[1]`http://google.com`


olympics

Search

About 126,000,000 results (0.24 seconds)

Web
Images
Maps
Videos
News
Shopping
More

Glasgow, UK
Change location

The web
Pages from the UK

More search tools

London 2012 **Olympics** - Schedule, Results, Medals, Tickets, Venues
www.london2012.com/
Official Website & Mobile Apps for the London 2012 Summer **Olympics**. Live results, competition schedule, news, photos, videos, medal count, athlete bios.

Tickets
How to purchase tickets - About tickets - Search events - Help - ...

Games Makers
We call our London 2012 staff and volunteers 'Games Makers ...

More results from london2012.com »

News for **olympics**

Mitt Romney in London **Olympics** 'progress' row
BBC News - 14 minutes ago
US presidential candidate Mitt Romney predicts a "successful" London **Olympics** after earlier raising doubts about Britain's readiness.

Britain is not ready for the **Olympics** says Mitt Romney as he questions whether the na...
Daily Mail - 2 hours ago

London 2012 **Olympics**: one day to go – live blog
The Guardian (blog) - 4 hours ago

Figure 1.1: Example search results for the query 'olympics' to the Google Search engine on the 22/07/12.

hand, the second ranked result contains current newswire articles about the London Olympics, retrieved from the news vertical.

Notably, a news vertical has some unique characteristics that distinguish it from other types of vertical. Firstly a typical news vertical searches over newswire articles and other documents from multiple providers. In our previous example illustrated by Figure 1.1, two newswire articles and one blog post were returned: one article from the BBC[1]; one article from the Daily Mail[2]; and one blog post from the official Guardian blog[3]. We refer to the different types of documents retrieved by the news vertical in general as *news-related content*. This news-related content is then integrated into the Web search results. Secondly, the news-related queries that the news vertical serves are often *real-time* in nature, i.e. they are concerned with ongoing or recent events. As a result, news-related content retrieved by the news vertical is often timely. For example, from Figure 1.1, we observe that the news-related content returned was published between 14 minutes and 4 hours prior to the time the query was issued. Retrieving timely news-related content is a one of the main challenges of for the news-vertical, since for fast-moving events, content may quickly become out-dated.

[1]http://bbc.co.uk
[2]http://dailymail.co.uk
[3]http://guardian.co.uk

The search process of a universal Web search engine that supports both a Web and news vertical can be described as follows. A user initiates the search by entering a textual query into the universal search engine. The search engine then routes that query to each of the Web and news verticals. The Web vertical ranks Web documents for the query, producing the Web search ranking. The news vertical first classifies the user query as news-related or not. If the query is not news-related, then the Web search ranking is left unaltered. On the other hand, if the query is news-related, then news-related content is first ranked for the user query. Next, the most relevant and timely of this news-related content is selected. Finally, the selected news-related content is either interleaved into the Web search results or displayed nearby in a manner that the search engine considers will increase relevance and coverage for the user query.

However, recall how the shift in the online news landscape has changed the way that news is now being reported. In particular, social media websites are now known for reporting news (Munk, 2009). Moreover, the speed at which news is reported and spread has greatly increased (Shamma *et al.*, 2009). For example, when a passenger plane crashed in the Hudson river in the U.S., the event was first reported from the scene via the microblogging website Twitter only minutes after the plane touched down (O'Brien, 2009). Indeed, the first Twitter tweet was posted a full 15 minutes before the first newswire article was published. Furthermore, one of the main consequences that this has had for universal Web search engines is that users are searching for news about events soon after they occur. For instance, when New York city suffered a power cut, the Google search engine reported observing related queries within two seconds of the event (Hansell, 2007), well before any newswire articles were published about it.

These changes have introduced new challenges for universal Web search engines that support a news vertical. In particular, we identify four key challenges and describe each below:

Top Events Identification: A news vertical only integrates additional news-related content into the Web search ranking for news-related queries. We consider a query to be news-related if it refers to a current event, such as a celebrity death. However, not all events are potentially newsworthy. For instance, newswire providers such as Reuters continually report on stock market fluctuations[1], but in most cases one would not consider these newsworthy. Hence, there is a need to identify what are the *important* events happening at any point in time. For instance, if the user entered the query 'ash' during April 2010, then the news vertical would need to know that the Eyjafjallajkull volcano in Iceland had erupted[2]

[1]`http://uk.reuters.com/business/markets`
[2]`http://news.bbc.co.uk/1/hi/8623534.stm`

and that this event is sufficiently important to make 'ash' a potentially news-related query. Moreover, the events considered important may change rapidly. For instance, a prominent unpredictable event such as an earthquake is likely to dominate the news headlines, reducing the importance of all the other events reported on that day. Since news is now being reported in user-generated content sources, it may be no longer sufficient to rely on newswire providers, e.g. CNN[1], since their top stories may be out-of-date. For example, when Michael Jackson died, the news broke first on Twitter (Munk, 2009). Hence, the first challenge we identify is how a news vertical can automatically identify the top events right now?

News Query Classification: When a new query arrives, the news vertical must first classify it as news-related or not. However, news-related queries are particularly challenging to classify, since they change rapidly over time in line with the newsworthy events of the moment. Furthermore, due to the increased speed at which breaking news is reported via user-generated content, news-related queries can now appear in advance of newswire articles reporting on the event (Hansell, 2007). For instance, in our previous Hudson river plane crash example, there was a delay of 15 minutes between the story breaking on Twitter and the first published newswire article (O'Brien, 2009). This raises the second challenge that we identify, namely how can we accurately classify news-related queries in the current real-time news environment?

Ranking News-related Content: If a user query is identified as holding a news-related intent, then news-related content needs to be ranked for the query. However, for queries relating to events reported first in social media, newswire articles will not have yet been published, rather, only user-generated content sources will contain relevant documents at that time. Hence, a traditional news vertical that only tracks newswire sources would not retrieve any relevant content that could then be integrated. Moreover, user-generated content has the potential to provide relevant content in cases where newswire coverage is unavailable or suspect. For example, during a bombing in Damascus (Syria), on the ground investigators used social media to report what they could confirm had happened in the absence of a free-press (Fiona McCann, 2012). Hence, a universal Web search engine that supports a news vertical needs to rank content from user-generated sources in addition to newswire sources to satisfy news-related queries where relevant newswire articles have not yet been published or are insufficient to provide full coverage of the event. As a result, the third challenge that we identify is how can user-generated content be effectively ranked for a news-related user query?

[1]`http://cnn.com`

News-Related Content Integration: The introduction of user-generated content ranked from a variety of diverse sources, e.g. Twitter, Blogs or Diggs. complicates the task of integrating this content into the Web search ranking. In particular, given a large volume of newswire and user-generated content ranked for the user-query, some subset of this content needs to be selected for inclusion into the Web search results. Moreover, the types of content selected will vary from query-to-query, since the value added by user-generated content is both query and time-dependent. For example, user-generated content can have a high value when there are few or no newswire articles yet available, but is less useful if there are readily available high quality newswire articles that will satisfy the query. As a result, the fourth challenge that we identify is how to effectively integrate this news-related content into the Web search ranking, such that the relevance to, and coverage of, the event that the user is looking for is increased.

1.3 Thesis Statement

This thesis states that user-generated content can aid in satisfying news-related queries submitted to universal Web search engines in real-time. In particular, we propose a news search framework that allows to fully integrate the functionalities that are needed to deploy a news vertical within a universal search engine: a classifier to identify news-related queries, a mechanism to identify the useful news-related content and a strategy to integrate this selected content into the Web search ranking. The essential argument made in this thesis is that user-generated content can improve the performance of each component of the framework. In particular, we hypothesise that user-generated content can increase the accuracy of the deployed query classifier. We also hypothesise that user-generated content may include unique, valuable and relevant information to answer some types of news-related queries, such as those relating to breaking events. Furthermore, we postulate that the appropriate integration of user-generated content into the Web search ranking will increase the coverage of events that users are looking to know more about. Finally, we postulate that all the aforementioned functionalities can be enhanced by identifying the most important events at the time the query was issued using the volume of available user-generated content.

1.4 Contributions

The main contributions of this thesis are the following. Firstly, we introduce our news search framework that describes the functionalities required by a universal Web search engine to deploy a news vertical. This framework is defines four distinct components. Three of these components directly relate to the three news search functionalities described earlier (see Section 1.3), i.e. the classification of news queries, the selection of news-related content and the integration of that news-related content into the Web search results. The final component produces rankings of events by their current importance, which are used to enhance the performance of the other components. Notably, our framework is designed such that these four components each encapsulate one of the four news search challenges that we identified in Section 1.2. Our framework is therefore a suitable basis for investigating these news search challenges using user-generated content. For clarity, we refer to each framework component using the label for its associated news search challenge.

Next, for each of the four components, we propose, develop and evaluate novel approaches to enable or enhance them. We describe our contributions for each component below:

Top Events Identification: We propose a fully automatic real-time approach for the identification of currently important news-related events based upon voting theory. This approach leverages the volume and relatedness of discussions within different user-generated content streams to rank events. Using real-time discussions from both tweet and blog post streams, we experiment to determine how effectively it can rank events represented by newswire articles. In addition, we investigate to what extent the age of discussions used impact the event ranking.

News Query Classification: We propose a real-time classification approach that leverages recent discussions from a variety of parallel news and user-generated content sources. This approach has two aims. Firstly, it uses up-to-the-minute discussions in user-generated content to increase classification accuracy for queries relating to breaking news. Secondly, it combines evidence from multiple news and user-generated content sources simultaneously, e.g. newswire articles and blogs, to increase classification accuracy overall. We evaluate how accurately news and non-news queries can be classified using parallel newswire, blog, tweet, Wikipedia edit and digg streams.

Ranking News-related Content: We use the phrase 'news-related content' to describe a wide variety of different document types from newswire and user-generated sources. Prior works have shown that IR document weighting models can be used to effectively rank much of this content, from newswire articles to Wikipedia pages. However, some types of (user-generated) content can be difficult to rank, since their characteristics differ greatly from typical Web documents. For instance, tweets are constrained in length to 140 characters. We propose approaches that leverage machine learning for rank those types of user-generated content that have unique characteristics. In particular, we propose to overcome the challenges that these characteristics bring, by representing each user-generated document as a set of features, with the aim of better describing what makes these unique types of document relevant. For example, useful feature about a tweet might include whether it contains a link to a news article. We identify tweets and blogs as having unique characteristics and develop machine learned ranking models for each. We then evaluate these models to determine whether they are effective.

News-Related Content Integration: Finally, through a large-scale user-study performed using the emerging field of crowdsourcing, we evaluate how user-generated content can be effectively integrated into the Web search ranking. The aim is to increase the relevance and coverage of the Web search ranking for news-related user queries. Here, we aim to determine the types of content should be selected for each query, e.g. tweets or news-articles. We also examine the manner in which this content should be displayed. For instance, should blogs be displayed after news articles, and how much screen real-estate should be used for each? In general, we study when and where end-users consider the addition of user-generated content to the search ranking to be advantageous.

A additional contribution of this work is the development of methodologies for the production of new evaluation datasets for each of the four framework components. In particular, develop methodologies that leverage the emerging field of crowdsourcing to produce relevance assessments, label queries and evaluate our final search rankings with end-users. We experimentally evaluate our crowdsourcing methodologies to determine whether they are able to produce accurate assessments and labels as well as conclude on best practices.

1.5 Origins of the Material

Some of the work within this thesis can be traced back to conference and journal publications. The following are our publications that are both relevant to this thesis and form the basis for research detailed in one or more chapters:

- Chapter 4: The components of our proposed news search framework were first published in (McCreadie *et al.*, 2010*b*) and (McCreadie, 2010).
- Chapter 5: Our crowdsourcing methodologies for dataset development have been published in (McCreadie *et al.*, 2010*a*), (McCreadie *et al.*, 2011*a*) and (McCreadie, Macdonald & Ounis, 2012).
- Chapter 6: Our approaches for identifying the most important events of the moment have been published in (McCreadie *et al.*, 2010*c*) and (McCreadie *et al.*, 2011*b*).
- Chapter 8: The blog post ranking approach that we propose was first reported in (Santos, McCreadie, Macdonald & Ounis, 2010). Notably, it produced the best performance of the all systems submitted to the TREC 2010 Blog track.

Furthermore, during the PhD research programme, we also developed a novel information retrieval infrastructure to efficiently tackle large volumes of user-generated content, which both supports our experiments throughout this thesis and that lead to conference and journal publications. In particular, we developed a large-scale distributed MapReduce indexing infrastructure using Hadoop[1] to enable the fast indexing of large-scale user-generated and other collections within the open source Terrier IR platform[2], subsequently published in (McCreadie, Macdonald & Ounis, 2009*a*) and (McCreadie *et al.*, 2011*c*).

1.6 Thesis Outline

In this thesis, we investigate how user-generated content can aid in satisfying news-related queries submitted to Web search engines. In particular, we propose a news search framework to describe the functionalities required to tackle news-related queries in the current fast-paced news search setting. Using this framework as a base, we examine whether user-generated content can be used to more accurately classify queries as news-related and provide better coverage for the user query. The outline of this thesis is as follows:

- Chapter 2 introduces important information retrieval (IR) concepts that are used all throughout the thesis. In particular, we begin by describing the underlying architecture of a basic IR system in terms of its indexing and retrieval components, in addition to more advanced aspects of such systems such as terabyte-scale indexing using the MapReduce paradigm. Furthermore, we discuss the evaluation of IR systems both in general and more specifically within the context of the Text REtrieval Conference (TREC). We then discuss more specialist areas of IR that are used in later chapters in this thesis to tackle particular news search tasks, namely: machine learning; news vertical search; aggregate ranking; and resource selection.

- In Chapter 3, we motivate the use of user-generated content for news-related tasks. We describe how each of five different user-generated content sources, namely: Blogs; Digg; Query-logs; Microblogs and Wikipedia, might be useful either as a source of news-related information and/or as a source of content to return to the user. Finally, we identify the knowledge gap within the literature that this thesis addresses, namely can user-generated content aid in satisfying news-related queries within a universal Web search setting?

[1] `http://hadoop.apache.org/`
[2] `http://terrier.org`

- Chapter 4 presents our framework for news search. It defines the four framework components that are needed to deploy real-time news vertical search, describes the purpose of each component and discusses the challenges that need to be addressed within each. We experimentally investigate each of these four components in Chapters 6, 7, 8 and 9.

- Chapter 5 describes all of the datasets that we use for evaluation of each framework component within their subsequent experimental chapters (Chapters 6, 7, 8 and 9). In particular, it details prior datasets that we use, in addition to the creation and quality assurance of new datasets that we develop using crowdsourcing.

- In Chapter 6, we tackle the first of our framework components, i.e. the identification of the important events of the moment for display to the user. In particular, we propose both learned and unlearned story ranking approaches to rank newswire articles representing events and then test their effectiveness for real-time news story ranking when driven by blog and tweet streams. Finally, we analyse how newswire article ranking differs when using blogs and tweets to drive the ranking process, conclude on the effectiveness of our proposed approaches and whether user-generated content can be used to effectively identify important events in real-time.

- Chapter 7 tackles the second of our framework components, i.e. the automatic classification of news queries using user-generated content. We propose a novel news query classification approach that extracts classification evidence about each query from a set of parallel news and user-generated content streams. We evaluate the effectiveness of this approach on two different datasets, each comprised of multiple streams of news and user-generated content, concluding when and where the addition of user-generated content can increase classification accuracy for news-related user-queries.

- In Chapter 8, we investigate the third of our framework components, i.e. the ranking of user-generated content for a news-related query. In particular, we propose to use machine learning to tackle the unique characteristics two types of user-generated content, namely blog posts and microblogs. We evaluate whether our proposed approaches are more effective than using traditional document weighting models for these unique types of document.

- The last of our experimental chapters, Chapter 9, examines the final component of our framework, i.e. the integration of news-related content into the Web search ranking. We propose both resource selection and heuristic-based techniques to merge newswire articles, blogs, tweets, Wikipedia pages and diggs into the Web search ranking. We perform a crowdsourced study with end-users to evaluate whether these approaches can produce rankings that provided better coverage for the user query, i.e. to conclude when and where end-users want user-generated content added to their Web search results.

- Chapter 10 closes this thesis by summarising the conclusions and outcomes from each of the individual chapters, in addition to providing possible new research directions uncovered by this work.

Chapter 2

Information Retrieval

2.1 Introduction

The field of information retrieval (IR) deals with the representation, storage, organisation of and access to information items (Baeza-Yates & Ribeiro-Neto, 1999). It was created to solve the issues with information access to large heterogeneous collections of information, normally natural language corpora. Research in IR is concerned with efficient storage of information, the efficient retrieval of that information and effectiveness in satisfying the user's information need. A user typically expresses their information need as a query (one or more terms), which an IR system then searches for and retrieves information items that it believes are relevant to the user's information need. These retrieved items are normally presented to the user as a ranked list, where higher ranked items are deemed more likely to be relevant to the user's information need.

The IR process can be loosely split into the following four parts. To begin with, the input corpora must be converted into a data structure that can be easily searched, conventionally called an index (van Rijsbergen, 1979). A user with an information need then formulates a query and provides it to the system in a form that it can understand (conventionally as a natural language text). The IR system matches the information items within the index to the query, selecting those that are conceptually close. We refer to the information items within an index as documents in this work. A weighting model is then applied, which provides the final similarity score between multiple information items (Baeza-Yates & Ribeiro-Neto, 1999). Lastly, the retrieved documents are presented to the user to satisfy their information need. Unfortunately, the user's query may be a poor representation of their information need and as such, even an effective IR system may not return relevant results. As a result of this, in light of the retrieved documents, the user may choose to re-formulate their query to be either more specific or otherwise as needed.

This IR process has subsequently been extended to leverage more advanced document ranking techniques, e.g. Learning to Rank (Liu, 2009), whereby many features of the documents are combined within a learned model to provide more accurate matching to the user query. Other approaches also build upon this basic process to tackle more complex search tasks, e.g. expert search (Macdonald, 2009), or combine many instances of this process over diverse indices to provide composite rankings, e.g. resource selection (Si & Callan, 2003*a*).

Research in IR has focused on the relationship between documents and the queries that should retrieve them. Moreover, advances have been empirically measured through a long history of experimentation to evaluate how a particular search system is performing. Standard evaluation of an IR system involves measuring the performance of the system on a query for which some relevant documents are known. Success can then be measured in terms of how many of these documents where retrieved, and at what ranks. This will be repeated over multiple queries and then an aggregate function used to determine the final effectiveness of the system.

All of the work in this thesis is strongly grounded in IR, and indeed much is built upon or further enhances the traditional IR process to tackle the novel challenges posed by real-time news search. In this chapter, we describe background IR literature that is relevant to both to the news search framework that we will propose in Chapter 4 and to our later experiments. In particular, the remainder of this chapter is structured as follows:

- Section 2.2 provides a background into the indexing process of an IR system. We describe in detail the preparation of documents, indexing strategies and the final index structures produced. We make use of index structures to drive retrieval of both news and user-generated content for the user query in Chapters 6, 7 and 9. Furthermore, we discuss some more advanced aspects of the indexing process in a Web environment, namely: how corpora spanning billions of documents can be efficiently indexed using the MapReduce paradigm; and how noise in web pages can be reduced through boilerplate removal techniques. We use MapReduce to index very large Twitter corpora spanning hundreds of millions of documents (see Table 5.1), while boilerplate removal is used to clean unprocessed documents downloaded from the Web, e.g. blogs.

- Section 2.3 further describes the document retrieval process once an index has been created. In particular, we discuss how users represent their information needs as queries, and how these queries are processed by an information retrieval system. Then we discuss both popular approaches and frameworks for matching documents to each query and techniques for automatically expanding the user query with the aim of increasing search effectiveness. We use these approaches extensively throughout the experimental portions of this thesis (Chapters 6, 7 and 9) both to extract evidence about the documents related to a user-query for classification as well as for the ranking of different types of documents for display to the user.

- In Section 2.4, we detail evaluation methodologies in the field of IR and describe the Text REtrieval Conference (TREC) that is central to the evaluation of multiple of the sub-tasks of the news search framework that we describe in Chapter 4. Furthermore, we describe the document ranking evaluation measures that we use later to evaluate the effectiveness of our news search sub-tasks.
- Section 2.5 introduces the field of machine learning. Machine learning approaches have become increasingly popular in the field of information retrieval, as they enable the automatic learning of effective models from large-scale datasets, e.g. learning to rank (Liu, 2009). Indeed, we use machine learning extensively in this thesis within Chapters 6, 7 and 9.
- In Section 2.6, we provide a background into the field of topic detection and tracking (TDT), that is involved with the monitoring of news events over time. TDT is relevant to the task of top news identification that we tackle in Chapter 6.
- Section 2.7 describes prior work in the field of aggregate search. Aggregate search tasks involve the ranking of groups of documents representing some concepts, instead of ranking individual documents. For example, expert search systems rank experts based upon the documents that they themselves have authored. We propose an aggregate voting approach to tackle the identification of current news stories for the user query in Chapter 6.
- In Section 2.8, we provide an overview of prior work in resource selection, that tackles search over multiple heterogeneous document corpora. We use resource selection techniques in our final experimental chapter to build a news search system combining heterogeneous news and user-generated content for display to the user (see Chapter 9).
- We provide a summary of this chapter in Section 2.9.

2.2 Indexing

To make access to information as efficient as possible, index data structures need to be created that accurately summarise the content of all information items to be searched. All information items considered in this work are comprised of natural language text. The characteristics of information items from different sources can vary greatly, compare a Twitter tweet to a news article for example. For simplicity, we refer to information items generally as *documents* for the remainder of this thesis, but may use more specific terminology where appropriate. The creation of index structures summarising documents is known as indexing. Indexing is used to create index structures for all of the news and user-generated content sources used in this thesis (see Chapter 5). In the following, the indexing process is explained with reference to the example document below, specifically the first paragraph taken from a news article from the BBC news website on the 17/6/2009[1]:

```
``One of the biggest surprises in the Digital Britain report was
 the news that everyone with a fixed line telephone would pay
a broadband tax."
```

2.2.1 Tokenisation

The first stage within the indexing process is known as tokenisation. In this stage, the text is split into chunks based on some boundary condition, normally white-space characters. This is approximately equivalent to splitting a document into its individual words - through non-standard encoding formats within some documents mean that this will not always be the case. At the tokenisation stage, the text is also lower-cased and all punctuation removed. Once tokenised, the above text might be represented as follows (where '—' represents a term boundary):

```
``one|of|the|biggest|surprises|in|the|digital|britain|report|
was|the|news|that|everyone|with|a|fixed|line|telephone|would|
pay|a|broadband|tax"
```

2.2.2 Stopword Removal

In his work on mechanised encoding and searching, Luhn (1957) described how the descriptive power of a word follows a normal distribution with respect to the number of times it is used (its term frequency). This means, the more often a word is used, the less informative it becomes. In particular, the most common words (e.g. the) cannot be used to distinguish between documents because they are contained in nearly all documents. Since these words are therefore not useful, they are known as *stopwords* and are completely removed from the index (Baeza-Yates & Ribeiro-Neto, 1999). Rather than calculate this on a per-collection basis, often a pre-defined list of stopwords is used. Indeed, articles, prepositions and conjunctions are likely candidates to be included. It should also be noted that removing stopwords has

[1]`http://news.bbc.co.uk/1/hi/technology/8105068.stm`

the positive effect of reducing the size of the final index structures. After stopword removal the example document might look like the following:

```
``biggest|surprises|digital|britain|report|news|everyone|fixed|
line|telephone|pay|broadband|tax"
```

2.2.3 Stemming

In natural language, many words have multiple variants (e.g write, wrote, writing, written). These syntactical variations make direct matching between the user's query and the text unlikely to succeed (Baeza-Yates & Ribeiro-Neto, 1999). One solution to this problem is to transform these words down into their root forms - this is known as conflation. Conventionally, this is performed through the process of stemming, where the syntactical suffixes of words are removed based of a set of logical transformation rules. The first stemming algorithm was published by Lovins (1968). Meanwhile, possibly the most well known stemming algorithm is Porter's stemmer (Porter, 1980). After stemming using Porter's stemmer, our example text contains the following:

```
``biggest|surpris|digit|britain|report|new|everyon|fix|line|
telephon|pai|broadband|tax"
```

It is interesting to note that not all words are changed (e.g. "biggest"), and that some words once transformed do not correspond to real English words (e.g. "pai"). This is not an issue however, since the user's query will be similarly transformed such that a mapping can be obtained.

In most IR systems a *bag-of-words* approach is taken to store the final text. In this case, the ordering of terms is ignored, instead, only the number of times a term appears within the text is recorded. The storage structure used for each piece of text (document) is known as a document-posting list. The posting list for our example is shown in Table 2.1.

2.2.4 Index Data Structures

To be useful, the index structures need to be as compact as possible, support fast random access and store meta information about documents. There are various techniques that can be applied to achieve this. As IR systems are typically interested in accessing on a per-term basis (as opposed to a per-document basis), an *inverted index* is normally used (van Rijsbergen, 1979). In an inverted index, each entry is a term-posting list (the inverse of a document-posting list), that lists the documents in which the term occurs.

term	frequency
biggest	1
surprises	1
digit	1
britain	1
report	1
new	1
everyon	1
fix	1
line	1
telephon	1
pai	1
broadband	1
tax	1

Table 2.1: A document-posting list

term	posting-list (List(docid delta, term frequency))
2414	(1,1), (3,2), (24,1)
5735	(1,1), (2,1), (3,2), (5,1)
8523	(1,1), (4,5) (2,1) (5,2) (2,1)

Table 2.2: An example inverted-index

Each document in a collection typically has a unique identifier. This normally takes the form of a string, for example, Web pages might use their universal resource locator (URL). However, to save space, this can instead be represented as a mapping between the unique identifier and a unique number, known as the document-ID (docid). Typically, an integer is used within the inverted index to store the docid, as this will require less storage space than the original string[1]. Then a *document index* is created that holds the mapping between the docids and its associated identifier. A series of small integers can also be used to store the term frequencies in each document.

The size of the inverted index is largely dependent on the number of documents that are indexed. With the growth in collection sizes in recent years, and the need for redundancy of data - to avoid data loss through hard disk failure, which can be difficult or time-consuming to correct - means that the inverted index is often compressed. Index compression provides the core advantage that less storage is needed, lowering the hardware barrier to index these collections, as well as reducing disk input/output (IO) during the indexing process. Compression is commonly achieved by storing only the delta-gaps between each pair of docids (Zobel & Moffat, 2006) in combination with a variable length encoding. In particular, this means that the docids are sorted into ascending order and then the distances between

[1]The author notes that the use of integers as docids may not remain feasible as collection sizes continue to increase. For instance, we estimate that 15 days worth of all Twitter tweets would exceed the number of docids representable using a 32bit positive integer.

docid	Document Identifier	Pointer
2414	biggest	#324fse32
5735	digit	#42a3ee32
8523	surprises	#923567b1

Table 2.3: An example document index

docids are stored, these distances are often much smaller than the original docid. An example inverted index is shown in Table 2.2, while the associated document index is shown in Table 2.3. In this example, the term 'biggest' (2414) appears once in document 1, twice in document 4 and once in document 27. To take advantage of the knowledge that we are storing small integers, an efficient variable length encoding like Elias gamma encoding (Elias, 1975) is used to minimise the amount of space required. Variable length encoding uses a variable number of bits of storage dependent on the size of the integer in question. Note that while we will likely only be storing small integers, this will not always be the case. Indeed, no assumption can be made on the largest integer that will need to be stored. This is why, while a fixed-length encoding might be more efficient, it is avoided (fixed-length values can overflow). The term frequencies, on the other hand, will always be small integers (bounded by the document size in tokens) and so an encoding that is further optimised for storing very small integers like Elias unary encoding (Elias, 1975) is often applied.

Additionally, other structures are also likely to be created to store the extra information needed to perform matching (as described in Section 2.3). These include:

- *Lexicon* : A structure that holds more detailed information about each specific term. This might include the total number of occurrences of the term in the collection or the total number of documents that it appears in. The lexicon links into the main inverted index by providing for each term a pointer to that term's posting list.

- *Direct/forward Index* : The "inverse of the inverted index". This structure holds term information (term id and frequency) for each document in the collection (Ounis, Amati, Plachouras, He, Macdonald & Lioma, 2006). In the same way as the inverted index, compression can be applied to save storage space for this structure. A direct index is used when a listing of all terms contained within a document is required, e.g. when using pseudo-relevance feedback for query expansion (see Section 2.3.5)

- *Meta Index* : This index structure holds additional information about each document indexed, sometimes referred to as document 'meta data'. The meta index facilitates lookups on this information given the docid of the document in question. For example, one might store the time a document was created, or the author of that document in the meta index. In general, the meta index is used to store document information for display later to the user.

These structures form the core of a search engine index (Brin & Page, 1998). Once these structures have been created for the document corpus in question, then the system is ready to service user queries. However, in a large-scale Web search setting, both the structure of the documents to be indexed and the efficiency of the indexing process itself need to be considered. In the following two sub-sections, we discuss these aspects of indexing in a Web context.

2.2.5 Large Scale Indexing and MapReduce

Many common Information Retrieval (IR) tasks like indexing are very resource intensive in terms of both processing time and disk IO. The advent of the Web has brought terabyte-scale, ever-growing corpora of data that are of great research interest. For illustration, Table 2.4 reports the name, publication year and size of popular IR corpora over a 17 year period. From Table 2.4, we see that index sizes have increased dramatically, from hundreds of thousands in 1992 to over a billion in 2009. Efficient IR is therefore a major concern for the designers of both commercial IR products like search engines - where sub-second response times are expected - and for IR researchers who need to execute IR operations such as indexing, parameter training, etc., over large quantities of data quickly - to maximise the number of experiments that can be done. The latter is of special interest currently, as there are now IR test collections spanning billions of documents, e.g. the ClueWeb09 corpus,[1] which was crawled for the Text REtrieval Conference (TREC) in 2009. Indeed, this test collection requires over 25 terabytes of disk space uncompressed to score and spans 1.2 billion documents. Moreover, large collections are not restricted to Web crawls alone. For instance, in this work, we use a corpus that is comprised of over 400 million tweets from the Twitter microblogging platform (see Chapter 5). Furthermore, in recent times, problems are increasingly being solved less by complex theories and more through the unreasonable effectiveness of data (Halevy *et al.*, 2009). In particular, access to large volumes of user-generated data allows for effective models to be learned for many tasks using machine learning techniques, with a high degree of success (see Section 2.5). As such, being able to access large data sources efficiently is of key importance.

As the volume of the data being processed increases, it is clear that the resources (processing power, disk space and memory) needed will likewise increase. However, in terms of hardware, there are two competing viewpoints on how this should be done. These are known as scale-up and scale-out (Davis, 2006). In particular, the scale-up approach involves increasing the resources available on a single large

[1]See `http://trec.nist.gov/call09.html`.

Collection	Year	Docs	Size(GB)
Disks 1-2	1992	740K	2.0
Disks 4-5	1998	500K	1.9
WT2G	1999	240K	2.0
WT10G	2000	1.6M	10.0
.GOV	2002	1.8M	18.0
Blogs06	2006	3M	13.0
.GOV2	2004	25M	425.0
Blogs08	2009	28M	1372.0
ClueWeb09	2009	1.2B	25000.0

Table 2.4: IR corpora in order of increasing size

machine or mainframe, while scale-out mandates the use of many commodity machines, where an increase in resources involves the adding more of those machines. The advantages of mainframes are well known, where strong locality of data and large internal memory stores mean that there are few overheads involved in processing and are ideal for large memory hungry applications (Davis, 2006). Indeed, historically, scale-up was the preferred method for scaling, due to the prevalence of mainframe setups and the use of large databases that could use large amounts of shared memory provided by mainframes. However, in recent years, scale-out approaches have seen an increase in up-take from the commercial sector, due in part to the low cost of commodity hardware and superior performance to cost ratio (Oracle BEA, 2008). Scale-up approaches tend to have little or no effect on software complexity, but are more costly due to the price of the large mainframes required. Scale-out approaches are cheaper, but introduce many challenges in the fields of distributed data storage and parallel processing. In an IR setting, scale-out approaches have been popularised by their adoption by commercial search engines (Dean & Ghemawat, 2004). Hence, in this work, we focus on scale-out approaches to tackle large-scale data,

MapReduce is a scale-out programming paradigm for the processing of large amounts of data by distributing work over multiple processing machines (Dean & Ghemawat, 2004). It was designed at Google as a way to distribute input/output (IO) intensive tasks that are run over large datasets over many machines. It is built on the idea that many tasks that are expensive involve doing a ‘map’ operation with a simple function over each ‘record’ in a large dataset. This operation emits key/value pairs comprising the result. The map operation itself can be easily distributed by running it on different machines, each processing different subsets of the input data. The output from each of these is then collected and merged into the desired result by ‘reduce’ operations.

By using the MapReduce abstraction, the complex details of parallel processing, such as fault tolerance and node availability are hidden in a conceptually simple framework (Manning *et al.*, 2008), allowing highly distributed tools to easily be built on top of MapReduce. Indeed, various companies

Word Counting - Map function pseudo-code

```
1: Input
     Document File-name, Name
     Contents of the Document, Contents
2: Output
     A list of (Word,1) pairs, one for each word in the
     document, Instances
3: for each Word in the Document loop
4:     emit(Word, 1)
5: end loop
```

Word Counting - Reduce function pseudo-code

```
1: Input
     A Word
     List of 1s, Instances
2: Output
     The Word and the size of Instances, Result
3: Integer result = 0
4: for each Integer 'i' in Instances loop
5:     result = result + i
6: end loop
7: emit(Word, Result)
```

Figure 2.1: Pseudo-code for the word-counting operation, expressed as map and reduce functions.

have developed tools to perform data mining operations on large-scale datasets on top of MapReduce implementations. Google's Sawzall (Pike *et al.*, 2005) and Yahoo's Pig (Olston *et al.*, 2008) are two such examples of data mining languages. Microsoft uses a distributed framework similar to MapReduce called Dryad, which the Nebula scripting language uses to provide similar data mining capabilities (Isard *et al.*, 2007). However, it is of note that MapReduce trades the ability to perform code optimisation (by abstracting from the internal workings) for easy implementation through its framework, meaning that an implementation in MapReduce is possibly not the optimal solution. On the other hand, by making abstractions, the resultant software will often be cheaper to produce and maintain (Johnson, 1997).

MapReduce is designed from a functional programming perspective, where functions provide definitions of operations over input data. A single MapReduce job is defined by the user as two functions. The map function takes in a key/value pair (of type <key1, value1>) and produces a set of intermediate key/value pairs (<key2, value2>). The outputs from the map function are then automatically grouped by their key, and then passed to the reduce function. The reduce task merges the values with the same key to form a smaller final result. A typical job will have many map tasks that each operate on a subset of the input data, and fewer reduce tasks, which operate on the merged output of the map tasks. Map or reduce tasks may run on different machines, allowing parallelism to be achieved. In common with functional programming design, each task is independent of other tasks of the same type, and there is no global state, or communication between maps or between reduces.

Counting word occurrences in a large data-set is an often-repeated example of how to use the MapReduce paradigm[1] (Dean & Ghemawat, 2004). For this, the map function takes the document file-name (key1) and the contents of the document (value1) as input, then for each word in the document emits the word (key2) and the integer value '1' (value2). The reduce then sums up all of the values (many 1s) for each key2 (a word) to give the total occurrences of that word. Figure 2.1 provides the pseudo-code for the word counting map and reduce functions. Note that this is a very simple example for illustrative purposes.

As mentioned above, MapReduce jobs are executed over multiple machines. In a typical setup, data is not stored in a central file store, but instead replicated in blocks (usually of 64MB) across many machines using a distributed file system (DFS) (Ghemawat *et al.*, 2003). This has a central advantage is that the map functions can operate on data that may be 'rack-local' or 'machine-local' - i.e. does not have to transit intra- and inter-data centre backbone links, and does not overload a central file storage service. Therefore high bandwidth can be achieved because data is always as local as possible to the processing CPUs. Intermediate results of map tasks are stored on the processing machines themselves. A central master machine provides job and task scheduling, which attempts to perform tasks as local as possible to the input data.

In this thesis, we use MapReduce to index large document collections (see Chapter 5). In particular, during the time period of this thesis, we developed an efficient MapReduce indexing strategy and implemented it within the Terrier IR platform (McCreadie, Macdonald & Ounis, 2009*a*,*b*) using the Java implementation of MapReduce – Hadoop[2]. In particular, under this strategy, the indexing process is split into multiple map tasks. Each map task operates on its own subset of the corpus. In particular, as documents are processed by each map task, compressed posting lists are built in memory for each term. However, when memory runs low or all documents for a map task have been processed, the partial index is flushed from that map task, by emitting a set of <term, posting list> pairs for all terms currently in memory. These flushed partial indices are then sorted by term, map and flush numbers before being passed to a reduce task. As the flushes are collected at an appropriate reduce task, the posting lists for each term are merged by map number and flush number, to ensure that the posting lists for each term are in a globally correct ordering. The reduce function takes each term in turn and merges the posting

[1]A worked example and associated source code is available at `http://hadoop.apache.org/core/docs/r0.19.0/mapred_tutorial.html`

[2]`http://hadoop.apache.org/`

MapReduce Indexing -
Map function pseudo-code

```
1: Input
        Key: Document Identifier, Name
        Value: Contents of the Document, DocContents
2: Output
        Key: Term
        Value: Posting list
3: for each Term in the DocContents loop
4 :     Stem(Term)
5 :     deleteIfStopword(Term)
6 :     if (Term is not empty) then add the current
        document for that term into the in-memory
        Posting List for that term
7: end loop
8: Add document to the Document Index
9: if (lastMap() or outOfMemory()) then
        for each Term in the in-memory Posting Lists
        emit(Term, Posting List)
10: if (lastMap()) write out information about the
11:     documents this map processed ("side-effect" files)
```

Figure 2.2: Pseudo-code for the Terrier indexing strategy Map function.

lists for that term into the full posting list, as a standard index. Elias-Gamma compression is used as in non-distributed indexing to store only the distance between docids. Figures 2.2 and 2.3 provides a pseudo-code implementation of map and reduce functions for this indexing strategy. Through experimentation over four TREC collections, we have shown that this implementation scales efficiently as the index size and resources allocated are increased (McCreadie *et al.*, 2011*c*).

2.2.6 Document Boilerplate

Modern web-pages have much in the way of surplus content attached to them. This comes in the form of document object model (DOM) elements (W3C, 2009) like navigation bars, page scripts (e.g PHP Group (2009)), advertisements or banners (Adar *et al.*, 2009). Indeed, the main content of an article may take up less than 50% of the screen real-estate. Importantly, this might have a significant impact when performing retrieval on these documents, for while it is easy for humans to make the distinction, it is much more challenging to identify non-relevant text in an automated fashion. We use boilerplate removal in this thesis to clean documents such as news articles that we crawl from the Web (see Section 5). Indeed, it is important to clean raw news articles since they typically contain links, to both similar articles and to other popular articles for that day. Conceptionally, there is no way to tell that those terms on their own are not representative of the document, since they are generally very infor-

```
MapReduce Indexing -
Reduce function pseudo-code
1: Input
      Key: A *Term*
      Value: List of Posting List, *PartialPostingLists*
2: Output
      Key: Term
      Value: Posting List
3 : List Posting-List = new PostingList()
4 : Sort PartialPostingLists by the map and flush
        they were emitted from
5 : for each PostList in *PartialPostingLists* loop
6 :     for each doc-ID in *PostList* loop
7 :         correct doc-ID
8 :         Merge PostList into Posting-List
9 :     end loop
10: end loop
11: emit(Posting-List)
```

Figure 2.3: Pseudo-code for the Terrier indexing strategy Reduce function.

mative terms overall. In general, retrieval performance is degredated because duplicated text between articles in the form of links, advertisements, etc, make the documents more similar than they should be. There are many possible approaches to this problem, including, sequence classification (Rosenfeld *et al.*, 2008), page comparison (Nam *et al.*, 2009), near-duplicate detection (Huang *et al.*, 2008) or change analysis (Adar *et al.*, 2009). Below, we describe two example approaches.

One solution might be to remove the additional text beforehand during indexing, on a similar pretext to stemming as described in Section 2.2.3. Nam *et al.* (2009) describes a very simple approach to achieving this in their paper on boilerplate removal in blogs. They just remove HTML formatting and then split the file on carriage returns. Relevant content, then, is the content in page p, which has changed since page p-1. Note that this assumes that the document collection is ordered by website, as comparison will work only against other pages from the same site, since inter-site structure can be incredibly varied. It is also true to say that not all surplus text is removed, because some text is automatically generated when the page is visited, for example the date.

Another solution that has been alluded to by Adar *et al.* (2009) in their paper on understanding the dynamics of Web content, is to remove page content that either has not changed since the page was originally created, or changes with abnormal frequency. In particular, this should remove static content like HTML structure because it does not change over time, but will also remove advertising, because not only does advertising change frequently, but it may be randomised, meaning that the page will change each time it is visited. However, this approach, while useful in theory, is difficult to put into practice, as it mandates the sampling of pages frequently over a period of time. Not only is this expensive in terms of time, bandwidth and storage space, but is also impractical for a large number of pages. Moreover, this approach is incompatible with the collections that we use later in this thesis, as we have only one copy of each document.

2.3 Document Retrieval

The goal of an IR system is to return a ranked list of documents in response to a user's query. Typically this is done in decreasing order of *predicted relevance*. This means that a superior system will return more relevant documents and less non-relevant documents. Moreover, while many documents may be relevant, some may be more relevant than others, and so should be ranked higher. For comparison, this is substantially different to the task of data retrieval that aims is to return all information items that satisfy a clearly defined condition (van Rijsbergen, 1979). This sets itself apart from IR in that a single erroneous item returned constitutes a complete failure for the system (Baeza-Yates & Ribeiro-Neto, 1999), while IR attempts to effectively satisfy an information need by returning top ranked items which are relevant. As such, only the results the users look at are judged to be correct or otherwise. Hence, while returning a perfect ranking may be of interest in limited cases, a few irrelevant items, especially further down the ranking can be safely ignored.

In recent years other qualities like *diversity* (how different are the top documents) (Clarke *et al.*, 2008) or *recency* (how fresh are the top documents) (Ounis *et al.*, 2011) are also being taken into account. Historically, there are two distinct metrics via which IR systems are judged. The most commonly considered is *effectiveness*, i.e. that the system should provide the documents that are most likely to satisfy the user's information need. However, sometimes in conflict with this, an IR system must also be efficient, i.e should respond to the user quickly. Indeed, sub-second response times are mandatory for the most common class of IR system in use today, i.e. Web search engines (Büttcher & Clarke, 2005).

Ranking of documents is performed through the use of a *document weighting model*. This model takes in the user's query and a document, producing a score for that document. This score is a prediction of the *relevance* for that document to the user's query (Baeza-Yates & Ribeiro-Neto, 1999). It is important to note that relevance is subjective, varying from user to user (Voorhees *et al.*, 2005). As such, no document weighting model can be perfect. Indeed, some document weighting models may be designed for certain types of queries (e.g. short queries) or for specific tasks (e.g. homepage finding).

Some document weighting models are based on the probability ranking principle (PRP) (Robertson, 1977), where the score assigned to a document is the probability that the document is relevant based on the information available. A document weighting model calculates a document's score based on a set of *features*. These features are qualities possessed by either the document being scored, the query being made, or in some cases the entire collection. It should also be noted that to make the documents held in the inverted index and the query comparable, they must be in the same format. To this end, the query will be subjected to the same processing (tokenisation, stopword removal and stemming) that the documents experience during the indexing process. In the following subsections we describe the document weighting models that we use later in this thesis.

2.3.1 Term Frequency - Inverse Document Frequency (TF-IDF)

Different document weighting models have been proposed within the literature. Each document weighting model combines features of the document and query together to form a weight for that document/query pair. The most common feature that is used in practice is the term frequency (tf) (Salton, 1971). This is the number of times in the document each term in the query appears. This models the intuition that a document that contains many of the words in the query will be about that query. Often however, a feature can be misleading dependent on the context in which it appears. Consider term frequency - the more times a term appears within a document, the more the document is concerned with that term. However, just because the most common word in the document happens to be 'the', doesn't mean that the document is about 'the'. To mitigate this problem, often features will be *weighted*. The most well known application of weighted term frequency is the TF-IDF model (Salton, 1971), which scores a document d based on query Q as shown below:

$$score(d, Q) = \sum_{tinQ} tf.log_2 \frac{N}{N_t} \tag{2.1}$$

where tf is the term frequency for term t, N is the number of documents in the collection and N_t is the number of documents that contain t. Importantly, the $log_2 \frac{N}{N_t}$ part is known as the inverse document frequency (IDF), which acts as the weighting factor (Spärck-Jones, 1972). Another interesting point to raise is that document length is not fixed. Should a document be very long, then it is likely that tf will over-emphasise terms because long documents contain many repeated terms and hold a large vocabulary (Singhal *et al.*, 1996). To adjust for this length bias, tfs are often *normalised*. We use TF-IDF as a feature about document-query pairs in Chapters 6 and 7.

2.3.2 The Best Match Weighting Model

Some of the most popular document weighting models in use today come from the Best Match (BM) family of weighting models. These models were created through the successful combination of PRP and the 2-Poisson indexing model (Harter, 1975). 2-Poisson is based on the idea that the distribution of within-document frequencies for certain words is Poisson for the elite documents, and also (but with a different mean) for the non-elite documents. This is where an elite document for a term is one in which that term occurs more often that in the rest of the collection, i.e. for each term, there is an elite set of documents and a non-elite set of documents. In particular, the original formulation of best match calculates the weight of a term t in a document based on the total number of documents in the collection N, the number of documents that the term appears in, N_t, the number of relevant documents which contain the term, r, and the number of relevant documents to the query, R, (Robertson *et al.*, 1981) as shown below:

$$w = log\frac{(r+0.5)/(R-r+0.5)}{(N_t-r+0.5)/(N-N_t-R+r+0.5)} \tag{2.2}$$

This, therefore, emphasises those terms which are contained in many relevant documents. However, often there is no relevance information available. Therefore, the equation was simplified into the following form (Croft & Harper, 1988):

$$ws = log\frac{N-N_t+0.5}{N_t+0.5} \tag{2.3}$$

This is similar to the well known IDF component of the TF-IDF weighting model described previously. Robertson *et al.* (1981) also modelled a term frequency component for this equation. The popular BM25 model is a further refinement of this which takes term frequency into account, while applying the appropriate normalisation based on the average length of all documents avg_l, as shown below:

$$score(d,Q) = \sum_{t\in Q} w^{ws}.\frac{(k_1+1)tfn}{k+1+tfn}.\frac{(k_3+1)qtf}{k_3+qtf} \tag{2.4}$$

where the normalised term frequency tfn is given by:

$$tfn = \frac{tf}{1+b+b.\frac{l}{avg_l}} \tag{2.5}$$

In this case b, k_1 and k_3 are variables, typically where b=0.75, k_1=1.2 and k_3=1000 (Robertson *et al.*, 1994). Note that the score for a document is the linear sum of the weights of its constitute terms. BM25 is popular because it has been shown to be effective (Büttcher & Clarke, 2005). However, one major issue with BM25 is that it can produce negative term weights for terms that appear in more than half the documents in the collection. In practice, this issue can be avoided through the correct application of stopword removal (Manning *et al.*, 2008), as by definition, words that appear in over half of all documents are unlikely to be useful. We use BM25 as a baseline ranking document ranking approach for different news and user-generated content sources in Chapter 9.

2.3.3 Divergence From Randomness

The Divergence from Randomness (DFR) framework for document weighting was proposed by Amati. (2003). It is a generalisation of the 2-Poisson model described earlier, specifically stating that the more a term's distribution in a single document is different to the random distribution (its distribution in the entire collection) then the more informative that term is. In any algorithm based on DFR there are three components: Inf_1 - the randomness model; Inf_2 - the after-effect; and the normalisation of the term frequency denoted tfn. The randomness model defines how informative a term t is, while the after-effect models the special case where we encounter rare terms that are particularly informative based upon the elite set of documents. Like the BM25 family of models, DFR generates the score for a document as the sum of the weights of the terms as below:

$$score(d,Q) = \sum_{t \in Q} qtw \cdot Inf_1 \cdot Inf_2 \tag{2.6}$$

where qtw is the normalised query term frequency of t, Inf_1 is calculated as:

$$Inf_1 = -log_2(prob_1(tfn|Collection)) \tag{2.7}$$

and Inf_2 is calculated as:

$$Inf_2 = 1 - prob_2(tf|E_t) \tag{2.8}$$

The probability $prob_1$ is the chance that the term frequency would be tfn by chance, thus is the measure of randomness. The less chance that tfn has occurred randomly, the more informative the term. Probability $prob_2$ meanwhile, is a function that models the information gain for the term based on the elite set of documents. In this thesis, we use two models generated from the DFR framework. In particular, we use the DPH and DFReeKLIM weighting models. DPH calculates the score for a document as shown below:

$$DPH(d,Q) = \textstyle\sum_{t \in Q} qtw \cdot \left((1-f) \cdot \frac{1-f}{tf+1}\right) \cdot \left(tf \cdot log_2((tf \cdot \frac{avg_l}{l}) \cdot \frac{N}{n}) + 0.5 \cdot log_2(2 \cdot \pi \cdot tf \cdot (1-f)\right) \tag{2.9}$$

where tf is the term frequency in the document, l is the length of the document, f is tf divided by l, avg_l is the average length of all documents in the corpus, N is the number of documents in the corpus and n is the number of documents containing t in the corpus. We use DPH to produce document rankings from news and user-generated content sources in Chapters 6, 7, 8 and 9. DFReeKLIM on the other hand is calculated as follows:

$$DFReeKLIM(d,Q) = \sum_{t \in Q} qtw \cdot tf \cdot log_2 \left(\frac{tf+1}{l+1} / f \right) \cdot log_2 \left(f / \frac{TF}{T} \right) \tag{2.10}$$

where T is the number of tokens in the collection and TF is the total number of times that the term t appears in the corpus. DFReeKLIM is notable in that it uses a document length normalisation designed to be more effective when applied upon short texts. In particular, $\frac{1}{l}$ is used since $\frac{avg_l}{l}$ can be misleading when avg_l and l are small, i.e. a small variance in l can cause a large change in the final score. We use DFReeKLIM to rank short Twitter tweets in Chapters 6, 7 8 and 9.

2.3.4 Language Modelling

Language modelling takes a slightly different approach to ranking documents. Instead of directly calculating the relevance of a document given a query, we instead calculate the probability of the document generating the query (Ponte & Croft, 1998). This is a measure of how well the document 'fits' the query (Berger & Lafferty, 1999) and is formally defined below:

$$p(d|Q) = \frac{p(Q|d)p(d)}{p(Q)} \tag{2.11}$$

This is where $p(d|Q)$ is the probability of the document given a query (relevance), and $p(Q|d)$ is the probability of the document generating the terms in the query, assuming those terms are independent:

$$p(Q|d) = \prod_{t \in Q} p(t|Q)^{n(t,Q)} \tag{2.12}$$

where $n(t,Q)$ is the number of times term t appears in the query. It should be noted that $p(t|d)$ (the probability of a term given a document) is calculated from the underlying model of the document, however there can be a problem with sparsity. Indeed, any document that does not contain the term cannot be returned. This is solved by smoothing the results, which involves combining the document model with the collection model (model of all documents) so that zero probabilities are removed (Croft & Lafferty, 2003). In this thesis we use a language modelling approach proposed by Zhai & Lafferty (2004), which uses Bayesian smoothing with a Dirichlet Prior. We refer to this approach as DirichletLM. DirichletLM, given a document and query, returns a score for that document as shown below:

$$score(d,Q) = \prod_{t \in Q} p(t|Q)^{n(t,Q)} \propto \sum_{t \in Q} n(t,Q) \cdot log_2 \left(1 + \frac{tf}{c \cdot (TF/T)} \right) + log_2 \left(\frac{c}{l+c} \right) \tag{2.13}$$

where c is a smoothing parameter (set to 2500 in this work); tf is the term frequency of term t from the query Q in document d; l is the length of d in tokens; TF is the term frequency of t in the entire collection; and T is the total token count for the corpus. We use DirichletLM as a baseline ranking document ranking approach for different news and user-generated content sources in Chapter 9.

2.3.5 Query Expansion and Pseudo Relevance Feedback

As noted earlier, the query used to represent the user's information need is often short, indeed 2-3 terms in length on average (Silverstein *et al.*, 1999). As a result of this, short user queries can be considered to be underspecified, leading to issues of vocabulary mismatch between documents and queries. For example, consider the two queries 'Current U.S. president' and 'Barrack Obama'. Both of these queries mean the same thing (at the time this thesis was written), although they share no terms in common. Hence, the first query will not return documents that talk only about Barack Obama, even though such documents would be relevant.

A classical IR technique for tackling this issue and hence for improving retrieval effectiveness, is query expansion. The central idea behind query expansion is that additional terms should be added to the user's original query, such that a wider range of related documents will be returned. One query expansion technique is pseudo-relevance feedback (PRF) (Salton, 1971). Under PRF, from an initial ranking of documents, the top returned documents are assumed to be relevant, and information from these 'pseudo-relevant documents' is used to refine the query. In a query expansion context, the most informative terms from the pseudo-relevant documents – selected via a term weighting model – are used to expand the initial query. The intuition is that the new expanded query will retrieve more relevant documents than the initial query, and at higher ranks. Notably, both the original query terms and the new terms used for expansion may be weighted during this process, such that the most informative terms receive the highest weight.

In this work, we use DFR-based term weighting models for query expansion. The basic idea of these term weighting models is to measure the divergence of a term's distribution in a pseudo-relevance set from its distribution in the whole corpus. The higher this divergence is, the more likely the term is related to the query topic. Below we list two term weighting models based upon the Divergence from Randomness framework (Amati., 2003).

Bo1 is a term weighting model based upon the Bose-Einstein statistics. Under Bo1, the weight of a term t in the top-ranked documents is given by:

$$w(t) = tf_x \cdot \log_2 \frac{1 + P_n}{P_n} + \log_2(1 + P_n) \qquad (2.14)$$

where the number of expansion documents selected normally ranges from 3 to 10 (Amati., 2003). P_n is given by $\frac{F}{N}$, F is the frequency of the term t in the corpus, and N is the number of documents in the corpus. tf_x is the frequency of the query term t in the top-ranked documents retrieved. The weights for each of the original query terms qtw are calculated as follows:

$$\begin{aligned} qtw &= \frac{qtf}{qtf_{max}} + \frac{w(t)}{\lim_{F \to tf_x} w(t)} \\ &= F_{max} \log_2 \frac{1 + P_{n,max}}{P_{n,max}} + \log_2(1 + P_{n,max}) \end{aligned} \tag{2.15}$$

where $\lim_{F \to tf_x} w(t)$ is the upper bound of $w(t)$. $P_{n,max}$ is given by F_{max}/N. F_{max} is the frequency F of the term with the maximum $w(t)$ in the top-ranked documents. If a query term does not appear among the most informative terms from the top-ranked documents, its query term weight remains equal to the original one. Note that there is also another implicit query expansion parameter, i.e. the number of terms with which to expand the original query. Typically, this is set to 10 terms (Amati., 2003).

A second term weighting model is based on the Kullback-Leibler (KL) divergence measure. KL measures how each candidate expansion term's distribution is different to the random distribution. Using the KL model, the weight of a term t in the feedback document set D is given by (Amati., 2003):

$$w(t) = p(t|D) \cdot \log_2 \frac{p(t|D)}{p(t|Coll)} \tag{2.16}$$

where $p(t|D) = tf_x/c(D)$ is the probability of observing the term t in the feedback document set D, tf_x is the frequency of the term t in the set D and $c(D)$ is the number of tokens in this set. $p(t|Coll) = TF/c(Coll)$ is the probability of observing the term t in the whole corpus, TF is the frequency of t in the corpus, and $c(Coll)$ is the number of tokens in the corpus.

Using the KL model, the query term weight qtw is calculated as before (see Equation 2.15), while the upper bound of $w(t)$ is given by:

$$\lim_{F \to tf_x} w(t) = \frac{F_{max} \cdot \log_2 \frac{c(Coll)}{l_x}}{l_x} \tag{2.17}$$

where F_{max} is the corpus frequency F of the term with the maximum $w(t)$ in the top-ranked documents, l_x is the length of the feedback documents, and *c(Coll)* is the number of tokens in the whole corpus. Note that the DFR query expansion framework is similar to Rocchio's relevance feedback method (Salton, 1971). The difference is that the former considers the whole feedback document set as a bag of words, while the latter averages term weights over single feedback documents.

An alternative query expansion technique is corpus enrichment (Diaz & Metzler, 2006; Kwok. & Chan, 1998; Macdonald *et al.*, 2005). Under corpus enrichment (CE), PRF is performed using an external, higher quality corpus, rather than the same document corpus that is being searched. In particular, documents from an external corpus, e.g. Wikipedia, are retrieved for the query. Informative terms selected from those documents are then used to expand the original query. The expanded query is then used to retrieve the final ranking of documents from the target corpus. As with traditional pseudo-relevance feedback approaches, the idea is that the expanded query will lead to better retrieval performance than the initial query.

We use both pseudo-relevance feedback query expansion and collection enrichment in this thesis to improve retrieval performance when retrieving documents from user-generated content sources. In particular, we use collection enrichment in Chapter 6 to improve the representation of each news story that we rank. We use pseudo-relevance feedback to improve the ranking of tweets for display to the user, in Chapter 9.

2.4 Evaluation

IR has a long history of improvement through experimentation. Evaluation of an IR system typically involves measuring the effectiveness of the system on a query for which some relevant documents are known. Success can then be measured in terms of how many of these documents where retrieved, and at what ranks. This will be repeated over multiple queries and then an aggregate function used to determine the final effectiveness measures for the system.

2.4.1 Evaluation Measures

A large number of evaluation measures have been proposed to evaluate IR tasks. Below we describe the main evaluation measures used in this work.

2.4.1.1 Precision and Recall

Initially, it is important to have a metric with which to measure performance. In IR, two different but related measures are used: precision and recall. Precision is the proportion of retrieved documents that are relevant to the query, while recall is the fraction of the documents that are relevant to the query that were successfully retrieved. In particular, precision measures how ‘good’ our returned documents are, calculated (using Table 2.5) as $[1]/([1]+[3])$. Recall on the other hand, measures how many of the correct documents we returned, in comparison to how many there were - calculated (using Table 2.5) as $[1]/([1]+[2])$. It is of note that precision and recall are also used in a classification context as well as for ranking. In this case, precision and recall are calculated in the same manner as above, however the meaning of entry in the Table 2.5 confusion matrix differs. In particular, [1] refers to the number of true positives, [2] the number of false positives, [3] the number of false negatives, and [4] the number of true negatives.

	Returned	Not-Retrieved
Relevant	[1]	[2]
Not-Relevant	[3]	[4]

Table 2.5: Document ranking/classification confusion table for calculating precision and recall.

Furthermore, metrics combining both precision and recall have also been proposed. In this thesis, we use one of these metrics, namely F_1 (Rijsbergen, 1979), to report overall precision and recall in a classification context. F_1 is calculated as follows:

$$F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}. \tag{2.18}$$

2.4.1.2 Mean Average Precision

One popular evaluation measure in an IR setting is the Mean Average Precision (MAP) measure. This is measure calculates the mean of the average precision (AP) values for all queries. The AP for a query is the average of all precision values calculated after each document is retrieved (Voorhees, 2003). It is notable that MAP is a top-heavy measure, i.e. documents ranked correctly near the top of the ranking contribute more to MAP performance than documents ranked near the bottom. The MAP is calculated as follows:

$$MAP = \sum_{q=1}^{Q} \frac{\sum_{k=1}^{n} Precision(R(q), k) \cdot Recall\delta(R(q), k)}{|Q|} \tag{2.19}$$

where $|Q|$ is the number of queries, n is the number of retrieved documents, k a the rank within the retrieved documents $R(q)$, $Precision(R(q), k)$ is the precision at cut-off k and Recallδ(R(q),k) is the change in Recall between ranks $k - 1$ and k.

2.4.1.3 (Normalised) Discounted Cumulative Gain

Mean Average Precision (MAP) has been used for many years to measure the effectiveness (Voorhees, 2003) of IR systems. However, as it is built upon the precision metric, which is binary in nature, i.e. each document is considered relevant or not, it is not useful when documents are evaluated with respect to multiple relevance grades. For example, a document might be considered in terms of a 3-grade system: highly relevant; relevant or not relevant. To evaluate tasks that use more granular evaluation labels, Discounted Cumulative Gain (DCG) metrics were proposed (Järvelin & Kekäläinen, 2002). These approaches assume that documents with higher relevance grades are more relevant than those with lower grades and that highly relevant documents are most useful when returned in the top ranks. Hence, DCG measures are compatible with multi-graded assessments and are top heavy in nature. DCG is calculated as follows:

$$DCG = rel_1 \sum_{i=2}^{p} \frac{rel_i}{log_2 i} \tag{2.20}$$

where rel_i is the relevance of the document at rank i.

However, DCG is not sufficient for a typical IR evaluation, because not all result sets for a query are of the same length, i.e. fewer documents than the rank cutoff may be retrieved (Järvelin & Kekäläinen, 2002). To account for this, Normalised Discounted Cumulative Gain (nDCG) was proposed, that normalises the cumulative gain across queries. This is achieved by dividing DCG by the ideal DCG, i.e. that which would have been achieved by the perfect ranking according to the relevance assessments.

2.4.2 Cranfield Paradigm

Evaluation in IR is driven by the need to provide better rankings of documents and hence focuses on how to distinguish between different document rankings for the same queries. Classical IR evaluation has centred around human relevance judgements. This is where a human looks at a document given a query and marks it relevant or not to that query. Many of these judgements are then combined together to form a *test collection*. This consists of the relevance judgements, the corpus of documents and the queries used. These test collections can then be used by multiple systems to generate a ranking for each query that can be compared to the relevance judgements. This approach was first pioneered by the Cranfield experiments. It is important to note that the Cranfield experiments used full relevance judgements, where each document was assessed for each query (Cleverdon, 1991). Since then, the corpora of documents that are used have become sufficiently large that full relevance judgements are not feasible. For example, the largest test collection currently available is the ClueWeb09 corpus, containing 1.2 billion documents (see Table 2.4). Indeed, even for a set of 50 queries, where an assessor is assigned a minute per document, it would take over 95,000 years to finish assessing. Instead, incomplete assessments are used, where only a very small proportion of the corpus is judged. However that small portion is carefully selected to contain as many relevant documents as possible.

Typically, a strategy known as *pooling* is used to select the documents to be judged (Sparck-Jones & van Rijsbergen, 1975). The aim of pooling is to create a high-recall sample of the collection. Under pooling, different IR systems rank documents for the query topics. The individual rankings from each system are combined to create a single ‘pool’ of candidate documents to be assessed. Ideally, all of the documents within this pool should be judged. However, the pool size may still exceed the resources available to judge it. In this case, the pooled documents are then ranked in some manner and then assessed in rank order until the assessment resources are exhausted or a rank cut-off has been reached.

2.4.3 TREC

The Text REtrieval Conference (TREC) is a collection of IR workshops sponsored by the National Institute of Standards and Technology (NIST) and the Disruptive Technology Office of the U.S. Department of Defence. Unlike standard conferences in IR however, TREC was designed to encourage evaluation within the IR community by providing the infrastructure necessary for large-scale evaluation. TREC each year runs a number of *tracks*, which each represent a topic of research. A single track will contain one or more tasks that IR groups can participate in, by providing an afore-mentioned test collection suitable for evaluating that task. At TREC, the test collections are created using *pooling* (Sparck-Jones & van Rijsbergen, 1975). All the participating groups provide initial document rankings using the corpus and queries, known as *runs*. Only the top n (normally 100) documents in each of these runs are judged by human assessors, which are then merged to create the relevance judgements. The idea is that by using runs from multiple diverse IR systems, the final relevance judgements will not be biased toward any single system or algorithm. Also, these judgements should be complete enough to judge systems that did not submit runs during the initial phase, since (hopefully) most of the relevant documents will have been identified and assessed. The assumption is that by using multiple IR systems, the probability of selecting the most relevant documents will be high.

2.4.4 Relevance Assessment

To assess approaches for various tasks, it is important to have a ground truth that can be compared against. Typically, this involves employing human assessors to judge documents from the pool to determine their true relevancy for the query that each was retrieved for. However, assessing large numbers of documents is time consuming and expensive. For example, if a document takes 30 seconds to assess (Voorhees *et al.*, 2005), then to judge each of the 19,381 pooled documents for the TREC 2011 Web track, for example, will take in excess of 161 man-hours, or 6.7 weeks for a single assessor (assuming a 7 hour working day). Indeed, assuming a national minimum wage of \$7.25 (US dollars) per hour, the cost of recreating the TREC Web track relevance assessments totals \$1,170.88. TREC, sponsored by NIST, has traditionally paid a group of specialist assessors to judge documents for the participants (Voorhees *et al.*, 2005).

However, NIST has a limited amount of funds to support the tracks that it runs. Indeed, in rare cases, TREC tracks have been known to have used the participants to judge documents if NIST could not supply sufficient funding (Macdonald, Soboroff & Ounis, 2009). However, such an approach is limited, as the number of documents that can be judged is determined by the number of participants. Furthermore, the size of the document pools used to assess systems, in comparison to size of the collections examined, i.e. the completeness of produced relevance assessments, has been diminishing almost year-on-year (He, Macdonald & Ounis, 2008). This violates the completeness assumption of TREC-style assessment (Voorhees *et al.*, 2005) to an ever greater degree, increasing the probability of error during evaluation. Furthermore, for novel tasks or when evaluating on new collections, relevance assessments may not (as yet) be available, hence the task of creating these assessments falls to the researcher. Later in this thesis we describe how we created relevance assessments to evaluate our news search tasks using crowdsourcing (see Chapter 5).

2.5 Machine Learning for IR

Machine learning refers to the field of approaches that automatically learn solutions to problems using prior data (Carbonell, 1990). Machine learning has become closely linked with information retrieval, as many tasks in information retrieval can be formulated in a manner that can be tackled by machine learning approaches, e.g. categorising documents (Yu *et al.*, 2002) or learning how to rank documents (Liu, 2009). Moreover, machine learned approaches have shown to be effective for many of these tasks (Agichtein *et al.*, 2006; Arguello *et al.*, 2011; Dai *et al.*, 2011; Kang *et al.*, 2011; Zeng *et al.*, 2004). Indeed, commercial Web search engines like Google and Bing use machine learned models to drive their search rankings.

An important concept within machine learning is that of a *feature*. A feature is some property about the subject of the learning. For example, for information retrieval ranking problems, the features might be about the documents or user queries. An example of a query feature is query length, while a document feature might be its PageRank (Page *et al.*, 1999) score.

In this thesis, we use two different types of machine learning, associated to two different tasks. In particular, we use machine learning for query classification and for document ranking. In the following two sub-sections, we describe machine learning with respect to these two tasks.

2.5.1 Classification

Classification approaches tackle problems that require the labelling of instances into two or more distinct classes. For example, Web page classification involves the labelling of Web pages into pre-defined categories, e.g. personal homepages, resume pages, etc. (Yu *et al.*, 2002). Classification is a supervised machine learning problem, i.e. the classifier uses a training set containing instances whose class is already known. This classifier builds a model using the training instances that can be then be used to estimate the class of new, unseen instances. This is done by first extracting a fixed set of features about each instance, then the learner uses the training data to identify and combine the features with discriminative power, i.e. those that are useful for discriminating between instances of each class. The final combination of features is referred to as the *classification model*.

There are a variety of different approaches proposed within the literature to learn a classification model. These can be summarised as, decision trees, rule-based learners, percepteron learning, statistical learning, instance-based learning and support vector machines (SVM) (Kotsiantis *et al.*, 2007). Decision trees, as their name suggests, build a tree-like structure, where features are used to make a decision at each branching point and leaf-nodes are the resultant classes. An example of a decision tree algorithm is C4.5 by Quinlan (1993). Rule-based learners are similar, in that they construct rules (combinations of features) that are each comparable to a single path through a decision tree. However, rule-based learners directly induce the rules from the training instances, rather then building tree-like structures (Fürnkranz, 1999). In contrast, single or multi-class percepteron learners build vectors of percepterons (Ivakhnenko, 1975), where each percepteron outputs a binary decision based upon a threshold for an input feature. Such a learner is trained by varying the percepteron thresholds until a vector that produces the correct result for all training instances is found (Kotsiantis *et al.*, 2007). Statistical learning approaches differ from the prior approaches described in that they define an explicit probability model to describe how an instance is related to each class. In particular, such approaches typically produce a probability estimate that each instance belongs to each class. The most well-known type of statistical learning approach is a Bayesian network, that defines a directed acyclic graph, where each node corresponds with an input feature and connections represent influences between the features (Jensen, 1996). Instance-based learning approaches, also known as lazy learners, are a special type of learning approach that avoids training a model beforehand. Instead, they store the training instances directly, and then compare new instances to the training set to find the closest match. The most well-known instance based approach is nearest-neighbour search (Aha, 1997). Finally, support vector machines represent each training instance in vector space and attempt to partition this space into distinct classes using hyperplanes (Vapnik, 2000). In particular, they attempt to maximise the distance between each hyperplane that separate the classes, as this has been shown to reduce generalisation error.

Work by Kotsiantis *et al.* (2007) indicates that in general, support vector machines produce the most effective classification models, at the cost of learning time. On the other hand, statistical approaches train models more quickly, but the resultant models may be less accurate (Kotsiantis *et al.*, 2007). In this work, we build models comprising of hundreds of individual features, across thousands of instances. For this reason, in this thesis we primarily use a faster statistical approach, namely linear logistic regression trees (Landwehr *et al.*, 2003) for classification, although we compare to other classifiers where possible. In particular, we use machine learning to build a real-time news query classifier in Chapter 7.

2.5.2 Learning to Rank

Learning to rank (LTR) approaches use machine learning to tackle document ranking problems. In an information retrieval setting, this typically involves ranking with respect to relevancy, although other ranking criteria are possible. The aim of learning to rank approaches is to improve a given document ranking with respect to some property. This is achieved by re-ranking an initial ranking such that documents with the desired property are promoted into the top ranks.

In their simplest form, learning to rank techniques use initial document rankings for a set of query topics, features about the individual documents within those rankings, and relevance assessments about the individual documents for each query topic, to form a ranking model. This model can then be applied to unseen document rankings, re-ranking them to increase relevancy (or some other desired ranking property). In particular, when building (or training) a model, an initial document ranking is created, referred to as the *sample*. A sample should have high recall in terms of documents with the desired ranking property, e.g. for relevancy-based rankings, the sample should contain many relevant documents (Macdonald *et al.*, 2012). However, these documents do not need to appear within the top ranks; indeed it is the aim of LTR to achieve this through re-ranking. Next, features about each of the documents are extracted. An effective feature should aid in distinguishing the documents that have the desired property, e.g. relevance to the query. In effect, the LTR approach aims to find a combination of these features that leads to effective ranking. Indeed, given the sample and its features, a learning to rank approach will try different combinations of those features to find those that lead to increased effectiveness when ranking the sample. LTR approaches repeat this process for many document samples to find the feature combination that leads to improved effectiveness across all of those samples. This feature combination is referred to as the ranking model. The idea is that the resultant ranking model will generalise to unseen sample rankings, if the training samples are representative of the types of rankings encountered.

Learning to rank approaches can be categorised into three different types. Each type of approach uses a different strategy to evaluate the sample ranking. These types are point wise, pair wise and list wise. Point wise techniques learn on a per-document basis, i.e. each document is considered independently. Pair wise techniques optimise the number of pairs of documents correctly ranked. List wise techniques optimise an information retrieval evaluation measure, like mean average precision, that considers the entire ranking list at one time (Liu, 2009). Prior work has indicated that list wise techniques learn more effective models (Liu, 2009). As such, we focus on list wise approaches to learning to rank in this thesis. In particular, we use a variety of learning to rank approaches to tackle real-time news search tasks. In Chapter 6 we use Automatic Feature Selection (AFS) (Metzler, 2007) to learn how to rank news stories in real-time. Meanwhile in Chapter 8, AFS and LambdaMART (Wu *et al.*, 2010) are used to learn how to rank blogs and tweets for news-related user queries.

2.5.3 Machine Learning Libraries/Toolkits

In this work, to aid our experimentation, we use freely available machine learning toolkits and libraries where possible. Below we list the machine learning toolkits used and where they can be obtained.

- Weka 3: Weka is a collection of machine learning algorithms for data mining tasks written in Java. It provides algorithms and tools for data pre-processing, classification, regression, clustering, association rules, and visualisation (Hall *et al.*, 2009). Weka is available for download from `http://www.cs.waikato.ac.nz/ml/weka/`.

- RankLib: RankLib is a library of learning to rank algorithms, it supports multiple additive regression trees (MART), RankNet, RankBoost, AdaRank, Coordinate Ascent and LambdaMART learning to rank approaches. RankLib is available for download from `http://people.cs.umass.edu/~vdang/ranklib.html`.

2.6 Topic Detection and Tracking (TDT)

Topic Detection and Tracking (TDT) was a DARPA-sponsored initiative to investigate how current events could be found and followed in news streams (Allan, 2002). Note that in difference to the ‘news streams’ examined later in this work, news streams here refer solely to news articles from traditional newswire sources, not user-generated content. TDT began with a pilot study that ran from the 1st of July 1994 to the 30th of June 1995 (Allan *et al.*, 1998), which was then followed by 8 years of open evaluation at the Text REtrieval Conference (TREC) (see Section 2.4.3), ending in 2004.

TDT is concerned with streams of text from newswire sources. The stated goal is three-fold: to split a stream of news into individual news stories; find those stories that discuss an event that have not yet been detected; and to group stories around a single news topic. Research into TDT was originally motivated by the need to develop a technology that would provide an analyst with up-to-date news and alerts concerning new and interesting events. In contrast, today, current events are of key interest to just about everyone. At the time there where no systems which met this information need, making TDT an important research area (Allan, 2002).

The TDT track at TREC uses some specific terminology to describe news-related concepts. An (event-based) topic is defined as a set of news stories that are strongly related by some real world event. ‘Story’ in this case is used to describe some form of information item about a topic. In practice, within the TDT setting, a story is a news article. Event-based topics are different from other types of topics commonly used - notably subject-based topics that are the used for ad hoc retrieval tasks. In comparison, event-based topics are clearly delimited both in time and space - i.e. an event will have a finite time for which it will be relevant and will likely have a geographical location and participants. Another important point is that event-based topics may fundamentally change after their conception, with information being made available as stories are shown to be inaccurate, etc.

Over the course of the 8-years of TREC evaluation, five TDT tasks were covered (Allan, 2002):

1. Story Segmentation - Detection of changes between topically cohesive sections.
2. First Story Detection - Detect if a story is the first story of a new, unknown topic.
3. Topic Detection - Build clusters of stories that discuss the same topic.
4. Topic Tracking - Keep track of stories similar to a set of example stories.
5. Story Link Detection - Detect whether or not two stories are topically linked.

Although this thesis does not directly tackle these TDT tasks, as we are primarily concerned with *search* of news-related content (from both news and user-generated sources), TDT is still relevant to this work in that it introduces news-related concepts that we build upon later in a search setting. Furthermore, the topic tracking task is particularly relevant to one aspect of the news search problem addressed in this thesis, i.e. the ranking of microblog content (see Chapter 8). As such, below we provide a short background into the topic tracking task of TDT.

2.6.1 Topic Tracking

The aim of Topic Tracking (TT) is to follow a topic over time, collecting related documents as they are published. In particular, TT approaches match multiple stories as they arrive in time stamped order to a specific topic. This can be formalised as: given multiple stories on a single topic, for each story that arrives, determine whether it covers the given topic (Allan, 2002). TT is also similar to the ‘filtering’ IR task which makes a binary decision on whether to retrieve a new incoming document given a stable information need. In this case, the documents are stories and the information need is the starting set of stories (the topic). The standard approach to solving such a task is to perform machine learning on a set of features using a training set - a set of documents that is representative of the corpus being trained for - to discover potentially predictive relationships from which a comparison between documents can be made. Many approaches have been proposed, including use of decision trees (Yang *et al.*, 1999) and the k-nearest neighbour approach (Yang *et al.*, 2000). In latter years, the tracking task was modified to allow for human supervision, i.e. a human observer can tell the system if it is making the right choices. This is known as a supervised learning approach. During TDT 2004, the best approach used an adaptive Rocchio method which uses centroids to define the boundaries between related documents (Yang, 2004). It also incorporated Pseudo-Relevance Feedback (Xu & Croft, 2000), which assumes that the documents already matched to the topic are relevant, allowing them to be learnt from. As noted earlier, this holds similarities to document ranking in a tweet environment, were we can consider the tweet stream as a set of documents to be filtered. We examine approaches to tweet ranking in Chapter 8.

2.7 Ranking Aggregates

Traditional Information Retrieval (IR) search tasks involve the ranking of documents for a user query. However, for some IR tasks the object of ranking is not a document, but rather a collection of related documents. One example of such a task is expert search (Craswell *et al.*, 2004). In expert search, the aim is to find people that are expert in a particular topic. The topic is represented in the form of a traditional user query. However, each person is not represented by a single document, but rather the set of documents that the person has authored. The aim is to score each person via their documents, facilitating the ranking of all people considered. We refer to tasks such as expert search generally as *aggregate ranking tasks* (Macdonald, 2009). For such tasks, we refer to the objects that are to be ranked as candidates.

In general an aggregate ranking task can be formulated as follows. Given a set documents D, and a mapping between each document $d \in D$ and one or more candidate aggregates $c \in C$ (e.g. people), rank each candidate c for the query Q using documents D_c, i.e. the documents comprising c. Hence, the aim is to calculate $score\_cand(Q, D_c)$. In the following sub-section, we describe prior approaches to aggregate ranking.

2.7.1 Aggregate Ranking Approaches

There have been a variety of approaches to tackle aggregate ranking. Craswell *et al.* (2004) proposed one of the first approaches, where each candidate c is represented by a virtual document that is simply the concatenation of each document $d \in D_c$. In this way, an aggregate ranking task can be reduced into a traditional document ranking task. Liu *et al.* (2005) later experimented with virtual document representations to tackle community-based question answering. They investigated the effect that the size of the candidate profile, i.e. the number of documents comprising each candidate $|D_c|$, had on the effectiveness of ranking community answers to a question. Their results indicated that larger profiles were advantageous.

Balog & de Rijke (2006) proposed two language models that can be used to tackle aggregate ranking tasks. In particular, the first model, which they refer to as Model 1, builds a language model for each candidate represented by a probability distribution over all terms within the vocabulary, effectively forming a type of virtual document. Their second approach, referred to as Model 2, does not directly model each candidate as a pseudo-virtual document. Instead, the score for each candidate is calculated as the sum of the relation between each document comprising that candidate and the query. This relation is defined by comparing the language models of the (candidate) documents and the query. Balog *et al.*'s Model 2 approach was later extended by Fang & Zhai (2007), to use IR relevance language models to better estimate the relation between each document and the query. Petkova & Croft (2006, 2007) further extended Fang and Zhai's version of Model 2 to take into account the term frequency of the candidate terms within each document and term proximity between candidate and query terms within those same documents.

Importantly, these '*Model 2*' approaches are all based upon summing the relatedness of documents for each candidate to the query. The larger the candidate profile (the number of documents related to a candidate) the larger the resultant score tends to be. Macdonald & Ounis (2006*a*) hypothesised that a simple sum may not be the most effective way to aggregate relatedness between documents. Instead, they proposed an alternative approach to aggregate ranking, referred to as the Voting Model. In particular, under the Voting model, each of the candidate documents acts as a vote for that candidate. The simplest expression of this is when each related document receives a single equal vote, hence the score for a candidate is equal to the number of candidate documents that it has. Macdonald & Ounis (2006*a*) proposed multiple alternative voting approaches inspired by data-fusion techniques. These approaches score each of the candidate documents using an IR retrieval model. The way in which these resultant scores are combined is defined by the voting technique employed. For example, the CombSUM voting technique follows a similar intuition to the '*Model 2*' approach by Balog *et al.*, in that it sums the scores over all of the candidate documents for the query as follows:

$$score\_cand_{CombSUM}(Q, D_c) = \sum_{d \in R(Q) \cap D_c} score(d, Q) \tag{2.21}$$

where $score(d, Q)$ is the score of the document d in $R(Q)$. The difference in this case, is that it is an IR retrieval model such as BM25 (Robertson *et al.*, 1994) or DPH (Amati., 2003) that can be used to define relatedness between each candidate document and the query, rather than the language models used under Model 2 approaches. More advanced voting techniques such as CombMNZ take into account the number of voting documents for each candidate (Macdonald, 2009):

$$score\_cand_{CombMNZ}(Q, D_c) = |R(Q) \cap D_c| \cdot \sum_{d \in R(Q) \cap D_c} score(d, Q) \tag{2.22}$$

where $|R(Q) \cap D_c|$ is the number of documents that both match the candidate set c and were retrieved for the query Q. Other voting techniques may consider only a subset of the candidate set. For example, CombMAX scores each candidate based upon its highest scoring document:

$$score\_cand_{CombMAX}(Q, D_c) = \max_{d \in R(Q) \cap D_c} score(d, Q) \tag{2.23}$$

More recently, Macdonald & Ounis (2011) proposed a learned variant of their approach that combined different document ranking features as well as aggregate ranking approaches together to form a composite model for ranking aggregates. In particular, under this approach, the scores produced by individual aggregate ranking approaches for a candidate act as features about that candidate. Many candidate features are combined using a learning to rank approach (see Section 2.5) to form an aggregate ranking model based upon a set of pre-defined training instances. The resultant aggregate ranking model can be used to score unseen candidates after the extraction of each feature. Through experimentation, they showed that this learned approach could outperform the individual approaches upon which it was built over multiple datasets.

In summary, aggregate ranking approaches tackle information retrieval problems where the objects (candidates) to be ranked are not represented initially as individual documents, but rather as multiple documents. Aggregate ranking approaches can be divided into two types, those that represent a candidate as a virtual document and those that consider each document individually, but then aggregate each document together to form a final score. One effective aggregate ranking approach is the Voting Model (Macdonald, 2009), that considers each candidate's document to be a vote for the that candidate. In this work, we build upon the Voting Model to tackle one of the news search tasks addressed in this thesis (see Chapter 6).

2.8 Resource Selection/Vertical Search

Resource selection is a specific field of information retrieval that deals with the ranking of content from multiple diverse sources, rather than a single centralised one (Callan, 2000; Craswell, 2000). Resource selection is also known as federated search. Similarly, vertical search is a specialisation of resource selection, where each of the external sources represent different domains of content, e.g. fresh news results, standard Web pages, product results, etc.

The motivation behind resource selection is the following. In an operating environment one might have multiple separate content sources each providing unique content that we might want to retrieve for the user query, returning an aggregate of those results. In a Web search scenario, vertical search over Web pages, news articles, blogs, etc. is a common instance of resource selection. However, some (or all) content sources may be controlled by 3rd parties, in the form of search services that do not expose the underlying statistics of the collection and/or may be rate limited. Hence, the challenge is to select a sub-set of the available sources to search and then combine the results in an effective manner for a given query (Craswell, 2000).

In general, the task of resource selection can be described as follows. Given a set of available document sources S that are searchable, and an incoming query Q, select a subset of sources $S' \subset S$. Next, retrieve the top k documents for the query Q ($R(Q)$) from each source in S' and merge the retrieved results into a final composite ranking that satisfies Q. Hence, resource selection can be seen as two distinct tasks. Firstly, the resource selection technique must chose a subset of the available sources to search. Secondly, once documents have been retrieved from each source, the most relevant of these need to be identified and ranked.

Importantly, there are two environments that determine the types of technique that are suitable for resource selection, namely: cooperative and non-cooperative. In a cooperative environment, all of the sources are considered to be working together to produce a final ranking of results. To this end, it is assumed that underlying statistics and indeed documents from each source are available to the merging application. On the other hand, in a non-cooperative environment, each source is only available as a search service. Both the selection of sources to retrieve from, and the subsequent merging of result lists is easier to achieve in a cooperative environment than in a non-cooperative one, as more information about each source is available (Craswell, 2000). Techniques that work in non-cooperative environments instead simulate the missing information by querying each source to produce document samples representing those sources (Si & Callan, 2003*c*).

Another critical distinction that should be made is that there are two different strategies for selecting the sources to search, dependant upon the final search goal. For federated search tasks, the aim is to find all of the *relevant documents*, hence approaches focus on estimating the probability that each source contains relevant content. In contrast, vertical search tasks tend to assume that relevant content will exist, instead focusing on the types/verticals (e.g. products, flight results, etc.) of content that should be returned to the user based on their query.

In this thesis, we employ resource selection techniques to merge news and user-generated content from different sources together. Notably, due to our experimental setting where we have indices for all of the sources used, we are working in a cooperative environment. In the following three sub-sections, we describe prior work in the field of resource selection. In particular, Section 2.8.1 describes relevance-focused source selection approaches, while Section 2.8.2 details approaches for selecting sources based upon vertical types. In Section 2.8.3, we describe prior techniques for merging result lists from different verticals with varying background statistics.

2.8.1 Source Selection via Expected Relevancy

Federated search tasks aim to select content sources that contain relevant content to the user query. One of the most well-known approaches to select sources from which to retrieve content is the CORI algorithm, proposed by Callan *et al.* (1995). CORI scores different sources by the probability that they contain relevant content using an inference network. In particular, the CORI algorithm is a type of virtual document approach, whereby a source that may be selected is represented as a single large document. Notably, CORI assumes that a virtual document for each source is available, this might be generated from the original document collection underlying each source, or be estimated from a document sample retrieved. CORI scores each source by the document frequency and a (variant of) the inverse document frequency for each term within the query Q. The document frequency comes from the source to be ranked s, while the inverse document frequency is calculated over all sources S. In particular, the score for a source given a term $t \in Q$ can be formulated as follows:

$$score_{CORI}(s,S,t) = b + (1-b) \cdot \frac{df(s,t)}{df(s,t) + 50 + 150 \cdot |c_s|/avg\_terms(S)} \cdot \frac{log\left(\frac{|S|+0.5}{|contain(t,S)|}\right)}{log(C+10)} \tag{2.24}$$

where b is a constant (typically 0.4 (Callan, 2000)), $df(s,t)$ is the document frequency of the term t in s, $|c_s|$ is the number of terms within s, $avg\_terms(S)$ is the average number of terms in all sources S, $|S|$ is the number of sources and $|contain(t,S)|$ is the number of sources that contain the term t. As can be seen, the CORI algorithm above (from left to right) is comprised of three components: a constant; a variation of Robertson's term frequency (Robertson & Walker, 1994), where the term frequency has been replaced with document frequency (*df*) and the constants have been multiplied by a factor of 100 to account for the larger *df* values; and the Turtle's (Turtle & Croft, 1989) inverse document frequency, where the number of documents has been replaced with the number of sources ($|S|$). Approaches that use a similar virtual document approach to CORI include CVV (Yuwono & Lee, 1997), and KL-divergence (Xu & Croft, 1999). CORI has been reported to be effective over multiple datasets (Callan, 2000; Craswell *et al.*, 2000; Powell & French, 2003). However, some work has contested that these claims (D'Souza *et al.*, 2004; Shokouhi, 2007).

Later, Si & Callan (2003c) noted that CORI often failed when the different sources vary greatly in size. To tackle this, they proposed an alternative to CORI, referred to as ReDDE. ReDDE estimates the number of relevant documents to the user query within each source using a sampling method (assuming a non-cooperative environment). In particular, they query each source to build up a sample of documents from that source beforehand. This sample is then indexed. For a user query, ReDDE submits that query to the index of sample documents, scoring the source based upon the documents retrieved. In particular, each source s is scored for a query Q as follows:

$$score_{ReDDE}(s, S, Q) = \sum_{d \in s(sample)} P(rel|d, Q) \cdot \frac{1}{|s(sample)|} * |S| \tag{2.25}$$

where $C_j(sample)$ is a sample set of documents from s and $|s|$ is the size of this set. $P(rel|d, Q)$ is the probability that document d in $s(sample)$ is relevant to Q, and is calculated as the retrieval score of d for Q over the combination of all sample document sets S. Si and Callan show that ReDDE performance is at least as good as CORI when selecting the top 10 resources when selecting from large numbers or resources.

Shokouhi (2007) proposed an alternative approach to ReDDE, where instead of using a probabilistic measure of relevancy, they instead use normalised retrieval scores produced from IR document weighting models. This approach is referred to as Central-Rank-based Collection Selection (CRCS). CRCS is calculated as:

$$score_{CRCS}(s, S, Q) = \frac{|s|}{(\max_{s' \in S} |s'|) \cdot |sample(s)|} * \sum d \in s(sample) R(d, Q) \tag{2.26}$$

where $|s|$ is the size of source s and $|sample(s)|$ is the size of the document sample and $R(d, Q)$ is the score for the document d for query Q using a document weighting model. Through experimentation on a testbed using the TREC GOV2 test collection, Shokouhi (2007) showed that CRCS provided superior performance in most cases to both CORI and ReDDE.

2.8.2 Source Selection via Vertical Type

In contrast to federated search, vertical search tasks choose sources to search based upon their vertical type (Arguello *et al.*, 2009). For instance, for the query 'laptop', a vertical search technique might decide to retrieve results from a product vertical. Related to this type of source selection is the field of query classification into topic categories (Li *et al.*, 2008), i.e. the incoming query is classified based upon its relation to multiple available categories, where each category represents a source (vertical).

For categorisation, query classification approaches typically involve the leverage of evidence from outwith the query text itself, since user queries are often short and under-specified. Beitzel *et al.* (2005, 2007) proposed one such approach referred to as selectional preference. This approach uses machine learning to discover textual relations from a large (unlabelled) query-log, such that queries like ‘laptop’ can be associated to category descriptors such as ‘electronic’ or ‘product’. In contrast, Shen *et al.* (2005) as part of the 2005 KDD CUP (Li *et al.*, 2005), used documents from each source to determine whether a query belonged to each category, in a similar manner to the ReDDE federated search approach by Si & Callan (2003*c*). Li *et al.* (2008) used machine learning in conjunction with lexical features from each query, in addition to category labels inferred from a query-log query-click graph, to classify unseen queries. Similarly, Diaz (2009) also used machine learning to classify queries, although for the news vertical only. In particular, Diaz extracted features from a collection of news articles and from web/vertical query-logs. However, Diaz also introduced click-feedback, i.e. allowing a user’s subsequent clicks to enhance the model over time. Arguello *et al.* (2009) later expanded upon the approach by Diaz for multiple verticals, extracting features from each vertical considered to build a classification model.

In this thesis, we focus on the news vertical only. Indeed, we perform an initial classification of the user query to determine whether it is news-related. This classification is investigated in Chapter 7. However, unlike in a traditional search setting where we would search only newswire sources, we instead propose to search multiple news and user-generated content sources for relevant content. Hence, in Chapter 9, we also use source selection via expected relevancy to chose from our different sources which ones to display content from.

2.8.3 Merging of Document Rankings

The task of merging document rankings can be summarised as follows. Given multiple document rankings for a query $S'_{R(Q)}$, where each individual ranking $s_{R(Q)} \in S'_{R(Q)}$ provides scores for each document, merge $S'_{R(Q)}$ into a single ranking $M_{R(Q)}$. Approaches to merge the document rankings focus on normalisation of the scores for the documents contained within each ranking, such that document scores across rankings become comparable even though they are generated by collections with different underlying statistics.

An effective approach for result merging in a cooperative environment was that deployed by the INQUERY system (Callan, 2000). This approach normalised each document by the (estimated) maximum and minimum scores that could be assigned to any source and the (estimated) maximum and minimum scores that any document could receive for the query Q, as follows:

$$score(d,s,S') = \frac{score(d,s) - d_{min_s}}{d_{max_s} - d_{min_s}} + 0.4 \cdot \frac{score(d,s) - d_{min_s}}{d_{max_s} - d_{min_s}} \cdot \frac{score(s) - s_{min}}{s_{max} - s_{min}} \tag{2.27}$$

where $score(d,s)$ is the score for document d retrieved from source s, d_{min_s} is the minimum score that any document could receive for query Q, while d_{max_s} is the maximum score that any document could receive. $score(s)$ is the score for the source s, using the CORI source selection approach described in Section 2.8.1. s_{min} is the minimum score that could be assigned to any source, while s_{max} is the maximum score that could be assigned to each source.

Approaches for merging ranked lists in non-cooperative environments have also been proposed (He *et al.*, 2011; Si & Callan, 2003*b*). In particular, Si & Callan (2003*b*) proposed to employ semi-supervised learning (SSL) to re-score documents within each ranked list. Firstly, each source is sampled to create document sets representing those sources. Next, a central document set is created by combining documents from all of the sample sets. SSL leverages documents that occur in both ranked lists retrieved from each source with documents within the central document set, using the ranks at which matching documents appear to build a regression function. This can then be used to convert the scores for non-matching documents retrieved from any of the sampled sources into a normalised central scores that are comparable and can be used for ranking. Shokouhi & Zobel (2009) later proposed the Sample-Agglomerate Fitting Estimate (SAFE) approach for merging result lists in non-cooperative environments. Like Si and Callan's approach, SAFE uses SSL to perform regression upon documents sampled from each source. However, instead of relying upon document overlap between the central document set and the retrieved documents, it instead estimates the rank of each document for a query using the uniform sampling assumption. All of the estimated ranks for documents retrieved for each query are then used for regression rather than only those that match the central document set.

2.9 Conclusions

In this chapter we have provided a summary of the key concepts within information retrieval (IR) that this thesis builds upon, in addition prior works in IR that are relevant to our later discussions or experiments. In particular, in Section 2.1 we introduced the field of information retrieval, while Section 2.2 detailed the process of indexing that is used to build the searchable structures that enable document retrieval in an IR setting, including more advanced topics such as large-scale indexing with MapReduce. Section 2.3 described the document retrieval process, where by an index is used to retrieve documents for a user query, in addition to techniques that we use in later experiments to increase search effectiveness. In Section 2.4, we discussed evaluation in an IR setting, including measures for determining the effectiveness of an IR system and the IR evaluation methodology that we use in later chapters. Section 2.5 introduced the concept of machine learning for IR and how both classification and ranking tasks that we tackle later can be achieved using machine learning. In Section 2.6, we provided an overview of prior work in the field of topic detection and tracking, that is related to the news search tasks addressed in this thesis. Section 2.7 describes approaches to tackle aggregate ranking tasks within IR, that we build upon in a subsequent experimental chapter. Finally, in Section 2.8 we cover prior works in the field of resource selection, that we use to combine content from many news and user-generated sources in our final experimental chapter. In the next chapter we discuss how user-generated content streams have used for news search tasks and identify the knowledge gap that this thesis addresses.

Chapter 3

Search and User Generated Content

3.1 Introduction

In the previous section we provided a background review of works in the field of Information Retrieval (IR) that are relevant to this thesis. However, IR research efforts that have traditionally focused upon Web search are increasingly shifting into search within user-generated content sources, as the availability and volume of such data increases. In this chapter, we provide an overview of recent works that use user-generated content sources for search and other applications that are relevant to this thesis.

In Chapter 1, we defined user-generated content as documents published on the Web by the general public, rather than by paid individuals, corporations or companies. We group user-generated content of similar types or from the same providers into *sources/streams*. For instance, the blogosphere, refers to the collection of all blogs posted by individuals. Using our terminology, the blogosphere is a user-generated content *source*. In practice, this can be seen as a time-ordered *stream* of blog posts. Within the context of this thesis, where we focus on a real-time search setting, the terms source and stream are interchangeable.

There are a wide variety of different user-generated content sources. We divide these sources into two distinct types, namely: explicit and implicit. Explicit user-generated content refer to documents that the user posts upon the Web with the intent for them to be viewed by other users. For instance, a Twitter tweet is an example of explicit user-generated content. Implicit user-generated content on the other hand, refers to data that is generated by users as a by-product of their online activities, possibly without their knowledge. A high profile example of an implicit user-generated content is the logs of all user queries that Web search engines store (Brenes & Gayo-avello, 2009; Craswell *et al.*, 2009).

In this thesis, we examine whether different user-generated content sources can aid in satisfying news-related queries submitted to universal Web search engines in real-time. In pursuit of this goal, during the course of this thesis, we experimentally investigate whether five user-generated content sources can enhance different components of the news search process. These five sources are: the *blogosphere*, *Digg*, *Twitter*, Web search engine *query logs* and *Wikipedia*.

The aims of this chapter are three-fold. Firstly, we introduce the different user-generated content sources that we use later in this thesis. Secondly, we describe prior works from the literature that have used these five user-generated content sources for tasks that are relevant to news vertical search. Thirdly, we expose the knowledge gap that this thesis addresses. We structure the remainder of this chapter around each of the five user-generated content sources that we use, one section per user-generated content source as follows:

- Section 3.2 introduces the blogosphere and how blogs and blog posts have been used for news-related tasks such as finding news-related blog posts and ranking news stories automatically by their importance.
- In Section 3.3, we introduce the social news site Digg and discuss works that have leveraged Digg to tackle news-related tasks.
- Section 3.4 discusses the emergence of the information sharing platform Twitter and describes recent works in event detection and real-time Twitter search.
- In Section 3.5, we introduce Web search engine query logs as a user-generated content source and describe how query logs have been used for news-related tasks such as classifying user-queries as news-related or not.
- Section 3.6 describes the public encyclopedia Wikipedia and the body of literature that uses Wikipedia for event detection and tracking.
- Finally, in Section 3.7, we provide a summary of the findings of this chapter.

3.2 The Blogosphere

The term blog is a contraction of the term weblog. A blog is a website, generally authored by a single individual, referred to as a blogger. Blogs often follow a set structure. In particular, a typical blog is comprised of three main components (Santos *et al.*, 2012), namely: an HTML *homepage*, containing the latest posts in the blog organised in reverse chronological order (possibly with links to other similar blogs); a *syndicated XML feed*, comprised of the recently published posts in a form that can be easily read by client applications known as aggregators; and a set of HTML *blog posts*, each typically covering a single topic of interest to the blogger (Blood, 2002). The blogosphere refers to the collection of all blogs published on the Web.

Although blogs often share the same overall structure, there are no accepted guidelines for formatting, style or language use in blogs. Furthermore, there are no rules regarding how a blog should be updated, leaving such choices to the blogger. For this reason, blogs can be modified at any time, either adding, removing, expanding or otherwise editing blog posts. Blogs are also known to contain multimedia content in addition to text, in the form of images, video or audio.

We structure our discussion of the blogosphere as follows. In Section 3.2.1, we discuss the types of information need that end-users currently use the blogosphere to satisfy. Section 3.2.2 describes prior works that tackle blog post search. In Section 3.2.3, we describe the TREC Blog track top news stories identification task, while in Section 3.2.4, we details prior works that leverage blogs to tackle news-related search tasks.

3.2.1 Blog Search Queries

Blog search engines, such as Google blog search[1] and Technorati[2] exist to facilitate the retrieval of blogs. As for traditional Web search queries, blog search queries can be divided into two distinct types, namely; ad hoc search and filtering queries (Baeza-Yates & Ribeiro-Neto, 1999). Ad hoc queries can be thought of as many one-off searches. In contrast, filtering queries are repeated continuously over time, such that new blogs or blog posts are identified as they are published. Indeed, from an analysis of the full query log of the blog search engine Blogdigger[3], Mishne & de Rijke (2006) reported that filtering queries are popular in the blogosphere. This indicates that end-users leverage the blogosphere to find information about emerging or evolving events. They also found that the majority of queries made to blog search engines are about named entities, such as celebrities. From a sample of 1000 queries, 52% of the adhoc queries, and 78% of the filtering queries were identified as being related to named entities. Furthermore, through an analysis of the 400 most popular queries, they reported that 20% of adhoc and 15% of filtering queries were news-related, suggesting that blog searchers are also interested in the blogosphere's response to breaking news. Indeed, blogs are a popular medium for the reporting and discussion of news-related topics. A poll by Technorati has shown that 30% of their respondents blogged on news related topics (Sussman, 2009). Furthermore, Thelwall (2006) explored how bloggers reacted to the London bombings, showing that bloggers respond quickly to news as it happens. These

[1]`http://google.com/blogsearch`
[2]`http://technorati.com`
[3]`http://blogdigger.com/`

works motivate the use of the blogosphere as a source of news-related evidence and content for the news vertical of a universal Web search engine.

3.2.2 Blog Post Search

Ad hoc search in the blogosphere can be expressed as ‘find me blog posts about x’, where x is a topic of interest. As we discussed in Section 3.2.1, the majority of ad hoc queries in a blog post setting are related to named entities, and often are news-related. Blog post search is typically tackled using traditional document weighting models, such as those described in Section 2.3.

However, blog post search introduces new challenges, since a blog post may contain irrelevant content in the form of advertisements, client-side scripting code, and its HTML template (Ounis, de Rijke, Macdonald, Mishne & Soboroff, 2006). Such irrelevant ‘boilerplate’ content within blogs has been shown to degrade retrieval effectiveness when searching for blog posts (Lee *et al.*, 2008). To tackle this problem, boilerplate removal approaches such as DiffPost (see Section 2.2.6) have both been proposed and reported to be effective (Lee *et al.*, 2008).

Moreover, the blogosphere is known to contain large volumes of spam blogs (Kolari, Java & Finin, 2006). Spam blogs are those that automatically generate their posts, often using content plagiarised from other legitimate sites, with the aim of generating revenue from on-page advertising or to boost the authority of other affiliated sites (Castillo & Davison, 2010). Indeed, Macdonald, Ounis & Soboroff (2009) analysed the rankings produced by the participant systems to the TREC Blog track from 2006 to 2008, reporting that 10% of the blog posts retrieved were in fact from spam blogs. To counteract spam blogs, machine learning approaches to identify them have been proposed. In particular, Kolari, Finin & Joshi (2006) proposed a series of content and link-based features about a blog to determine whether it is in fact a spam blog, including n-gram frequency counts from the post content, outgoing anchor-text and URLs, and named entities. They combine these features using an SVM classifier (see Section 2.5.1). Similarly, Lin *et al.* (2007) also proposed an SVM classification approach to identify spam blogs. However, rather than classifying a blogs at a single point in time, they developed features that captured how blogs change over time. The aim is to detect dynamic advertising elements contained within a blog in comparison to the main blog content that remains static. They report that through the addition of these features to more traditional content and link-based features, spam blog classification accuracy was increased.

In Section 2.3.5, we discussed query expansion techniques that enrich the user query with additional terms, extracted from top documents ranked from the target corpus. Query expansion techniques have been successfully applied in a blog search setting to improve retrieval performance. For example, within the context of the TREC 2008 Blog track, Lee *et al.* (2008) proposed a query expansion approach that separated blog posts into passages, ranked those passages and expanded the query using terms from the top retrieved of those passages. Relatedly, Weerkamp & Rijke (2008) proposed the use of collection enrichment (see Section 2.3.5) upon external Wikipedia and newswire corpora. Their aim was to exploit the connection between news and the blogosphere, in line with the observation that many blog search queries are news-related. These query expansion approaches were shown to be highly effective during the TREC 2007 and 2008 Blog tracks, resulting in top ranked runs (Ernsting *et al.*, 2007; Lee *et al.*, 2008; Weerkamp & Rijke, 2008).

Other works in the field of blog search have examined how to identify high quality blog posts. The idea underpinning quality-based search is that while a document may be relevant to the user query, it may be uninformative or poorly written (Zhu & Gauch, 2000). Hellmann *et al.* (2010) proposed a machine learning approach that incorporates quality metrics to classify blog posts as high quality or not. In particular, they proposed 150 blog post features for use when training a quality classifier for blogs, using a variety of classification algorithms (see Section 2.5.1). On an evaluation dataset comprised of German, English and Russian blogs, they show that high quality blogs can be classified with up to 98% accuracy.

In this thesis, we argue that the integration of user-generated content – including blog posts – into the Web search ranking for news-related queries can better satisfy end-users. If we are to integrate blog posts into the Web search ranking, then effective ranking approaches are needed, such that we can identify relevant posts to integrate. We investigate the ranking of blog posts for news-related user queries in Chapter 8, using techniques such as query expansion and machine learning inspired by these prior works.

3.2.3 Top News Stories Identification

It has been proposed that the blogosphere, given its strong news focus and tendency to report and discuss recent events, would be a prime source of evidence from which to infer the importance of current news stories from a blogger's perspective (Macdonald, Soboroff & Ounis, 2009). To investigate whether this is the case, the *Top Stories Identification task* was devised within the Text REtrieval Conference (TREC), with the purpose of exploring how accurately top news could be identified using the blogosphere (Ounis *et al.*, 2010). The identification of top news stories from one or more news providers can be considered as answering the question '*what are the most important news stories today*'. This can be seen as a ranking task, where a stream of current news stories—either pre-provided by major news providers or incrementally crawled online—are ranked by their newsworthiness for placement on the homepage or category pages of a news website, using evidence from the blogosphere. Intuitively, the importance of a newswire article can be inferred by measuring the volume and intensity of discussion about that newswire article. The more a newswire article is discussed, the more important it is considered to be. In the case of the blogosphere, this can be estimated using the number of blogs or individual blog posts that discuss a news newswire article. Indeed, based upon this initial idea, strategies of varying complexity and effectiveness have been proposed in the literature.

Mejova *et al.* (2009) proposed to count the number of blog posts that directly cite a newswire article to estimate its importance. In this case, direct hyperlinks pointing to a news article were considered to be a citation. However, when tested during TREC 2009, this strategy provided limited effectiveness due to the sparsity of links to each newswire article within the blogosphere.

Other strategies made use of the textual similarity between a newswire article and related blog posts to estimate that article's importance. One example of this type of approach was proposed by Xu *et al.* (2010), who estimated the current newsworthiness of a newswire article by summing the BM25 (Equation 2.4) scores for blog posts published during the prior 24 hours to that newswire article. In this way, they use a document weighting model to estimate relatedness between a newswire article and recent blog posts. This approach achieved the third best performance during TREC 2010 (Ounis *et al.*, 2010).

Lin *et al.* (2010) proposed a similar approach, whereby a vector-space representation was used describe each newswire article and blog post. These vectors were defined by the TF-IDF (Equation 2.1) scores of each term. They estimated the importance of a news newswire article by summing the similarity scores between that newswire article's vector and each blog post vector, where similarity was defined using the cosine measure. This approach was shown to be effective, achieving the 2nd highest performance at TREC 2010 (Ounis *et al.*, 2010).

Lee *et al.* (2010) proposed a probabilistic language modelling approach to rank newswire articles. Here, the likelihood of each newswire article generating an aggregate language model of recent blog posts, indicates that newswire article's newsworthiness. In particular, they used a clustering algorithm to create multiple topic models from all recent blog posts, representing the main topics of discussion within the blogosphere at the time. Next, a newswire article model was generated from the top retrieved blog posts for that newswire article. Finally, the topic models produced were compared to the newswire article model to determine how related that newswire article is to the current topics of discussion in the blogosphere and hence how important it was. This was the best performing run submitted to TREC 2010.

In this thesis, we postulate that the news search process of a universal Web search engine can be enhanced by identifying the most important events at the time the query was issued using the volume of available user-generated content. In particular, producing a ranking of events for a point in time may be useful when classifying news-related user queries and can be displayed to end-users looking to find out about the top events of the moment. In Chapter 6, we propose approaches to identify important events using a stream of blog posts and evaluate them using the TREC 2009 and 2010 Blog track top stories identification datasets. We also compare to the most effective of story ranking approaches described above as baselines.

3.2.4 News and Blogs

As noted previously in Section 3.2.1, approximately 20% of queries submitted to blog search engines have been reported to be news-related (Mishne & de Rijke, 2006). This indicates that many bloggers blog about news-related topics. A few works have attempted to leverage news-related content from the blogosphere to drive news-related tasks other than search. In particular, König *et al.* (2009) investigated how information from recent blog posts be used to augment news articles when attempting to predict the clickthrough-rate for news-related Web search results. They proposed three groups of features, namely: corpus frequency features, describing how often the query terms are used within the news and blog post corpora; context features, describing whether the query terms appear in similar documents or not; and query-only features, describing characteristics of the query itself such as the presence of stopwords or URL's. They combined these features using the Multiple Additive Regression-Trees (MART) machine learning algorithm (Wu *et al.*, 2008) trained on clickthrough data from the Microsoft Live search engine (now Microsoft Bing). König *et al.* (2009) report that the addition of blog features increased the accuracy of clickthrough rate prediction. Clickthrough prediction for news-related Web search results holds similarities to the identification of important newswire articles of the moment, that we investigate in Chapter 6.

Hassan Sayyadi (2009) proposed the use of blogs in conjunction to other types of user-generated content for event detection and tracking. Their proposed approach builds a network of keywords based on their co-occurrence in blog posts, while a community detection algorithm based upon keyword density in the network is used to discover events. Their results indicate that this graph-based approach is able to track events over time, highlighting the value that blogs can bring to real-time news-related tasks. Relatedly, Kim *et al.* (2012) performed a study of how news-related events are spread within different user-generated content sources, including blogs. Their results indicate that blog users both cite news media and in turn are cited back in some cases.

Finally, Tsagkias *et al.* (2011) examined the task of finding related user-generated content for a newswire article. In particular, they propose a three stage approach. First, they use language modelling (see Section 2.3.4) to generate multiple representations of the original newswire article. They generate these representations in three manners, namely: from news article itself; using user-generated content that explicitly links to the news article; and using reduced language models containing only a few target terms. Second, they retrieve ranked result lists from different user-generated content sources, including blogs, using the different newswire article representations as the query. Third, they combine the rankings produced using data fusion techniques (Shaw & Fox, 1994) into a single ranking. Tsagkias *et al.* (2011) report that when using the full news article text as a query, their approach retrieved the majority of blog posts that directly linked to that news article, i.e. returned many relevant documents. We investigate the similar task of ranking user-generated content for news-related user queries in Chapter 8. Within the context of a universal Web search engine, we also examine the merging of multiple types of user-generated content into a single ranking in Chapter 9, although the approach used and evaluation methodology employed differ from those approaches described above.

3.2.5 Summary

Overall, the blogosphere has been the subject of substantial research efforts in the fields of search and top news stories identification. However, despite a large proportion of blogs and blog searches being reported as news-related, few works have examined the application of blogs for news-related tasks, and none within the context of a user-generated content enhanced news-vertical. In this thesis, we investigate how the blogosphere can be leveraged to aid in satisfying news-related queries submitted to a universal Web search engine. In particular, we use blog post streams in Chapters 6, 7, 8 and 9.

3.3 Digg

Digg[1] is a social news aggregator that enables posts (representing news stories) to be submitted and then subsequently promoted or demoted by users, controlling their presentation on the Digg site. Promoting a Digg post is referred to as digging, while demoting a post is referred to as burying. A single promotion by a user for a post is referred to as a vote or digg. In August 2010, Digg introduced content submitted by news publishers such as newswire providers. Posts submitted to Digg can be considered to be comprised of six components, namely: a headline, with optional sub-heading; the source of the post, e.g. a link to the newswire provider or blog that supplied the post; an abstract summarising the post; the date the post was first submitted to Digg; profile information about the submitter; and the number of times that post

[1] `http://digg.com/`

had been digged. Prior to the re-implementation of Digg in July 2012[1], Digg provided access to its top stories through a programmatic API. This API has enabled developers to maintain an up-to-date list of Digg posts.

3.3.1 Digg Users

Digg users can post about any topic, linking to their blogs or Web pages. However, newswire providers can also post their newswire articles. This means that Digg can be seen as an aggregate source of both user-generated and newswire content. An initial study by Freund *et al.* (2011) examined the mix of professional news to user-generated content on Digg. Their results suggested that less than 50% of the stories submitted to Digg originated from professional journalists and mainstream media, i.e. non user-generated sources. The majority of stories are supplied by users. Hence, Digg is more of a source of user-generated content than newswire content.

It is also notable that due to the wide range of potential submitters and topics, Digg may have good coverage of events both soon after they break in the form of blogs and later as newswire providers publish about those events. Indeed, we examine this later in Chapter 9 of this thesis through a user-study.

3.3.2 Story Promotion

Digg's attraction is that it aggregates the opinions of thousands of its users to decide which posts to promote to the front page through its digg and bury mechanics, rather than rely on the opinion of a few editors. However, this is contested since it has also been shown that participation on Digg is skewed toward super-users who comprise the vast majority of activity on Digg (Lerman, 2007). None the less, the diggs for each post can be seen as a form of user-generated content, identifying those Digg posts that the community finds interesting.

In terms of the top posts that make the Digg front page, prior works have indicated that the Digg social graph is key to post promotion. Lerman (2006) showed that users tend to like stories submitted by friends and that users tend to like stories their friends read and like. Meanwhile, research by Lerman & Galstyan (2008) indicates that stories that quickly spread outside of the submitting user's neighbourhood tend to become highly popular. Indeed, Digg's social network has been compared to that of Twitter. Lerman & Ghosh (2010) reported that both networks are similar, although sharing communities appear closer and more highly connected in Digg than in Twitter. Szab & Huberman (2008) proposed a Bayesian network to predict the final number of diggs (the popularity) that the story would have received 30 days after its original submission. They proposed three types of features for predicting

[1]`http://digg.com/about`

the final popularity of a story: the day of the week/hour of the day a story was submitted; news category of submission; and the number of diggs for that submission early in its life. They report that the number of diggs a post receives early in its lifetime is indicative of that post's final popularity.

Overall, the post digg and bury mechanics that Digg provides gives an indication of the importance of those posts to the Digg community. Later in Chapter 7, we use Digg posts as a source of evidence to identify news-related queries in real-time.

3.3.3 Summary

In summary, Digg is an aggregator that collects news-related content from both newswire and user-generated content sources. Documents are submitted both by newswire providers and by the general public. Documents on Digg vary in popularity, measured by the number of diggs that each receives. The popularity of a digged article is largely dependant upon the connectedness of the submitting user. Importantly, there has been no work examining Digg within a Web search engine context. For instance, it is unclear whether posts submitted to Digg would be more or less effective than using newswire articles to identify emerging news-related queries, or whether Digg posts are useful results to return to the user for such queries. In this thesis, we investigate whether Digg posts and the number of diggs that they receive can aid in satisfying news-related queries submitted to a universal Web search engine. In particular, we use a stream of Digg posts in Chapters 7 and 9.

3.4 Twitter

One of the most prominent sources of user-generated content that has emerged is microblogs. A microblog, as its name suggests, is a weblog (blog). However, each post within that blog is of limited length. The idea behind microblogging is that by promoting short posts, real-time updates can be made with limited effort, possibly from outwith a typical desktop environment. For example, many microblogging platforms facilitate posting via text messages (SMS).

Twitter[1] is at the time of writing the largest dedicated English microblogging service. Twitter enables anyone to sign-up and publicly post messages not exceeding 140 characters about any topic. It is highly popular, and currently has over 100 million active users generating 340 million tweets per day[2]. Other services that contain microblogging elements include Facebook[3] and Tumblr.[4] Another important

[1]http://www.twitter.com/
[2]http://blog.twitter.com/2012/03/twitter-turns-six.html
[3]http://www.facebook.com/
[4]http://www.tumblr.com

platform is Sina Weibo[1], China's most popular microblogging platform, that is of growing interest to the multi-lingual information retrieval community (Qu *et al.*, 2011).

Importantly, Twitter provides programmatic access to both its search service and to a portion of its content stream through open and freely available APIs.[2] For this reason, much of the research into microblogging has focused on Twitter exclusively. However, the *terms of service* for use of these APIs prohibit the re-distribution of tweet texts. This restriction has until recently slowed the creation of re-usable tweet test collections (Soboroff *et al.*, 2012).

We structure our discussion of Twitter as a user-generated content source as follows. Section 3.4.1 describes the unique characteristics of Twitter tweets. In Section 3.4.2, we discuss the challenges of search within Twitter and prior approaches that tackle it. Section 3.4.3 describes the TREC 2011 Microblog track that examined real-time Twitter search. In Section 3.4.4, we discuss prior works that detect events in real-time using Twitter. Section 3.4.5 summaries the main findings of this section.

3.4.1 Twitter Characteristics

Twitter has some notable characteristics that influence how it is used. In particular, users can follow other users, creating a default time stream or 'wall' containing all of the recent tweets by the people that the user follows in reverse-chronological order. This follower/followee relationship creates a rich social graph of users (Kwak *et al.*, 2010). Within tweets, terms beginning with the character '#' are used to denote topics and concepts, these terms are known as hashtags. Hashtags are used to link together many tweets about the same topic. Indeed, it has been reported that over 15% of tweets contain hashtags (Efron, 2011). Similarly, mentions, i.e., user names prefixed with the '@' symbol are used to indicate replies or direct messages to the user in question. Additionally, Twitter allows a user to retweet another's tweets, i.e., post an exact copy of another user's tweet, normally with a reference to the source user (Boyd *et al.*, 2010). This can be done in two manners. Firstly, as an automatic retweet, that just copies the post in question with Twitter itself logging and displaying the originating user. Alternatively, the user can retweet manually, whereby the user copies the tweet and adds RT:@USER to the front of it, optionally with a comment. Finally, when activated, Twitter automatically stores information regarding the geographical location of the person tweeting.

It has also been noted that popular content posted on Twitter is often related to named entities (Kwak *et al.*, 2010). Prior works have examined how to detect named entities in tweets. For instance, Ritter

[1] `http://weibo.com/`
[2] `http://dev.twitter.com/`

et al. (2011) presented a natural language processing framework for Twitter that can be used to identify named entities. Meanwhile, Chenliang *et al.* (2012) proposed an unsupervised approach that uses Wikipedia and a Web n-gram corpus to partition tweets into phrases, subsequently identifying named entities based upon their common relation to other named entities via phrase graph analysis.

In a similar manner to blog posts, microblogs are known for their expression of opinions and sentiments. In a study comprising 216 manually labelled tweets, Pak & Paroubek (2010) reported that over 84% expressed either a positive or negative sentiment. Meanwhile, in a brand analysis context, Jansen *et al.* (2009) reported that over 20% of tweets expressed an opinion relating to a brand or product. Furthermore, for news events, Twitter has been shown to be a good gauge of public sentiment about those events (Bollen *et al.*, 2009). Indeed, Tumasjan *et al.* (2010) showed that Twitter tweets can be used to predict the outcome of U.S. elections, while Diakopoulos & Shamma (2010) used tweets to characterise public opinions during elections.

3.4.2 Twitter Search

Within microblogs, users query for recent information about a real-world events. Indeed, in a microblogging setting, results returned for such queries are often displayed in reverse chronological order, in difference to typical relevance-based rankings seen in Web search results. Alternatively, users may look for other users to 'follow' that share their interests. For instance, both Twitter and Facebook suggest other users to follow/friend without the need for a query. In comparison to blog search, this type of search is similar to searching for concepts, i.e. I am looking for people that closely resemble myself (a concept).

Teevan *et al.* (2011) identified three types of information that users typically search for in a microblogging setting, namely: timely information; social information; and topical information. Indeed, the temporal aspect of microblogging is particularly important, with users often searching for breaking news, real-time content and popular trends. Furthermore, for Twitter, queries have been shown to be even shorter than those observed in Web search, due to users searching with single term queries, e.g., hashtags or mentions (Teevan *et al.*, 2011).

Traditional adhoc search on tweets has seen some investigation, but remains a challenging problem due to the short length of each tweet. In particular, Duan *et al.* (2010) first examined how tweets can be ranked using machine learned approaches. They applied a learning to rank (Liu, 2009) approach using content relevance features, Twitter-specific features, as well as account authority features for tweet ranking. They show that tweet length and the presence of a URL are important features for finding relevant tweets. Relatedly, prior to the development of the social network and microblogging platform

Google+[1], the Google search engine ranked tweets for display in its Web search results for queries requiring recent content, e.g., breaking news queries (Crum, 2010). The idea behind this was to serve fresh discussions from Twitter within the search results, when other forms of content about the topic, e.g., news articles, may not have yet been published. When searching tweets, it has also been reported that tweets containing multiple hashtags are more likely to be spam (Crum, 2010).

However, it remains an open question whether a traditional relevance ranking of tweets is the best way to satisfy users in a microblog setting (Nagmoti *et al.*, 2010). For example, Twitter provides tweet rankings in reverse-chronological order, encapsulating the temporal nature of microblog search. In this case, there is a trade-off between returning fresh tweets that, while recent, may add little value, or returning older tweets that are more relevant but risk being seen as out-of-date. How this trade-off should be tackled to achieve effective search from a user perspective has yet to be fully explored. Other search tasks such as author/expert search have also been proposed in a microblog setting (Nagmoti *et al.*, 2010), but have not yet seen substantial investigation.

Within the context of a news vertical, Twitter is potentially a valuable source of real-time content for display to the user. However, as we have discussed above, Twitter search can be challenging. Hence, in Chapter 8 we investigate the ranking of tweets for the user query and examine the display of tweets to the end-user in Chapter 9.

3.4.3 TREC 2011 Microblog Track

In 2011, the Text REtrieval Conference (TREC) ran a pilot Microblog track, investigating adhoc tweet search (Ounis *et al.*, 2011)[2]. The aim of this task was to find the most relevant tweets for the user query in a real-time setting, i.e. to retrieve tweets on or before a point in time. However, in contrast to other adhoc search tasks, the results returned were ranked in reverse chronological order. This ranking approach was to mimic search as seen on Twitter itself.

To facilitate this track, the first legally redistributable Twitter test collection, named Tweets2011, was developed through collaboration between TREC and Twitter (McCreadie, Soboroff, Lin, Macdonald, Ounis & McCullough, 2012; Soboroff *et al.*, 2012). Indeed, a unique feature of this test collection is the distribution mechanism, which facilitates the sharing of a standard tweets dataset, while respecting Twitter's terms of service. In particular, the collection is distributed as a set of unique tweet identifiers ('tweet ids'), in conjunction with tools allowing the crawling of the data from the Twitter website. The necessary tools for downloading this test collection are available from the TREC website.[3]

[1]`https://plus.google.com/`
[2]`https://sites.google.com/site/microblogtrack/`
[3]`http://trec.nist.gov/data/tweets/`

For the 2011 adhoc task, participant systems returned tweets for a point in time in reverse chronological order for a query. Systems were evaluated in terms of the number of relevant tweets they returned in the top 30 results. To rank tweets in this manner, a variety of approaches were proposed. For example, Amati *et al.* (2011) proposed a new DFReeKLIM retrieval model from the divergence from randomness framework (see Section 2.3.3) that accounts for the very short nature of tweets. The most effective approaches to the 2011 adhoc task focused purely on relevance (Ounis *et al.*, 2011). In particular, the approach by Metzler & Cai (2011) combined learning to rank with pseudo-relevance feedback to find the 30 most relevant tweets to the user query and returned only those 30 tweets. In this thesis, we use the TREC 2011 Microblog track as a test-bed for evaluating how to effectively rank tweets for the user query in Chapter 8.

3.4.4 Event Detection in Twitter

Another prominent topic within the field of microblogs is that of event detection. In line with the strong news-focus of Twitter, research in this area has focused on detecting significant world events, e.g. earthquakes (Sakaki *et al.*, 2010). A variety of automatic event detection approaches have been proposed. For instance, Weng & Lee (2011), proposed an approach that analyses term frequencies over time. They use wavelet analysis to identify potentially event-related terms and then cluster those terms into events. Long *et al.* (2011) proposed an end-to-end topic detection and tracking approach that uses machine learning and tweet features to find topical words representing events. They then construct a bipartite graph to capture the relationship between pairs of events occurring at adjacent times. By joining pairs of events together over time, they build an event chain to describe an event throughout its lifetime. Yuheng *et al.* (2012) examined event detection with tweets from a classification perspective. In particular, they combine latent dirichlet allocation (Blei *et al.*, 2003) to model topics in Twitter, with a tweet a classifier for identifying event-related tweets. Petrovic *et al.* (2010) proposed a light-weight locality-sensitive hashing-based event detection approach that partitions a stream of tweets into bins based upon textual similarity. Finally, Lee & Sumiya (2010) examined event detection at a local level, to detect festivals in Japan using live tweets, tweet-based crowd detection and moving twitterers.

From the perspective of a news vertical, event detection is an important topic, since the search engine requires knowledge of emerging events to be able to identify news-related queries about them. In this thesis, inspired by these works, we use machine learning to develop a real-time news-query classifier that leverages tweets in Chapter 7.

3.4.5 Summary

In summary, Twitter is a real-time information sharing platform, that receives hundreds of millions of posts each day. Posts on Twitter have some unique characteristics that introduce challenges when using them for IR tasks. Twitter is well known as a source of real-time news content, that has lead to research in the fields of real-time search and event detection. The Microblog track at TREC was proposed in 2011 to investigate real-time ad hoc search, however, research in the field is still in its infancy. Furthermore, no prior work has examined how tweets can be used within the context of a news vertical of a universal Web search engine. For instance, it has not been shown when or where tweets are useful results to return for news-related queries. In this thesis, we investigate whether Twitter can be used to more effectively satisfy news-related queries submitted to Web search engines. We use Twitter streams later in Chapters 6, 7, 8 and 9.

3.5 Query-Logs

Modern universal Web search engines record each query submitted to them by users, normally in a structure referred to as the query log. The query log normally contains the query each user submitted, the time that query was submitted and the IP-address and/or other user identification information (Craswell *et al.*, 2009). A query log is highly useful, since it shows how end-users searched over a period of time. The query log is normally accompanied by a click log that records what document the user clicked on (if any). Due to the large numbers of users that search each day, query logs are rich sources of information. Indeed, the Google Web search engine serves around a billion of user queries each day (Norman, 2011).

Web search engines do not often release query or click logs due to privacy concerns[1]. Below we list six query logs have been released for analysis by researchers, although these may no longer be available:

- **Excite, March 1997**: 51,473 queries from the Excite search engine (Jansen *et al.*, 1998).
- **Fireball, July 1998**: 16 million end-user queries from the German Fireball search engine (Hoelscher, 1998).
- **AltaVista, August-September 1998**: approximately 993 million queries from the AltaVista search engine (Silverstein *et al.*, 1999).
- **FAST, February 2001**: 500,000 queries from the European search engine FAST (Spink, Ozmutlu, Ozmutlu & Jansen, 2002).

[1]`http://www.nytimes.com/2006/08/09/technology/09aol.html?_r=1`

- **AOL, March-May 2006**: 20 million queries from over 650,000 users from the AOL search engine (Brenes & Gayo-avello, 2009).
- **MSN, May 2006**: approximately 15 million queries from the MSN Live Search engine (now Microsoft Bing) (Craswell *et al.*, 2009).

Query logs have been the subject of wide ranging investigations. As such, we structure our discussion of query logs into two main sub-sections, each representing a set of prior works relevant to news-vertical search. In particular, in Section 3.5.1, we describe works that have analysed how users search in general and over time. Meanwhile, Section 3.5.2 discusses works that categorise user queries and examine vertical search. Section 3.5.3 summaries the findings of this section.

3.5.1 Query Log Analysis

Each of the aforementioned Web search engine query logs have seen some examination. For instance, Jansen *et al.* (2000) performed a detailed analysis of the Excite query log, while Silverstein *et al.* (1999) analysed the AltaVista query log. These works expose similar querying behaviour by users. For example, they conclude that users typically issue short queries of only 2-3 terms in length and view only the top ten search results. An analysis of the Excite query log from May 1997 indicated that about 10% to 20% of Web search queries contain query operators, e.g. phrase searches (Jansen *et al.*, 1998). However, from an analysis of 100 queries with operators from the Excite search query log from the 1 May 2001, they later showed that such query operators had little effect on search performance (Eastman & Jansen, 2003).

Prior works have noted that query frequency tends to follow a long tail distribution, where few queries appear often and many queries appear only one or two times. For instance, Jansen & Pooch (2001) reported that 57% of query terms from the Excite log from March 1997 were used only once, while 78% were used less than three times. Meanwhile, Silverstein *et al.* (1999) during their analysis of the AltaVista query log reported that the top 25 queries represent 1.5% of the total query volume.

Queries submitted to Web search engines are time orientated. For example, Beitzel *et al.* (2004) investigated how the Web query traffic varied hourly. They show that less than 1% of the day's total queries appear between 5-6am, while 6.7% appear between 9-10pm. However, they also report that the ratio of distinct to total queries in a given hour is nearly constant throughout the day, i.e. the proportion of head and tail queries remains the same. Diaz & Jones (2004) also used the temporal profiles of Web search queries to improve the prediction of average precision for a query, while Chien & Immorlica (2005) used temporal query profiles to find other similar queries.

Other works have examined the identification of news-related queries that experience a burst of activity (Kleinberg, 2002) in tandem with an event. Vlachos *et al.* (2004) performed one of the first query burst detection studies using an early MSN query log. They built a time series for each query n-gram using Fourier analysis to identify news-related queries that undergo either short term or long terms bursts of activity. Jones & Diaz (2007) proposed a classification for bursty queries into those that exhibit no bursts in activity, those that exhibit one large burst and those that exhibit multiple smaller bursts. They relate these bursty queries to events described by newswire article corpora. Subasic & Castillo (2010) later examined one year of query logs, identifying bursty news-related queries based on increased search volumes and clicks. Relatedly, Kulkarni *et al.* (2011) examined how queries and the documents that users click on for those queries change over time using a propitiatory query log from the Bing search engine. They identified four time-based features that can be leveraged to classify user queries that abruptly change their behaviour over time, namely: the number of bursts; the periodicity of those bursts (for those that exhibit repeating patterns); the shape of the burst; and the overall trend (up, down, flat or up then down). Their results showed that news-related queries tend to exhibit a single large burst, with bursts of querying activity for unexpected events dropping off quickly after the event.

From a news vertical search perspective, the most important aspect of query logs are the bursts of activity that (news-related) queries experience when a related event breaks. However, from these prior works, it is not clear how effective burst detection techniques are when identifying news-related user queries. Indeed, later in Chapter 7, we examine how techniques such as burst detection in query logs can be leveraged to identify news-related queries in real-time.

3.5.2 Query Types and Vertical Selection

One research area that has seen extensive investigation is the types of queries that users submit to Web search engines. Various prior works define query taxonomies for different domains. For instance, early work by Spink *et al.* examined how Web search queries could be categorised into topical categories. They analysed Excite query logs comprised of over a million queries for single days in 1997, 1999, and 2001 (Spink, Jansen, Wolfram & Saracevic, 2002; Spink *et al.*, 2001; Wolfram *et al.*, 2001). Using 2,500 queries from each log, they categorised them into 11 topical categories, including: entertainment, recreation; commerce, travel, employment and economy; and computers and the internet. They found that the mix of search topics have changed over the years. However, little prior work has examined the type of query examined in this thesis, namely *news queries*.

Broder (2002) proposed a general taxonomy of Web search queries that is comprised of the three categories shown below:

- **Navigational**: The intent is to reach a particular website, e.g. queries that contain the URL to the site that the user is intending to reach.
- **Informational**: The intent is to acquire some information assumed to be present on one or more Web pages.
- **Transactional**: The intent is to perform some activity that requires interaction with a Web service, e.g. online shopping.

News queries can be considered to be informational in nature, where the user's intent is to acquire information about a news story. Broder's taxonomy was later extended by other researchers by subdividing the three originals into further sub-categories, or by adding abstraction layers. For instance Rose & Levinson (2004) added Directed, Undirected, Advice, Locate and List sub-categories to the informational category. Meanwhile, Jansen *et al.* (2008) supplied a hierarchical classification of user intents as expressed by Web queries based upon prior works. However, these works do not consider news queries explicitly.

Automatic query categorisation approaches have also been proposed to identify queries of specific types. For example, Pu *et al.* (2002) proposed an automatic query categorisation approach by attributing the categories of top Web search results returned for those queries to them. Broder *et al.* (2007) later extended this approach by using a machine learned classifier to automatically classify top Web pages retrieved for those queries. Query categorisation approaches can be used to drive vertical selection. In particular, by categorising a query, specialist content for that category from a vertical can be integrated into the Web search ranking and displayed to the user. For instance, if a query can be categorised as relating to a recent event, then news vertical content can be integrated for that query.

Vertical selection for the news vertical has seen some investigation in the literature. In particular, Arguello *et al.* (2009) examined what types of evidence are useful when selecting a vertical for the user query. They investigated 19 different verticals, including the news vertical. Their results indicate that using machine learning to combine multiple features describing each vertical's relevance to the query provides the best performance. However, they also show that of the verticals tested, the news vertical was one of the hardest to select queries for. Arguello *et al.* (2010) later investigated how to build a general model that can select verticals without vertical specific training data. This approach uses resource selection techniques, such as those described in Section 2.8, to estimate when a vertical contains relevant content. They report that an effective general model needs to use portable features, i.e. those that work well across multiple verticals. However, their results indicate that the news vertical is difficult to predict, receiving the second lowest overall accuracy using a trained model under the average precision measure. Diaz (2009) examined the news vertical in isolation. He proposed a system to integrate news-related content into Web search results. This approach uses a trained classifier to distinguish news and non-news queries that leverages features describing the past querying levels within both a universal query log and a news vertical query log, in addition to the number of related documents retrieved for each query. Their classifier is dynamically updated with click feedback from search users. Using query and click logs from spring 2007 and winter 2008, they show that using click feedback could provide evidence to adaptively improve the classification of queries as news-related or not. König *et al.* (2009) examined the similar task of clickthrough prediction for news-related queries. They use features from the query in addition to features describing the distribution of the query terms in blog and newswire corpora to estimate the proportion of users that will click on a document. Through an evaluation upon unseen queries, they report that clickthrough for 82.5% of the test queries were correctly estimated within an 'error band' of +/-10% of their true clickthrough rate.

In this thesis, we argue that user-generated content can be used to increase the accuracy news query classifiers that leverage only newswire content. In Chapter 7, inspired by these prior works, we propose a real-time query classification approach that leverages machine learning to identify emerging news-related queries based upon related discussion in news and user-generated content streams.

3.5.3 Summary

In summary, query log research has focused upon the analysis of how users query and the types of queries that users submit. However, considering news-related queries that are the focus of this thesis, few works have examined them in detail. Related works have investigated the temporal aspects of Web queries, which naturally encompass queries about breaking news that experience a burst of usage in line with the event. Similarly, some works have investigated news vertical selection and the classification of news-related queries within that vertical. However, these works either use query logs in isolation or tackle related tasks such as clickthrough prediction. In contrast, in this thesis, we investigate how multiple aligned streams of user-generated content, including query logs, can be used to aid in satisfying news-related queries. In particular, in Chapter 7, we use query logs to aid in identifying news-related queries in real-time.

3.6 Wikipedia

Wikipedia[1] is an online free encyclopedia, where articles can be created or edited by anyone. Almost all articles can be modified my the public, excepting a few cases where editing is restricted to prevent

[1] `http://wikipedia.org/`

disruption or vandalism (normally for contentious topics like politics). Wikipedia was created in 2001. Since then it has become one of the largest reference online websites, with in excess of 470 million unique visitors each month[1]. Wikipedia reports that it has over 77,000 active contributors and 22 million articles spanning 285 languages. The language with the most pages is English with in excess of 4 million articles. Since the general public creates the content present on Wikipedia, we consider it a user-generated content source.

3.6.1 News Events and Wikipedia

It has been reported that Wikipedia articles relating to news events are often updated by journalists and interested members of the public with more rigour than the majority of articles on Wikipedia (Lih, 2004). Moreover, Wikipedia has been proposed as a source of real-time information about emerging events. For instance, Michael Jackson's Wikipedia page was updated a total of 104 times on the day of his death, with a further 641 updates on the following day[2]. Wikipedia provides statistics on how each of its pages are updated and how often they are viewed over time. Prior works have leveraged page views in particular to detect emerging events. For instance, Ciglan & Nørvåg (2010) proposed a service named WikiPop that facilitates automatic detection of events and popular topics based upon page views in Wikipedia. Meanwhile, Ahn *et al.* (2011) proposed an event detection approach that uses clustering of page view bursts for related pages into topically-related groups representing events. Wikipedia page views have also been proposed as supporting evidence to improve event detection accuracy using other sources. In particular, Osborne *et al.* (2012) performed a study using Wikipedia page views to filter out spurious events from a real-time Twitter-based event detection system (Petrovic *et al.*, 2010). They showed that Wikipedia page views could be used to filter out non-events detected in Twitter. They also reported that news in Wikipedia appeared to lag around 2 hours behind Twitter.

In general, due to the news-related edits and page views that Wikipedia receives, it may be a useful source of evidence to aid in satisfying news-related queries. For instance, should we note that a Wikipedia page is receiving a large number of edits, then queries related to that page may be more likely to be news-related. Indeed, we examine whether Wikipedia edits can be used to aid in the identification of news-related queries in Chapter 7.

3.6.2 Summary

Wikipedia is a open collaborative encyclopedia that is consulted by millions of users each day. It provides statistics both regarding edits made to Wikipedia pages and views of those pages over time. Prior

[1]`http://stats.wikimedia.org/reportcard/`

[2]`http://wikipedia.org/wiki/Michael_Jackson?action=history`

works have focused on the analysis of Wikipedia page views to detect and track events. However, little research has examined how Wikipedia can be leveraged in a news setting. In this thesis we examine how Wikipedia pages and their edit information can be used when satisfying news-related queries, both as evidence for identifying these news-related queries and as high quality documents to return to the user. We use a stream of Wikipedia edits in Chapter 7 and full Wikipedia pages in Chapter 9.

3.7 Conclusions

In this chapter, we have introduced the five different user-generated content sources that we use later in this thesis, namely, Blogs, Digg, Twitter, Query logs and Wikipedia. We have described each of these sources and discussed prior works that have used these sources for news-related tasks. Finally, for each source, we have motivated why prior works have not considered the use of that source in a news vertical context as we do in this thesis, or have examined only a portion of the whole process. In particular:

- Section 3.2 introduced the blogosphere and how blogs and blog posts have been used for news-related tasks such as finding news-related blog posts and ranking news stories automatically by their importance. However, despite a large proportion of blogs and blog searches being reported as news-related, few works have examined the application of blogs for news-related tasks, and none within the context of a user-generated content enhanced news-vertical.

- In Section 3.3, we introduced the social news site Digg, and discussed the few works that have leveraged Digg to tackle news-related tasks. However, there has been no work examining whether Digg posts can be leveraged to better satisfy news-related queries in a real-time search setting.

- Section 3.4 discussed the emergence of the information sharing platform Twitter and described recent works in event detection and real-time Twitter search. However, whether tweets can be leveraged to more accurately identify emerging news-related queries or whether such queries can be better satisfied by integrating tweets unto the search results have not been investigated.

- In Section 3.5, we introduced Web search engine query logs as a user-generated content source and described how query logs have been used for news-related tasks such as news vertical selection. However, prior works either use query logs in isolation or tackle related tasks such as clickthrough prediction. In contrast, in this thesis, we investigate how many types of user-generated content, including query logs, can be used to aid in satisfying news-related queries.

- Meanwhile, Section 3.6 described the public encyclopedia Wikipedia and the body of literature that uses Wikipedia for event detection and tracking. In this thesis we examine how Wikipedia pages and their edit information can be used when satisfying news-related queries, both as evidence for identifying these news-related queries and as high quality documents to return to the user.

In the next chapter, we describe our news search framework that describes a universal Web search engine that supports a news vertical and integrates these five user-generated content sources.

Chapter 4

A Framework for Real-time News Search

4.1 Introduction

Recall that in Section 1.2, we identified four key challenges that universal Web search engines face when tackling news-related queries, due to the shift toward real-time news reporting by the general public through social media. In particular, these challenges are: how to identify the most important events of the moment; how to accurately classify incoming queries as news-related or not; how to effectively rank news-related content for the user query; and how to merge content from a variety of newswire and user-generated sources into the Web search ranking. In the Chapter 3, we discussed prior work in the field and showed that these challenges have seen little or no prior investigation in the literature. In this chapter, we present a news search framework that describes the functionalities that a universal Web search engine requires to tackle these challenges, and hence more effectively serve news-related queries in real-time.

The framework that we propose is divided into four main components. Each component tackles one of the challenges that we have previously identified, using user-generated content. For each of these components, we describe what is involved in tackling it with user-generated content and why this is important in a real-time news search environment.

One of the key motivations for leveraging user-generated content to tackle news queries within a universal Web search engine is the *real-time* nature of the information needs underlying those queries. In this case, by real-time, we mean that the results the user is interested in will have been published only recently. For this reason, we first discuss the topic of real-time news queries in a universal Web search context. We then describe a taxonomy of the news-related queries that we consider in this thesis. Next, we describe our proposed news search framework in general, followed by its individual components in more detail. In particular, the remainder of this chapter is structured as follows:

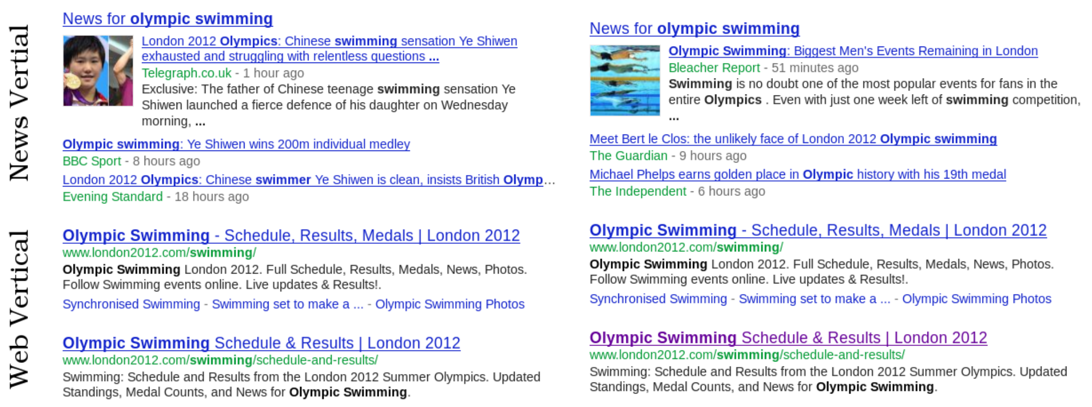
- In Section 4.2, we discuss in more detail the real-time nature of news-related queries and the consequences for universal Web search engines.
- Section 4.3 defines a taxonomy of news-related queries that we consider in this thesis.
- In Section 4.4, we present an overview of our news search framework and its four components, describing the functionality that each provides. In the following four sections, we then discuss the challenges that the fast paced news search environment has raised for each component and how user-generated content might be used to overcome these challenges.
- Section 4.5 describes first component of our news search framework, namely the ranking of news events by their current importance (Top Events Identification).
- In Section 4.6, we discuss the second of our components, namely the real-time classification of queries to determine whether each news-related or not (News Query Classification).
- Section 4.7 details our third component, namely the ranking of news-related content for the user query (Ranking News-Related Content).
- In Section 4.8, we describe our fourth and final component, namely the merging of news content into the Web search results (News-Related Content Integration).
- Section 4.9 summarises the contents of this chapter and provides conclusions.

4.2 Real-Time News Queries

In the context of this thesis, we define *real-time* to mean ‘that which is dependant on recent information published before the current moment’. Many of the functionalities that a universal Web search engine requires to tackle news-related queries are real-time in nature. This is because the events that drive news-related queries tend to both be recent and change rapidly over time.

For illustration, consider the query ‘olympic swimming’. Figure 4.1 shows the top ranked results returned by the Google Web search engine for this query when submitted on the 1st of August 2012 at both 12 noon and 8pm GMT time. From Figure 4.1, we see that at both points in time, the query was considered news-related[1] and as such, Google’s news vertical had inserted ranked newswire articles as

[1]The 2012 London Olympics were taking place at the time.



News for **olympic swimming**

London 2012 **Olympics**: Chinese **swimming** sensation Ye Shiwen exhausted and struggling with relentless questions **...**
Telegraph.co.uk - 1 hour ago
Exclusive: The father of Chinese teenage **swimming** sensation Ye Shiwen launched a fierce defence of his daughter on Wednesday morning, **...**

Olympic swimming: Ye Shiwen wins 200m individual medley
BBC Sport - 8 hours ago

London 2012 **Olympics**: Chinese **swimmer** Ye Shiwen is clean, insists British **Olymp**...
Evening Standard - 18 hours ago

Olympic Swimming - Schedule, Results, Medals | London 2012
www.london2012.com/**swimming**/
Olympic Swimming London 2012. Full Schedule, Results, Medals, News, Photos. Follow Swimming events online. Live updates & Results!.
Synchronised Swimming - Swimming set to make a ... - Olympic Swimming Photos

Olympic Swimming Schedule & Results | London 2012
www.london2012.com/**swimming**/schedule-and-results/
Swimming: Schedule and Results from the London 2012 Summer Olympics. Updated Standings, Medal Counts, and News for **Olympic Swimming**.

(a) 12:00 noon

News for **olympic swimming**

Olympic Swimming: Biggest Men's Events Remaining in London
Bleacher Report - 51 minutes ago
Swimming is no doubt one of the most popular events for fans in the entire **Olympics** . Even with just one week left of **swimming** competition, **...**

Meet Bert le Clos: the unlikely face of London 2012 **Olympic swimming**
The Guardian - 9 hours ago

Michael Phelps earns golden place in **Olympic** history with his 19th medal
The Independent - 6 hours ago

Olympic Swimming - Schedule, Results, Medals | London 2012
www.london2012.com/**swimming**/
Olympic Swimming London 2012. Full Schedule, Results, Medals, News, Photos. Follow Swimming events online. Live updates & Results!.
Synchronised Swimming - Swimming set to make a ... - Olympic Swimming Photos

Olympic Swimming Schedule & Results | London 2012
www.london2012.com/**swimming**/schedule-and-results/
Swimming: Schedule and Results from the London 2012 Summer Olympics. Updated Standings, Medal Counts, and News for **Olympic Swimming**.

(b) 8:00 pm

Figure 4.1: Top ranked results from the Google Web search engine for the query ‘olympic swimming’ on 01/08/12 at two points in time.

the top result. However, the newswire articles selected for these two points in time were completely different, although little time had passed and the query remained the same. In particular, at 12 noon (Figure 4.1 a), the newswire articles returned were about Ye Shiwen’s second gold medal win and the controversy surrounding it. These results are clearly real-time in nature, as the originating event (the race) occurred the night before, with subsequent developments published over the proceeding day, leading to the hour old newswire article that Google selected as the top result. However, eight hours later (Figure 4.1 b), the next round of swimming competitions had taken place, displacing the ‘old’ newswire articles with fresh ones.

As can be seen from the previous example, a universal Web search engine needs to return recent results for news-related queries. It also demonstrates that even for a single news-related query, the correct results to return for that query can change rapidly over time, in light of current events. Contrast this with the Web vertical results displayed in the second and third ranks in Figure 4.1, which remained stable.

4.3 A Taxonomy of News Queries

For simplicity, until this point, we have considered news-related queries to be one unified class. However, it is worth noting that there are multiple types of news-related query. This has implications for the design choices that we make later in this thesis. For example, to effectively classify news-related queries, it is critical to have a clear definition of what a news-related query is. In this section, we develop a taxonomy to describe the different types of news-related query that are submitted to universal Web search engines.

To create our taxonomy, we perform an analysis of queries from a Web search query-log sample, which is comprised of queries from American users of the MSN Search engine (now Microsoft Bing[1]) in May 2006. Indeed, at the time of writing, this query log was the most recent sample of end-user queries available to academia. The full query log sample contains almost 15 million queries (Craswell *et al.*, 2009). Further details regarding the MSN query log sample can be found later in Section 5.5. From this query log sample, we then further sampled 2268 queries using Poisson Sampling (Ozmutlu *et al.*, 2004). It is these 2268 queries that we use to identify the types of news-related query that are submitted to Web search engines, forming our taxonomy.

We first classified each of these 2268 queries as news-related or not. To do so, we used the crowdsourcing marketplace Amazon's Mechanical Turk (MTurk)[2] to have humans workers assess each query. In particular, for each query, a worker was provided with links to three Web search rankings, representing what the top retrieved results for that query would have been at the time it was originally made (according to the query log). Each worker was instructed to use the rankings provided to determine whether each query was news-related or not, and to provide a link to a Web document supporting this decision. Three workers classified each query, where the final label was the majority of the classes assigned. Assessments by workers that did not view the Web search rankings or did not provide a supporting Web document were rejected. Rejected work was re-submitted to be performed by other workers. The development and experimental evaluation of this crowdsourcing approach can be found later in Section 5.5[3]. The 'Crowdsourced Classification' rows of Table 4.1 report the number of queries that were classified as news-related or not. From Table 4.1, we observe that 7.7% (176 of 2268) of the queries were identified as having a news-related intent. This is lower than the 11% reported by Bar-Ilan *et al.* (2009), but is still represents a large proportion of the queries submitted to Web search engines. For instance, 7.7% represents around 77.6 million queries each day to Google (Norman, 2011).

Next, from the 176 news-related queries identified by the MTurk workers, we manually grouped them, forming four classes that comprise our taxonomy. These four classes are:

- **Breaking**: Queries related to events that have broken within the prior 12 hours.
- **Recent**: Queries related to a recent news event that occurred between 12 and 48 hours previously.

[1]`http://bing.com`

[2]`http://mturk.com`

[3]We omit the details on how this crowdsourcing methodology was arrived at here to avoid breaking the flow. However, these details and the quality assurance techniques employed are described later in Section 5.5

	Property	Value
	# Queries	2268
Crowdsourced Classification	# News-Related	176
	# Non-News-Related	2092
Taxonomy	# News-Related Breaking	62
	# News-Related Recent	69
	# News-Related Long-Running	24
	# News-Related Generic	23

Table 4.1: Classifications for the 2268 queries sampled from the MSN query log.

- **Long-Running**: Queries relating to older events that are still of interest, e.g. 'enron trial'[1].
- **Generic**: General news-page finding queries (e.g. 'news' or 'fox news'). Here the user does not specify any particular news event and is instead is searching to find what is happening at that time.

The 'Taxonomy' rows of Table 4.1 report the distribution of queries over these four news-related query classes. From Table 4.1, we see that queries relating to breaking and recent news events are the most common type of news-related query. On the other hand, generic news queries and queries relating to older events are less frequent. The news search framework that we propose supports all four types of news-related query.

4.4 Framework Overview

Figure 4.2 illustrates our proposed news search framework that describes the functionality of a universal Web search engine and supports a news vertical. Recall from Section 4.1 that our framework is comprised of four components, where each tackles one of the news search challenges that we identified previously in Section 1.2. Each component is represented in Figure 4.2 by a rounded corner box with a solid border. The rounded corner box with a dashed border represents the Web vertical that generates the Web search ranking. Solid block arrows denote how a query is processed by each of the components in turn, resulting in the final document ranking. Unfilled block arrows denote the passing of information about currently important events, which are not dependant upon the user query. From Figure 4.2, we observe that when the user submits a query, that query is processed sequentially by three of the four components, namely: News Query Classification, Ranking News-Related Content and News-Related Content Integration. The fourth component — Top Events Identification — supports the News Query Classification and News-Related Content Integration components by providing up-to-date rankings of

[1]The trial of staff from the U.S. energy, commodities, and services company, Enron, took place during the period of our sampled queries and lasted for many weeks.

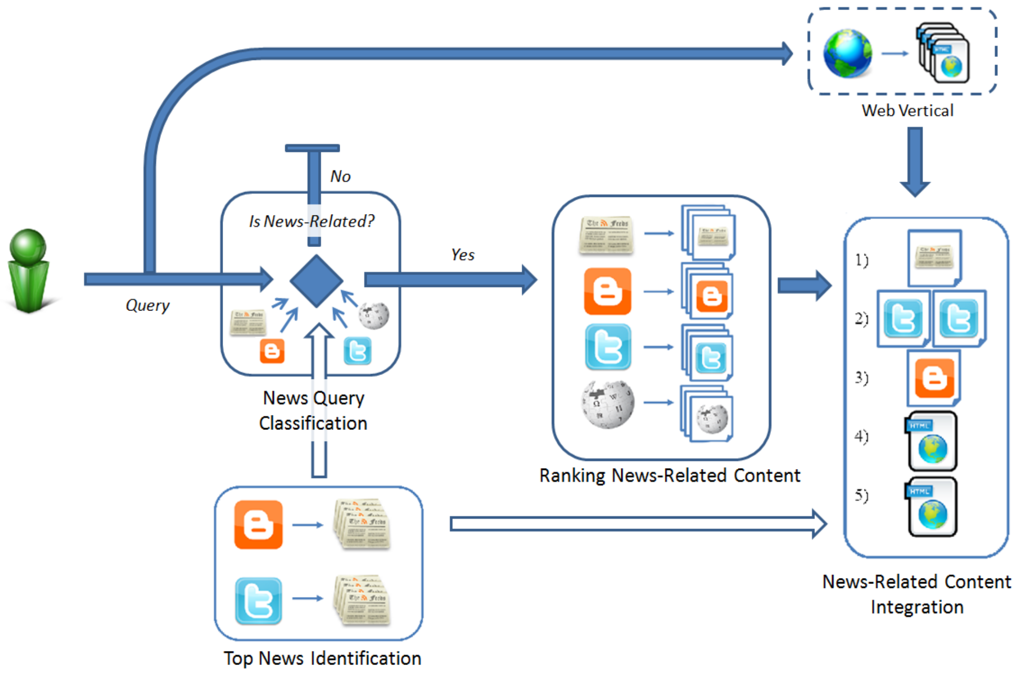

Web Vertical
No
Is News-Related?
Yes
Query
News Query Classification
Ranking News-Related Content
1)
2)
3)
4)
5)
News-Related Content Integration
Top News Identification

Figure 4.2: Overview of the proposed News Search Framework.

important events. We state the purpose of each of the four framework components and formalise their inputs and outputs below:

Top Events Identification: As we have motivated in the Section 1.2, one of the challenges that a news vertical faces in a real-time Web search setting, is how to identify from the set of all recent events those that are actually important. In particular, assume that within our universal Web search engine, the news vertical is continuously crawling newswire sites and feeds to maintain a set of newswire articles A representing recent events. The aim of the Top Events Identification component is to rank these newswire articles A by their current importance at a point in time t. To facilitate this ranking, we assume that the component has access to a real-time stream of documents from which it can estimate the importance of each newswire article, we denote this stream S. Hence, the Top Events Identification component can be considered use a function I that produces a score for each article $a \in A$:

$$I(a, t, S) \tag{4.1}$$

where a is a newswire article representing an event to be scored, t is the current point in time and S is a stream documents to use as evidence. In this thesis, we use newswire articles from the New York Times and Reuters news providers for a. We use streams of blog posts and tweets for S. The output from the News Query Classification component is a list of the newswire articles a ranked by their importance I.

News Query Classification: When a user submits a query Q at time t, the news vertical needs to determine whether that query is news-related or not. If a the user query is news-related, then news-related content will be selected and then merged into the Web search ranking, else the Web search ranking will be returned unaltered. To make this classification, the classifier has access to one or more real-time streams of documents, e.g. current newswire articles or blogs. As before, we denote these streams S. News Query Classification can be considered as a function C that outputs a binary decision for an input query Q at the time that the query arrives t, using evidence from one or more available content streams S:

$$C(S_{1..n}, Q, t) \tag{4.2}$$

where S is a stream of documents, n is the number of available streams S, Q is the user query and t is the time that query was made. We use newswire articles, blog posts, tweets, Wikipedia pages and digg streams for S in this thesis. The output from the News Query Classification component is a binary decision for the query Q, i.e. is it news-related or not.

Ranking News-Related Content: Once a query Q has been identified as news-related at time t, news-related content relevant for that query needs to be selected, such that it can be integrated into the Web search ranking. Within our news search framework, we consider the news-related content selection to be a ranking task, where news-related content from each available newswire and user-generated source is to be ranked for the user query. The Ranking News-Related Content component can be seen as a series of functions R, one for each source S, which ranks documents within S published before time t for the query Q:

$$R(S, Q, t) \tag{4.3}$$

where S is the source from which to rank documents, Q is the user query and t is the time that query was made. The sources S that we use are the same as the five streams used for news query classification above. The outputs of the Ranking News-Related Content component are n rankings R for the query Q, one for each source S.

News-Related Content Integration: After news-related content has been ranked for the user query Q at time t, the last stage of the news search process is to integrate these results into the Web search ranking.

The aim is to increase coverage for the user's news-related query by providing additional content from a variety of sources. In particular, the output from the Ranking News-Related Content component is a series of n rankings R for the user query, one from each of the available sources S. The Top Events Identification component also ranked newswire articles by their importance, producing a ranking I. Dependant upon the user query, some of the top documents from these rankings will be merged into the Web Search ranking, denoted R_W. The News-Related Content Integration component can be seen as a function M that merges the n rankings R for the query Q and the newswire article ranking I into the Web search ranking R_W:

$$M(R_{1..n}, R_W, I, Q, t) \tag{4.4}$$

where R_n is the ranking for the nth source, R_W is the Web search ranking, I is a ranking of newswire articles by their current importance, Q is the query and t is the time that query was made. The output of the News-Related Content Integration component is a single ranking M that may contain content from any of the rankings R or the newswire article ranking I.

Note that each of the components is time dependant (see the parameter t in Equations 4.1 to 4.4), in line with the real-time nature of news-related queries. For completeness, it also worth noting that within our news search framework we do not consider the *Web vertical* in detail, since generating effective Web search rankings is out of the scope of this thesis. Indeed, we assume that we have an oracle that can produce Web result rankings for us.

Having described the purpose of each of the four components of our news search framework, we now discuss in more detail the challenges involved in tackling each. The remainder of this chapter is structured as follows. In particular, Section 4.5 discusses the Top News Identification component, while Section 4.6 details the News Query Classification component. In Section 4.7, we describe the Ranking News-Related Content component. Section 4.8 details the News Content Integration component. We summarise the findings of this chapter and provide conclusions in Section 4.9.

4.5 Top Events Identification

The Top Events Identification component aims to rank in a fully automatic manner events by their current importance/newsworthiness for a point in time. This differs from a typical document ranking task, in that we are not ranking documents for a query. Rather, we have a set of events, each represented by a newswire article a. The goal is to effectively rank these newswire articles, such that those representing important events are ranked higher than those representing lesser events, with respect to the current time

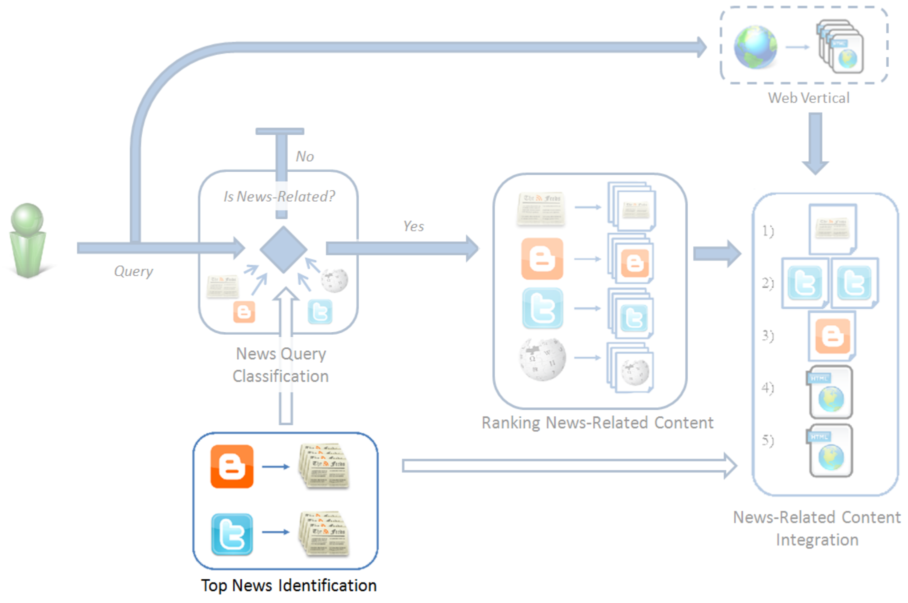
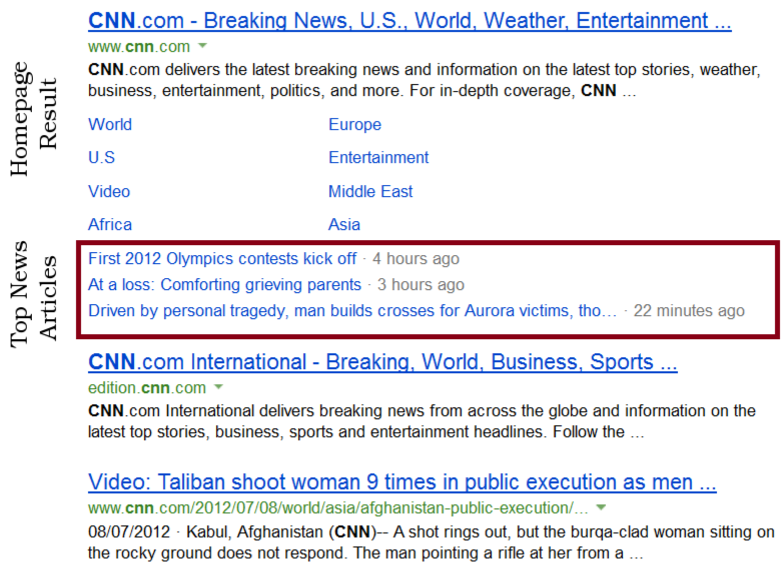

Web Vertical
No
Is News-Related?
Yes
Query
News Query Classification
Ranking News-Related Content
1)
2)
3)
4)
5)
News-Related Content Integration
Top News Identification

Figure 4.3: Top news identification component within our news search framework.

t. Note that in the remainder of this section, we use the term 'event' to describe the wider concept of something happening in the world, and 'newswire article' to describe representation that we use for those events. For the purposes of the Top Events Identification component only, events and newswire articles refer to the same thing.

We identify two ways in which a ranking of currently important events can be used to enhance a universal search engine for news-related queries. First, we might be able to reduce the likelihood of unimportant events leading to the incorrect classification of queries. To illustrate, on 01/08/2012, the Wall Street Journal[1] published a newswire article entitled 'Next's Recession Beating Performance Ups Pressure on M&S'[2] ('M&S' refers to the shopping chain Marks and Spencer). We consider this newswire article to be relatively unimportant, because it is of interest to only a very small community. However, ignoring the low importance of this newswire article might result in queries such as 'M&S' being miss-classified as news-related, when the vast majority of users were just searching for the shopping store[3]. Hence, by considering the importance of each article, we may be able to more accurately classify news-related queries. Second, we have observed that major universal search engines often use newswire article rankings to enrich their Web search results for *generic* news queries (see Section 4.3).

[1]http://online.wsj.com/
[2]http://blogs.wsj.com/source/2012/08/01/nexts-recession-beating-performance-ups-pressure-on-ms
[3]http://www.alexa.com/siteinfo/marksandspencer.com



CNN.com - Breaking News, U.S., World, Weather, Entertainment ...
www.cnn.com
CNN.com delivers the latest breaking news and information on the latest top stories, weather, business, entertainment, politics, and more. For in-depth coverage, CNN ...

World	Europe
U.S	Entertainment
Video	Middle East
Africa	Asia

First 2012 Olympics contests kick off · 4 hours ago
At a loss: Comforting grieving parents · 3 hours ago
Driven by personal tragedy, man builds crosses for Aurora victims, tho... · 22 minutes ago

CNN.com International - Breaking, World, Business, Sports ...
edition.cnn.com
CNN.com International delivers breaking news from across the globe and information on the latest top stories, business, sports and entertainment headlines. Follow the ...

Video: Taliban shoot woman 9 times in public execution as men ...
www.cnn.com/2012/07/08/world/asia/afghanistan-public-execution/...
08/07/2012 · Kabul, Afghanistan (CNN)-- A shot rings out, but the burqa-clad woman sitting on the rocky ground does not respond. The man pointing a rifle at her from a ...

Figure 4.4: Top Bing search results for the query ‘cnn’ on 27/06/2012.

For instance, Figure 4.4 illustrates how the Bing[1] search engine integrates a ranking of newswire articles within its top search result for the news-related query ‘cnn’. Within our news search framework, we propose to use rankings of top newswire articles to enhance the news vertical in both these ways. Indeed, from Figure 4.3, we observe that the unfilled block arrows pass the newswire article rankings from this component to both the News Query Classification and News-Related Content Integration components, where they are used to enhance the classification of user queries and may be integrated into the Web search ranking, respectively.

However, ranking by importance is difficult, since importance is to some degree subjective. For instance, in a newspaper setting, a person is normally employed in an *editorial* role to decide upon the importance of individual newswire articles. In particular, he/she oversees the organisation and selection of newswire articles written by their journalists or obtained from other newswire services, into the final layout most likely to interest their readership. The most important newswire articles will be ranked prominently, while lesser newswire articles will be placed less prominently or not at all. There are common pre-defined criteria that are known to impact the importance of an event (Wavelength Media, 2009), for example:

- **Timing**: important events are usually new, or at least current.
- **Significance**: the number of people affected by an event will have a bearing on its importance.

[1]`http://bing.com`

- **Proximity**: geographically, the nearer that the event is to the readership, the more important its bearing.
- **Prominence**: events happening to celebrities, politicians or other famous people are more likely to be newsworthy.

However, in a Web search environment, hundreds of news providers report on current events in the form of newswire articles (Newspaper Association of America, 2010). Hence, it is not possible to employ editors to select the most important newswire articles for display. Instead, a fast automatic method of estimating the importance of a newswire article in real-time is required.

Galtung & Ruge (1965) showed that, criteria like those listed above, greatly impact the chances of an event being reported. Indeed, despite different readership demographics and interpretations of these criteria, newspaper editors still choose similar newswire articles to grace their front-pages on any given day (Wavelength Media, 2009). This indicates that there are some events that need to be reported on regardless of newspaper orientation, sometimes referred to as hard news, as opposed to newspaper features (Bell, 1991). Indeed, such newswire articles may cover significant *predictable events*, like elections, or prominent *unpredictable events*, such as the death of a celebrity.

Instead of using editorial staff to rank news stories by their importance, we propose to use our newswire and user-generated content sources S to estimate the importance of a newswire article a at a point in time. Indeed, user-generated content sources, e.g. the Blogosphere and Twitter, may be suitable for this top events identification, as they contain large quantities of reporting and discussion about news-related topics (Kwak *et al.*, 2010). The idea is to estimate the importance of a story based upon the level of discussion about it in one or more user-generated content sources. The advantages of leveraging user-generated content are two-fold. Firstly, (some) user-generated content sources, most notably Twitter, provide large volumes of real-time content that may enable newswire articles relating to breaking events to be identified and promoted early in their lifetime, as discussion builds in social media. Secondly, newswire article rankings by newspapers can be both contaminated by those newspapers editorial line (Sear, 2012) and be rendered less relevant due to isolation of the editors from their readership (Bell, 1991). By using user-generated content to directly monitor what users are talking about, we may be able to produce newswire article rankings that more accurately represent the user-base of a Web search engine.

However, challenges remain regarding how we can use user-generated content sources to effectively rank newswire articles by their importance. In particular, time is a critical component when estimating the importance of a newswire article. The newswire article ranking approach needs to determine whether each event — represented by those newswire articles — is being currently discussed by many social media users. However, if only very recent posts are considered, then there is a risk that random bursts of discussion may cause unimportant events to be disproportionately promoted within the ranking. On the other hand, using older posts may make the resultant newswire article ranking lag in identifying emerging events of interest, reducing its usefulness. Moreover, it is unclear which user-generated content sources will be effective for automated newswire article ranking, or whether different sources provide complementary evidence. In Chapter 6 of this thesis, we propose automatic real-time news-article ranking approaches and investigate these challenges.

4.6 News Query Classification

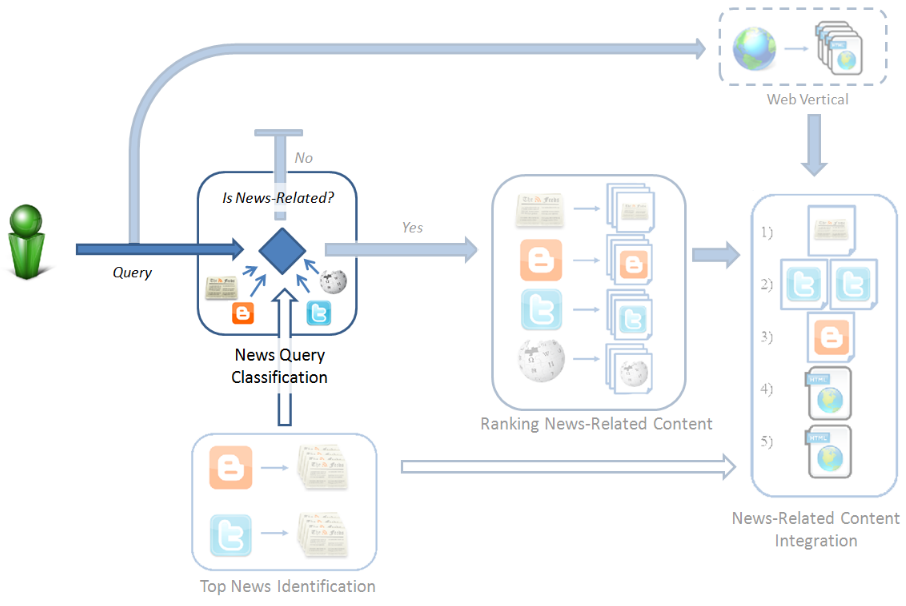
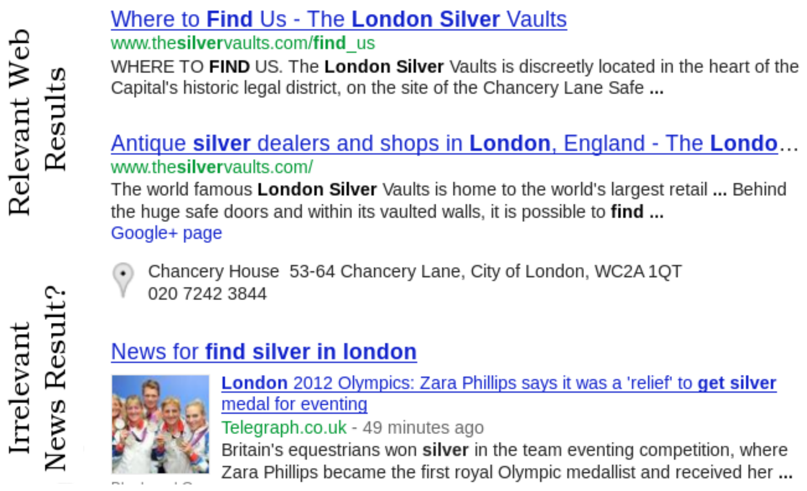

Web Vertical
No
Is News-Related?
Yes
Query
News Query Classification
Ranking News-Related Content
1)
2)
3)
4)
5)
News-Related Content Integration
Top News Identification

Figure 4.5: News query classification component within our news search framework.

The News Query Classification component is responsible for classifying the user query Q as news-related or not at the time it was submitted t. Figure 4.5 illustrates where the News Query Classification component resides within our news search framework. Effective News Query Classification is critical to the news search process as a whole, since inaccurate classification of user queries can result in irrelevant or redundant documents being added to the Web search ranking. For example, Figure 4.6 illustrates a case where the Google search engine returned news-related content within the top results for a query



Where to Find Us - The London Silver Vaults
www.thesilvervaults.com/find_us
WHERE TO FIND US. The London Silver Vaults is discreetly located in the heart of the Capital's historic legal district, on the site of the Chancery Lane Safe ...

Antique silver dealers and shops in London, England - The Londo...
www.thesilvervaults.com/
The world famous London Silver Vaults is home to the world's largest retail ... Behind the huge safe doors and within its vaulted walls, it is possible to find ...
Google+ page

Chancery House 53-64 Chancery Lane, City of London, WC2A 1QT
020 7242 3844

News for find silver in london

London 2012 Olympics: Zara Phillips says it was a 'relief' to get silver medal for eventing
Telegraph.co.uk - 49 minutes ago
Britain's equestrians won silver in the team eventing competition, where Zara Phillips became the first royal Olympic medallist and received her ...

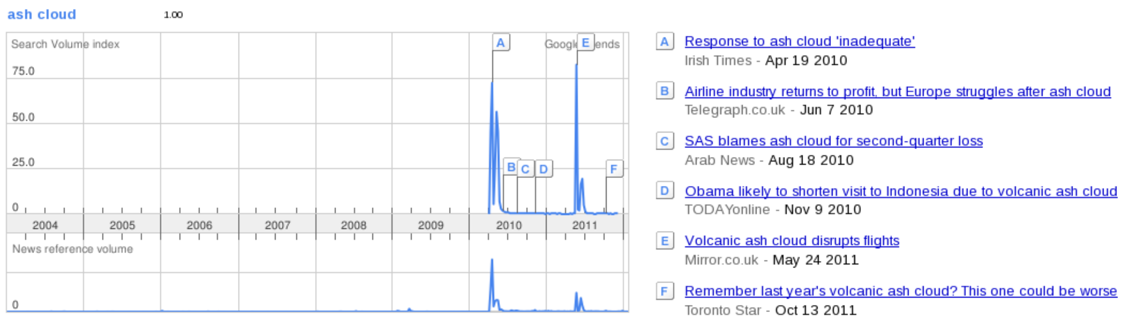
Figure 4.6: Top news-related results returned for the query 'find silver in london' from the Google Web search engine on 01/08/12.

unlikely to be news-related. In particular, a user entering the query 'find silver in london' is likely looking for places that sell silver, rather than searching about the Olympic silver medal results.

Accurately classifying news-related queries in real-time from the larger proportion of non-news queries is a challenging task. In particular, the majority of news-related queries relate to events that have just emerged (see Section 4.3). These types of queries are inherently *unpredictable*, because the events that spark them are similarly unpredictable. For example, Figure 4.7 shows the query usage for the query 'ash cloud' for the Google Web search engine during the period between 2004 and 2011. We observe that the first time that this query was made was during April 2010, with a large peak of query usage, followed by an almost complete absence of usage. During April 2010, we consider 'ash cloud' to be a news-related query about Iceland's Eyjafjallajkull volcano eruption that grounded flights worldwide[1]. From Figure 4.7, we see that the same query observed a large peak almost exactly one year later, in April 2011. Similarly, the query once again became a news-related query, this time relating to the separate Grimsvotn volcanic eruption[2]. The unpredictable nature of these queries make them difficult to tackle with statistical text classification approaches (Nigam *et al.*, 2000), as a query loses its news-related status as interest in the related event dies down. This means that a classification model based upon the occurrence of specific terms will not be robust over time. Furthermore, the identification of news-related queries can be difficult due to the underspecified nature of those queries. Indeed, the average query is 2-3 terms in length (Bar-Ilan *et al.*, 2009). The ambiguity of such short queries makes it

[1]`http://news.bbc.co.uk/1/hi/8623534.stm`

[2]`http://www.mirror.co.uk/news/top-stories/2011/05/24/volcanic-ash-cloud-disrupts-flights-115875-23152491/`



A Response to ash cloud 'inadequate'
Irish Times - Apr 19 2010

B Airline industry returns to profit, but Europe struggles after ash cloud
Telegraph.co.uk - Jun 7 2010

C SAS blames ash cloud for second-quarter loss
Arab News - Aug 18 2010

D Obama likely to shorten visit to Indonesia due to volcanic ash cloud
TODAYonline - Nov 9 2010

E Volcanic ash cloud disrupts flights
Mirror.co.uk - May 24 2011

F Remember last year's volcanic ash cloud? This one could be worse
Toronto Star - Oct 13 2011

Figure 4.7: Google Trends result for the query ‘ash cloud’ (made on 30/01/2012).

all but impossible to be certain as to the user’s intent. For example, consider the query ‘ash’ made during late April 2010. There are multiple possible interpretations of this query, e.g. looking for the Action on Smoking and Health (ASH) homepage, looking for information on the rock band Ash, or it could be about the Eyjafjallajkull volcano eruption, which would count as a news-related query. Hence, we require a classification strategy that can both classify queries relating to breaking events and is accurate when classifying other types of ambiguous news-related query.

To achieve accurate real-time classification of news-related queries, up-to-date knowledge about current events is necessary, e.g. such that we can know that there is a volcanic eruption occurring and hence we can expect news-related queries about it. Newswire articles from large news providers are natural sources of evidence to use, on the grounds that they provide good coverage of current events. Indeed, in the previous section we discussed the Top Events Identification component that can support news-query classification, by providing a ranking of the most important newswire articles of the moment. However, to be able to accurately classify queries relating to breaking news, relying on newswire articles may be insufficient, since when users start to query about an event, no newswire articles may yet have been published (Hansell, 2007).

We propose to use news reporting and discussion in user-generated content to augment evidence extracted from newswire articles for news query classification. In particular, we propose to use user-generated content for news query classification in two main ways. First, to enable more accurate classification of news queries by the leveraging of the chorus effect (Diamond, 2001) from many different content sources. Indeed, if multiple sources indicate that the topic of the query is currently of interest, then we can have more confidence that the query is in fact news-related than when considering a single source alone. Secondly, to increase the speed at which we can classify news-related queries that relate to breaking events.

However, there are challenges in leveraging user-generated content for news query classification. Foremost of these challenges is how to relate postings from different newswire and user-generated content streams into a unified estimate of a query's news-relatedness. Another challenge arises from the temporal aspects of news query classification. In particular, news-related queries tend not to occur in isolation, as many users use the same query to search about an event over time. Hence, an important challenge is develop a classifier that can maintain or even increase its accuracy over time. In Chapter 7, we propose a novel news query classification approach and investigate both these challenges.

4.7 Ranking News-Related Content

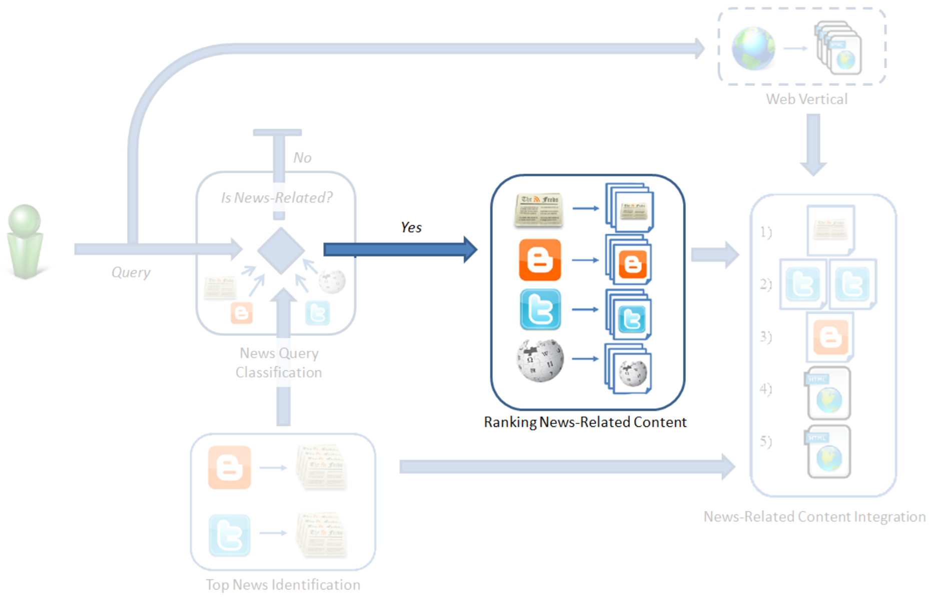

Web Vertical
No
Is News-Related?
Yes
Query
News Query Classification
Ranking News-Related Content
1)
2)
3)
4)
5)
News-Related Content Integration
Top News Identification

Figure 4.8: Ranking News-Related Content component within our news search framework.

The Ranking News-Related Content component aims to find relevant content that can be then integrated into the Web search results for display to the user. In particular, assuming that the universal search engine has access one or more sources of continually updating news content, for each of those sources, the goal is to find relevant content for the user's news-related query. Figure 4.8 illustrates the Ranking News-Related Content component within our news search framework.

In Section 2.3, we described how an information retrieval (IR) system uses document weighting models to rank documents with respect to a query. Document weighting models have been shown to be effective, although they can fail for some queries (Savoy, 2007). In this thesis, we rank five different types of newswire and user-generated content, namely: newswire articles, blogs, tweets, Wikipedia pages and diggs. Prior works have shown that newswire articles can be effectively ranked using document weighting models, like those described in Section 2.3 (Voorhees *et al.*, 2005). Moreover, the content of individual diggs are newswire article titles and snippets, hence it is reasonable to assume that document weighting models will also be effective for ranking them. No works explicitly examine the use of document weighting models for ranking Wikipedia pages for a user query. However pseudo-relevance feedback techniques that are based upon document weighting models have been shown to be effective when using Wikipedia (Xu *et al.*, 2009). Therefore, for these three sources, we use the DPH document weighting model (Equation 2.9) to rank them. However, blogs and tweets have some unique characteristics that are both relevant to their ranking for news-related queries and that are not accounted for by document weighting models. We describe these characteristics in the following two subsections.

4.7.1 Blog Posts

The blogosphere is a prime example of user-generated content. The term *blogosphere* refers to all of the blogs on the Web. The term blog is a contraction of the word 'weblog' that describes the act of someone using the Web to record their thoughts on a particular subject. Importantly, one blog may contain multiple blog posts in chronological order, where each blog post is normally a statement of opinion or viewpoint on a given subject by the blogger.

The large volume of blogs posted each day may make the blogosphere a valuable source of information about current news stories, as bloggers post about the stories that interest them. Indeed, a poll by Technorati has shown that 30% of their respondents blogged on news related topics (Sussman, 2009). Work by Mishne & de Rijke (2006) also showed a strong link between blog searches and recent news - indeed almost 20% of searches for blogs were news-related, indicating that news is popular in the blogosphere. Moreover, Thelwall (2006) explored how bloggers reacted to the London bombings, showing that bloggers respond quickly to news as it happens.

Blog or blog post ranking can be challenging however. In particular, typical document weighting models only consider the relatedness between the query and the document (see Section 2.3), while in a blog setting it might be effective to account for the unique characteristics of blog posts when ranking (see Section 3.2.2). For instance, for news-related queries, end-users might be interested in opinionated blog posts or those that are more authoritative. Hence, to rank blogs or blog posts, we require more advanced ranking approaches that can account for these characteristics.

4.7.2 Real-time Tweet Search

Twitter is a popular microblogging service, which provides an easy means for users to publicly and instantly post messages – known as tweets – not exceeding 140 characters on the Web. The content posted to Twitter varies greatly, from informative news snippets to spam and scams (Kwak *et al.*, 2010). Twitter provides a search service to access relatively recent tweets[1]. In general, tweets retrieved using Twitter search are presented in reverse chronological order (Nagmoti *et al.*, 2010), with the exception of 'promoted' tweets paid for by companies (Rickns, 2010).

Ranking in a microblog setting differs markedly from traditional information retrieval search tasks. In particular, rather than ranking in order of relevance (Manning *et al.*, 2008), microblogs are often returned in reverse chronological order (Nagmoti *et al.*, 2010). The reason behind this different ranking approach is that information needs posed to microblog search engines hold a strong temporal component. Indeed, search in a microblog setting can be considered as answering the question '*find me the most recent information about X*'. Indeed, if we consider Twitter as a source of content to display for news-related user queries, then it is clear that the value Twitter might bring would be with regard to very recent content.

In a universal Web search setting, it might be advantageous to return groups of related tweets or provide live tweet updates as they are posted. Indeed, one of Google's search enhancements was a 'Latest Results' feature that would display scrolling updates in reverse-chronological order for a news-related query, including tweets (Singhal, 2010). Hence, for breaking news stories, it may be useful to display tweets about the story in reverse-chronological order.

However, even when displaying tweets in reverse-chronological order, tweets need to be ranked for the user query. In particular, for any given query, many of the tweets that contain one or more of that query's terms will be irrelevant. Hence, tweets need to be ranked for the query and the most relevant of those selected (Duan *et al.*, 2010). Indeed, Twitter provides similar functionality with its 'top tweets' feature.

Ranking tweets effectively for a query is challenging however, since tweets have some unique structure and characteristics that impact their relevance and quality. Firstly, tweets are by design very short — only 140 characters in length maximum. This short length may make document weighting models such as BM25 (Equation 2.4) less effective, since the term frequency component of such models provides little information as each term is likely to only appear once in a tweet. Furthermore, the shorter tweet length may make vocabulary mismatch between the query and relevant tweets more acute, reducing the recall of the tweet rankings produced with standard document weighting models. Next, the

[1]http://www.twitter.com/search

tweets themselves contain Twitter-specific vocabulary. For instance, when Twitter users post, hashtags, i.e., words beginning with the character '#' are used to denote topics and concepts. These are used to link together many tweets about the same topic. Indeed, it has been reported that over 15% of tweets contain hashtags (Efron, 2011). Similarly, mentions, i.e., user names prefixed with the '@' symbol are used to indicate replies or direct messages to the user in question. Additionally, Twitter allows a user to retweet another's tweets, i.e., post an exact copy of another user's tweet, normally with a reference to the source user (Boyd *et al.*, 2010). Finally, one of the key characteristics of tweets is the inclusion of links to related content. For example, in a news vertical context, a tweet with little textual content might be considered relevant if it links to a useful news article. Hence, to rank tweets, a more advanced approach than using a document weighting model is needed.

In Chapter 8 of this thesis, we investigate how to effectively rank both blog posts and tweets. In particular, we propose learning to rank approaches (see Section 2.5.2) that leverage the unique characteristics of tweets and blog posts. These approaches aim to better capture the aspects of blog posts and tweets that may make them relevant, by representing them as sets of novel real-time stream features.

4.8 News-Related Content Integration

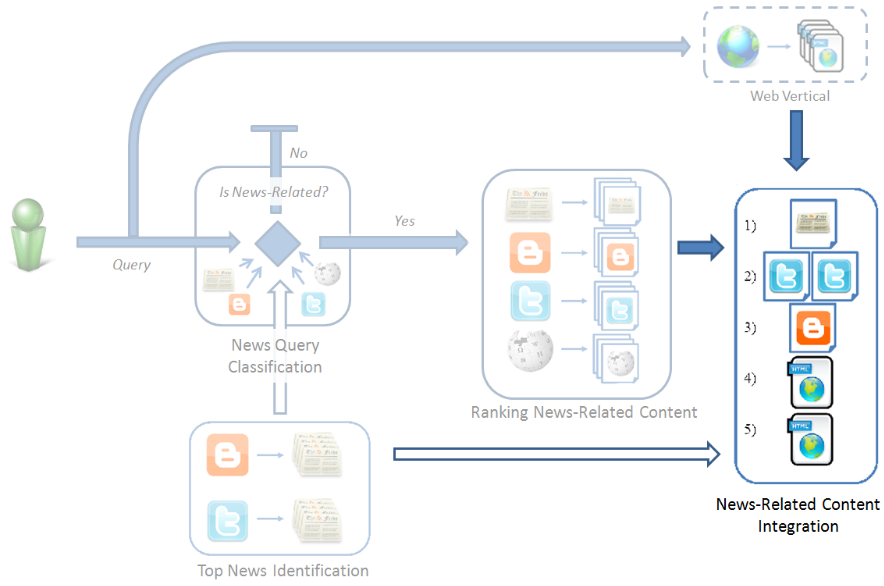

Web Vertical
No
Is News-Related?
Yes
Query
News Query Classification
Ranking News-Related Content
1)
2)
3)
4)
5)
News-Related Content Integration
Top News Identification

Figure 4.9: News-Related Content Integration component within our news search framework.

The News-Related Content Integration component is responsible for combining news-related content into the Web search ranking. Figure 4.9 illustrates the News Content Integration component within our news search framework. From Figure 4.9, we observe that the News-Related Content Integration component has three inputs. First the Web search ranking is provided by the Web vertical. Second, news-related content from newswire and user-generated sources that were ranked for the user query are provided by the Ranking News-Related Content component, i.e. newswire articles, blogs, tweets, Wikipedia pages and diggs in our case. Third, a ranking of the most important events of the moment, represented by newswire articles, is provided by the Top Events Identification component. The News-Related Content Integration component selects some of the top ranked news-related content, possibly in addition to the top new articles that represent currently important events, and merges them into the Web search ranking for the user. The aim is to increase the coverage of the event the user is searching about.

The News-Related Content Integration component is vital to the search process. In particular, the idea behind a vertical within a universal Web search engine is to enhance the Web search ranking with specialist content (Arguello *et al.*, 2009). Meanwhile, the majority of search users never look beyond the first few Web search results (Brin & Page, 1998). Therefore, we require a strategy to select the best of the content that we have ranked for display to the user. Indeed, in our previous 'olympic swimming' example (see Figure 4.1), even though this had a high probably that the user is looking for news-related results, only three newswire articles were integrated into the search ranking within a news results box.

The real-time nature of news reporting motivates the integration of user-generated content in addition to newswire content for some queries. In particular, as news stories are now being reported first in user-generated content sources such as Twitter, for a short period those sources will be the only ones that contain relevant content. Meanwhile, in cases where there are newswire articles that can be returned, returning user-generated content instead of, or in addition to them, may still be advantageous. For instance, by returning a selection of blog posts for a query relating to a U.S. political event, we might be able to provide both democrat and republican viewpoints. Furthermore, for fast-paced events, e.g. sporting matches, user-generated content might allow a live stream or timeline to be added to the results.

It is of note that not all news-related queries are related to a specific event. Recall that in Section 4.3, we identified a class of generic news queries, e.g. 'cnn news'. For this class of queries we might not want to add any news-related content at all. Instead, for these queries, it might be valuable to enhance the Web search results with the ranking of current news events from Top Events Identification component, in a similar manner to that shown previously in Figure 4.4.

In Chapter 9 of this thesis, we examine how to best integrate news-related content into the Web search results. In particular, we perform novel large-scale user-study evaluating whether the integration of user-generated content into the Web search ranking can better satisfy end-users than returning unaltered Web search rankings. We adapt the CORI resource selection/federated search (Craswell, 2000) approach to unify the documents scores for documents ranked from different sources. We also develop a novel framework for comparative ranking evaluation, that simulates the ranking presentation by major Web search engines and facilitates preference assessment across rankings by workers (see Section 9.3). Using workers recruited from the crowdsourcing platform Amazon's Mechanical Turk, we evaluate to determine whether end-users find that the integration of top events, newswire articles, blog posts, Digg posts, Twitter tweets and Wikipedia pages, better satisfy end-users than unaltered Web search rankings for news-related queries.

4.9 Conclusions

In this section, we proposed a new news search framework to describe the search process within a universal Web search engine for news-related queries. This news search framework is comprised of four components. For each component, we have described the functionality of that component, identified the challenges involved and motivated addition of user-generated content. In particular, in Section 4.2, we first discussed the real-time nature of news-related queries and the consequences for universal Web search engines. In Section 4.3, we presented a taxonomy to describe the types of news-related query that we consider in this thesis. Section 4.4 presented an overview of our news search framework and its four components, while Sections 4.5 to 4.8 detailed the challenges involved in each. In the next chapter, we describe the datasets that we use later to evaluate the approaches that we develop for each component of our news search framework.

Chapter 5

Evaluation Datasets and Crowdsourcing

5.1 Introduction

To investigate the proposed news search framework described in Chapter 4, we first need datasets upon which we can evaluate each component. In a search setting, a typical evaluation dataset is comprised of three components, namely: one or more document *corpora* from a particular time-frame that are the subject of the search task; a set of search *topics* that describe the user information need; and a set of *assessments* that define the 'correct' answer to each topic, providing a ground truth to facilitate evaluation. In this thesis, we require one or more evaluation datasets to evaluate each of the four components of our news search framework.

Information retrieval (IR) tasks like ad hoc Web search typically use standard datasets for evaluation, such as those used in IR evaluation workshops and forums like TREC (see Section 2.4.3). These datasets are advantageous as they enable comparison between different approaches to the same task using the same data. Where possible, we use standard evaluation datasets, e.g. the TREC 2011 Microblog track dataset (Ounis *et al.*, 2011), to evaluate the components of our news search framework.

However, the news search process of a universal search engine has not previously been the focus of research by IR evaluation forums and similar venues, while prior work that has investigated similar tasks has predominantly used proprietary or private datasets. In fact, one of the main contributions of this thesis is the development and evaluation of approaches that leverage multiple temporally aligned content streams (corpora) simultaneously, while most existing datasets examine a single stream only. Hence, except when investigating the Top Events Identification and Ranking News-Related Content components — where we use TREC derived datasets — we develop new datasets for evaluation.

Dataset generation is a time consuming and expensive process due to the need to employ human assessors. Crowdsourcing has been championed as a fast and cheap means to generate new datasets (Alonso *et al.*, 2008). Crowdsourcing in general is the act of outsourcing tasks, traditionally performed by a specialist person or group, to a large undefined group of people or community (referred to as the "crowd"), through an open call (Howe, 2010). We extensively use crowdsourcing in this thesis to generate datasets where none are available.

The aims of this chapter are two-fold. First, we describe each of the evaluation datasets that we use in the subsequent four experimental chapters in terms of the topics, corpora and assessments that comprise them. Secondly, for the subset of those datasets that include assessments that we have developed ourselves using crowdsourcing, we provide a short description for each discussing how those assessments were developed, tested and validated.

The remainder of this chapter is structured as follows:

- In Section 5.2, we provide a listing of each dataset that we use in this thesis and detail their statistics.
- Section 5.3 discusses the field of crowdsourcing and the challenges to be overcome when developing datasets using the medium.
- In Section 5.4, we describe the creation of the first of our crowdsourced assessments. These assessments facilitate the evaluation of the Top Events Identification component later in Chapter 6.
- Section 5.5 details the creation of our second set of crowdsourced assessments that facilitate the evaluation of the News Query Classification component of our news search framework, which is investigated in Chapter 7.
- In Section 5.6, we describe our third set of crowdsourced assessments for the evaluation of blog post ranking. We use these assessments later in Chapter 8.
- Section 5.7 details the creation of our final set of crowdsourced assessments that we use in Chapter 9.
- In Section 5.8, we summarise the conclusions of this chapter.

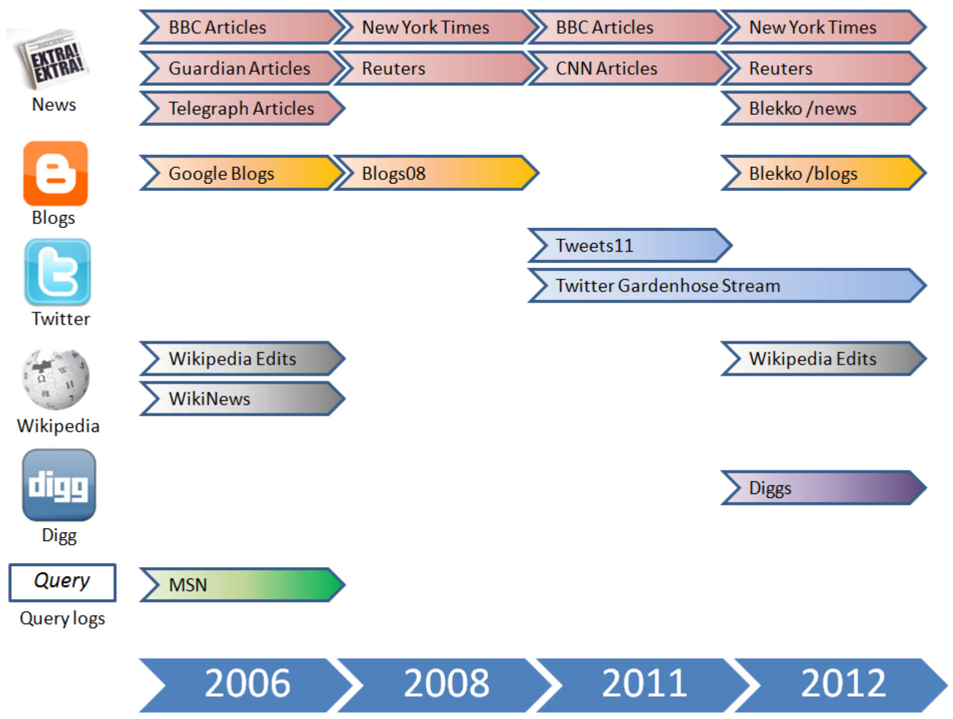


Figure 5.1: Illustration of available document corpora from different years.

5.2 Datasets Overview

In this thesis, we use a total of ten different datasets from time periods between 2006 and 2012 to evaluate whether the use of user-generated content can aid in satisfying news-related queries submitted to universal Web search engines. These datasets are split over the four components of our news search framework (see Section 4.4), which we evaluate in turn in the following experimental chapters to see if each can be enhanced using user-generated content. In particular, we use five datasets during our investigation of the Top Events Identification (TEI) component in Chapter 6; two datasets to evaluate the News Query Classification (NQC) component in Chapter 7; two datasets to examine the Ranking News-Related Content (RNRC) component in Chapter 8; and one dataset during our evaluation of the News-Related Content Integration (NRCI) component in Chapter 9.

As we noted in the previous section, one of the main contributions of this thesis is the development and evaluation of approaches that leverage multiple temporally aligned content streams. As a result, many of the datasets that we develop contain multiple corpora representing different aligned content streams. However, data availability for some sources can be sparse over time. For example, as dis-

Corpus Name	Document Type	Time Range	Number of Documents
$BBC_{May2006}$	News Article	01/05/06 → 31/05/06	13,075
$Guardian_{May2006}$	News Article	01/05/06 → 31/05/06	7,515
$Telegraph_{May2006}$	News Article	01/05/06 → 31/05/06	10,428
$WikiNews_{May2006}$	News Headline	01/05/06 → 31/05/06	400
$Blogs_{May2006}$	Blog Post	01/05/06 → 31/05/06	169,448
$MSN_{May2006}$	End-user query	01/05/06 → 31/05/06	14,921,286
$WikiUpdates_{May2006}$	Page update text and comment	01/05/06 → 31/05/06	28,146,576
$Blogs08$	Blogs/Blog Posts	14/01/08 → 10/02/09	28,488,766
$NYT08$	New York Time Newswire Articles	01/01/08 → 28/02/09	102,853
$TRC2$	Reuters Newswire Articles	01/01/08 → 28/02/09	1,800,370
$Tweets2011$	Twitter Tweets	23/01/11 → 08/02/11	16,141,809
$NYT_{Dec2011}$	New York Times Newswire Articles	17/12/11 → 31/12/11	1,456
$NYT_{Jan2012}$	New York Times Newswire Articles	05/01/12 → 12/01/12	2,310
$Reuters_{Jan2012}$	Reuters Newswire Articles	05/01/12 → 12/01/12	3,102
$Tweets_{Dec2011/Jan2012}$	Twitter Tweets	17/12/11 → 12/01/12	34,600,345
$BlekkoNewsSnippets_{Apr2012}$	Title + Snippet	11/04/12 → 23/04/12	47,085
$BlekkoNewsArticles_{Apr2012}$	News Article	11/04/12 → 23/04/12	47,085
$BlekkoBlogSnippets_{Apr2012}$	Title + Snippet	11/04/12 → 23/04/12	92,547
$BlekkoBlogsFull_{Apr2012}$	Blog Post	11/04/12 → 23/04/12	92,547
$DiggSnippets_{Apr2012}$	Title + Snippet	11/04/12 → 23/04/12	1,116,028
$DiggsFull_{Apr2012}$	Referred Page	11/04/12 → 23/04/12	1,116,028
$WikiUpdates_{Apr2012}$	Page update text and comment	11/04/12 → 23/04/12	2,103,078
$Tweets_{Apr2012}$	Tweet	11/04/12 → 23/04/12	373,121,976

Table 5.1: Statistics of all corpora representing document streams used in this thesis.

cussed in Section 3.5, query logs are rarely made available to the academic community. Moreover, many sources do not provide freely accessible archives, e.g. Twitter. As a result, it can be difficult to align multiple sources together for a time period. Figure 5.1 illustrates the corpora representing different data streams that are available to us for four years between 2006 to 2012 (inclusive). The six document sources that we use in this thesis are listed on the Y-axis, while the year is listed on the X-axis. As we can see from Figure 5.1, there are no time periods for which we can possibly align all six sources. For instance, in 2006, we have aligned newswire articles, blogs, Wikipedia pages/edits and query-logs. However, we lack Twitter and Digg for that same time period, since Twitter had not yet been created in 2006, while Digg was still in its infancy (see Chapter 3). Instead, we use aligned streams where possible and use multiple datasets from different time periods and compare the results between them otherwise. For example, for when examining the News Query Classification component of our news search framework, we use a dataset from 2006 containing newswire articles, blogs, Wikipedia pages/edits and query-logs and a second dataset from 2012 comprised of newswire articles, blogs, Wikipedia pages/edits, Digg posts and Twitter tweets.

For easy reference, Table 5.1 provides the statistics of all of the document corpora (streams) that we use in this thesis, ordered by date. Some corpora may be used in multiple datasets, where those datasets span the same time range. Table 5.2 lists for each component of our news search framework, the datasets that we use for evaluation and the corpora contained within each dataset. From Table 5.2, observe the

Component	Dataset	TREC Dataset?	Crowdsourced Assessments?	Corpora Used
TEI	$BlogTrack_{2009}^{TopNews}$	✔	✖	$NYT08$ $Blogs08$
TEI	$BlogTrack_{2010}^{TopNews-Phase1}$	✔	✔*	$TRC2$ $Blogs08$
TEI	$Twitter_{Dec2011}^{TopNews-NYT}$	✖	✖	$NYT_{Dec2011}$ $Tweets_{Dec2011/Jan2012}$
TEI	$Twitter_{Jan2012}^{TopNews-NYT}$	✖	✖	$NYT_{Jan2012}$ $Tweets_{Dec2011/Jan2012}$
TEI	$Twitter_{Jan2012}^{TopNews-Reuters}$	✖	✖	$Reuters_{Jan2012}$ $Tweets_{Dec2011/Jan2012}$
NQC	$NQC_{May2006}$	✖	✔	$BBC_{May2006}$ $Guardian_{May2006}$ $Telegraph_{May2006}$ $Blogs_{May2006}$ $WikiNews_{May2006}$ $WikiUpdates_{May2006}$ $MSN_{May2006}$
NQC	$NQC_{Apr2012}$	✖	✖	$BlekkoNewsSnippets_{Apr2012}$ $BlekkoNewsArticles_{Apr2012}$ $BlekkoBlogSnippets_{Apr2012}$ $BlekkoBlogsFull_{Apr2012}$ $DiggSnippets_{Apr2012}$ $DiggsFull_{Apr2012}$ $Tweets_{Apr2012}$ $WikiUpdates_{Apr2012}$
RNRC	$BlogTrack_{2010}^{TopNews-Phase2}$	✔	✔*	$Blogs08$
RNRC	$MicroblogTrack_{2011}$	✔	✖	$Tweets2011$
NRCI	$NRCI_{Apr2012}$	✖	✔	$BlekkoNewsSnippets_{Apr2012}$ $BlekkoBlogSnippets_{Apr2012}$ $DiggSnippets_{Apr2012}$ $Tweets_{Apr2012}$ $WikiUpdates_{Apr2012}$

Table 5.2: A listing of the datasets and corpora used to evaluate each component of our news search framework. For each dataset, whether they were produced for the Text REtrieval Conference (TREC) and whether they use crowdsourcing for relevance assessment generation is also noted. Where the crowdsourced assessments column is marked with a ✔*, that indicates that we collaborated with TREC to produce the assessments using crowdsourcing.

distribution of datasets and corpora used for each task. For the TEI component, we use five datasets, where each dataset contains a newswire article corpus that provides events (newswire articles) to rank by importance and a second source of user-generated content with which to drive the importance estimation. The NQC component is evaluated using two datasets, where each dataset contains multiple corpora that represent news and user-generated content streams to use as evidence when classifying queries in real-time. For the RNRC component, we use two TREC datasets, one for blog post ranking and one for tweet ranking. Finally, the NRCI component is evaluated using a single dataset, containing multiple corpora that represent news and user-generated content streams from which content to be integrated is ranked.

Importantly, the new datasets that we develop contain assessments for the task that they are designed to facilitate evaluation of. However, recall from Section 5.1 that generation of assessments can be time consuming and costly due to the need to employ human assessors. Crowdsourcing provides a potentially fast and cheap means to generate assessments for our datasets (Alonso *et al.*, 2008). We use crowdsourcing to generate assessments for four of our datasets, as shown in fourth column of Table 5.2.

Furthermore, recall from Section 5.1 that we use standard TREC (see Section 2.4.3) datasets where available. The third column of Table 5.2 indicates if each dataset was produced by TREC. Traditionally, TREC uses paid assessors to produce the assessments for its datasets. However, the two TREC 2010 Blog track top news stories identification task datasets were an exception, due to insufficient funding being available. Instead, the author produced the assessments for this task on behalf of TREC using crowdsourcing. These datasets are two of the four for which we use crowdsourcing for assessment generation. In Table 5.2, these datasets are denoted by a ✔* in the fourth column. In the following four sub-sections, we detail how each of the ten datasets listed in Table 5.2 were created.

5.2.1 Top Event Identification Datasets

Recall from Section 4.5 that the purpose of the Top Events Identification (TEI) component of our news search framework is to identify the most important events that are occurring at any given moment, where events are represented by newswire articles. We propose to investigate whether the top events of the moment can be identified using real-time discussions in user-generated content sources. To evaluate this, we use datasets that contain newswire articles to represent events, a stream of user-generated content for estimation the importance of each newswire article and assessments identifying those news articles that represent important events.

Table 5.3 summarises the contents of each dataset that we use in this thesis to evaluate the TEI component of our news-search framework, in addition to the time-frame that each covers. As we can see from Table 5.3, we use five datasets. The first and second datasets are TREC datasets designed for

Dataset	TREC	Crowdsourced	# of	Time-Range	Corpora Used	
	Dataset?	Assessments?	Assessments		Newswire Articles	User-Generated Content
$BlogTrack_{2009}^{TopNews}$	✔	✖	10,887	01/01/08 → 28/02/09	$NYT08$	$Blogs08$
$BlogTrack_{2010}^{TopNews-Phase1}$	✔	✔*	8,000	01/01/08 → 28/02/09	$TRC2$	$Blogs08$
$Twitter_{Dec2011}^{TopNews-NYT}$	✖	✖	1,456	17/12/11 → 31/12/11	$NYT_{Dec2011}$	$Tweets_{Dec2011/Jan2012}$
$Twitter_{Jan2012}^{TopNews-NYT}$	✖	✖	2,310	05/01/12 → 12/01/12	$NYT_{Jan2012}$	$Tweets_{Dec2011/Jan2012}$
$Twitter_{Jan2012}^{TopNews-Reuters}$	✖	✖	3,102	05/01/12 → 12/01/12	$Reuters_{Jan2012}$	$Tweets_{Dec2011/Jan2012}$

Table 5.3: Top Events Identification datasets used in this thesis. Datasets that were produced by TREC or that contain crowdsourced relevance assessments are denoted with a ✔ in the associated column. Where crowdsourced assessments are marked with a ✔* the dataset assessments were crowdsourced by ourselves on behalf of TREC.

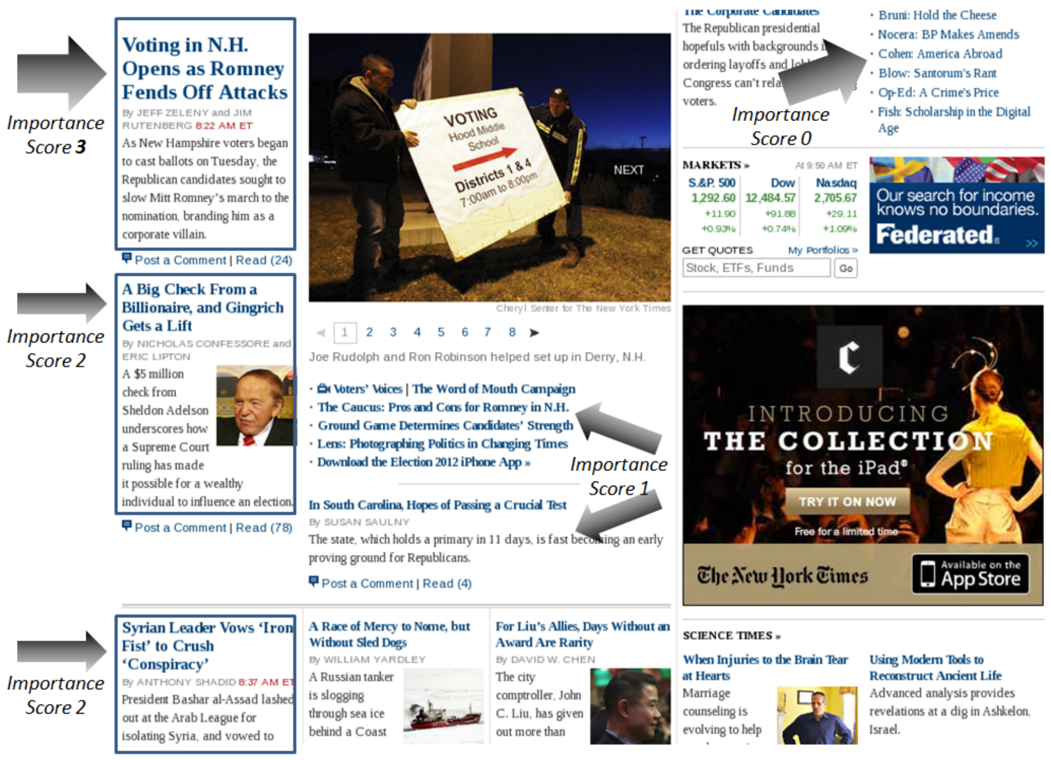
the Blog track 2009 and 2010 top news stories identification task (see Section 3.2.3). We denote these datasets $BlogTrack_{2009}^{TopNews}$ and $BlogTrack_{2010}^{TopNews-Phase1}$, respectively. They contain newswire articles from the New York Times ($NYT08$) and Reuters ($TRC2$) news providers that were published during the period of 2008. The newswire articles published on specific 'topic' days from these news providers were used to represent events to be ranked. The datasets also contain the $Blogs08$ blog post corpus, which systems use as evidence to rank the newswire articles for each topic day. These two datasets comes with pre-provided newswire article importance assessments for each topic day. For a day of interest, newswire articles from that day were assessed as important or not for that day. In particular, the $BlogTrack_{2009}^{TopNews}$ dataset provides 10,887 assessments for 55 topic days (Macdonald, Soboroff & Ounis, 2009), while the $BlogTrack_{2010}^{TopNews-Phase1}$ dataset contains 8,000 assessments for 50 topic days (Ounis *et al.*, 2010). Recall that in the preface to this section, we discussed how for the TREC 2010 Blog track top news stories identification task datasets, we crowdsourced the relevance assessments on the behalf of TREC. $BlogTrack_{2010}^{TopNews-Phase1}$ is one of these two datasets. We describe our methodology for creating these assessments, the validation strategies that we employ and evaluate the quality of the assessments produced later in Section 5.4. This is the first of the four datasets that use crowdsourcing to generate assessments.

Returning to Table 5.3, we see that the remaining three datasets instead contain Twitter corpora for systems to leverage when estimating the importance of events, i.e. the $Tweets_{Dec2011/Jan2012}$ corpus. These datasets once again use newswire articles from the New York Times and Reuters to represent the events to be ranked. Note that these datasets were not developed by TREC, rather we developed these datasets since no other datasets for the task existed. The aim of producing these additional datasets was to facilitate the evaluation of top events identification on tweets. We denote these three datasets as $Twitter_{Dec2011}^{TopNews-NYT}$, $Twitter_{Jan2012}^{TopNews-NYT}$ and $Twitter_{Jan2012}^{TopNews-Reuters}$, respectively. For these datasets, we generated newswire article importance assessments for different points in time based upon the ordering of articles present on the homepages of the news providers whose newswire articles are to be ranked. In effect, this means that our importance assessments are created by the newspaper editor who selected each article for display. In particular, from each homepage, we scrape the set of current newswire articles for our system to rank and also the ground truth ranking of stories against which we will compare. We assign each newswire article present on the homepage a score based upon its placement and prominence. These scores range from '3' to '0', where a score '3' is the most important and a score '0' is the least important. Figure 5.2 illustrates this process for the New York Times homepage on the 10th of January 2012. The primary feature that is used to determine the score of a story is the font size of the headline, the larger the font the higher rank is assigned. Stories that appear further down the page, editorials and special interest stories are also demoted. Also note that there is only ever one score '3' story, which is considered to be the 'top' story of the moment. $Twitter_{Dec2011}^{TopNews-NYT}$ uses 45 homepages downloaded from the New York Times between the 17th and the 31st of December 2011 for assessments. $Twitter_{Jan2012}^{TopNews-NYT}$ uses 55 homepages from the New York Times between the 5th and 12th of January 2012, while $Twitter_{Jan2012}^{TopNews-Reuters}$ uses 63 homepages from Thomson Reuters over the same period.

5.2.2 News Query Classification Datasets

The second component of our news search framework that we investigate in this thesis is the News Query Classification (NQC) component. Notably, no standard news query classification datasets exist, hence we develop two new datasets for NQC evaluation. The datasets that we develop contain end-user queries, including some that are news-related for the component to classify. Each dataset also contains multiple aligned news and user-generated corpora from the same timeframe as its queries, representing real-time streams of content. These streams are used by the component as evidence for classification. Finally, for each of the queries to be classified, each dataset includes assessments identifying those queries that are news-related. These assessments are used as a ground truth using which classification accuracy can be computed.

The aim of our news query classification in Chapter 7 is to determine whether the use of user-generated content streams can lead to increased classification accuracy in comparison to using newswire streams. As a result of this, each dataset includes one or more newswire article corpora, to use as baseline evidence sources when classifying. Furthermore, if possible, the dataset should also contain a query log, again from the same time-frame, such that the state-of-the-art news query classification features proposed by Diaz (2009) can be used.


Importance Score 3
Importance Score 2
Importance Score 0
Importance Score 1
Importance Score 2
VOTING Hood Middle School
Districts 1 & 4 7:00am to 8:00pm
NEXT
INTRODUCING THE COLLECTION for the iPad®
TRY IT ON NOW
Free for a limited time
The New York Times
Available on the App Store
Our search for income knows no boundaries.
Federated.

Voting in N.H. Opens as Romney Fends Off Attacks
By JEFF ZELENY and JIM RUTENBERG 8:22 AM ET
As New Hampshire voters began to cast ballots on Tuesday, the Republican candidates sought to slow Mitt Romney's march to the nomination, branding him as a corporate villain.
Post a Comment | Read (24)

A Big Check From a Billionaire, and Gingrich Gets a Lift
By NICHOLAS CONFESSORE and ERIC LIPTON
A $5 million check from Sheldon Adelson underscores how a Supreme Court ruling has made it possible for a wealthy individual to influence an election.
Post a Comment | Read (78)

Cheryl Senter for The New York Times
1 2 3 4 5 6 7 8
Joe Rudolph and Ron Robinson helped set up in Derry, N.H.

- Voters' Voices | The Word of Mouth Campaign
- The Caucus: Pros and Cons for Romney in N.H.
- Ground Game Determines Candidates' Strength
- Lens: Photographing Politics in Changing Times
- Download the Election 2012 iPhone App »

In South Carolina, Hopes of Passing a Crucial Test
By SUSAN SAULNY
The state, which holds a primary in 11 days, is fast becoming an early proving ground for Republicans.
Post a Comment | Read (4)

The Corporate Candidates
The Republican presidential hopefuls with backgrounds i ordering layoffs and lo Congress can't rela voters.

- Bruni: Hold the Cheese
- Nocera: BP Makes Amends
- Cohen: America Abroad
- Blow: Santorum's Rant
- Op-Ed: A Crime's Price
- Fish: Scholarship in the Digital Age

MARKETS » At 9:50 AM ET

S.&P. 500	Dow	Nasdaq
1,292.60	12,484.57	2,705.67
+11.90	+91.88	+29.11
+0.93%	+0.74%	+1.09%

GET QUOTES My Portfolios »
Stock, ETFs, Funds Go

Syrian Leader Vows 'Iron Fist' to Crush 'Conspiracy'
By ANTHONY SHADID 8:37 AM ET
President Bashar al-Assad lashed out at the Arab League for isolating Syria, and vowed to

A Race of Mercy to Nome, but Without Sled Dogs
By WILLIAM YARDLEY
A Russian tanker is slogging through sea ice behind a Coast

For Liu's Allies, Days Without an Award Are Rarity
By DAVID W. CHEN
The city comptroller, John C. Liu, has given out more than

SCIENCE TIMES »

When Injuries to the Brain Tear at Hearts
Marriage counseling is evolving to help

Using Modern Tools to Reconstruct Ancient Life
Advanced analysis provides revelations at a dig in Ashkelon, Israel.

Figure 5.2: An example of news story importance scoring on the nytimes.com website - 10/01/2012.

Dataset	TREC Dataset?	Crowdsourced Assessments?	# Queries	Time-Range	Corpora Used
$NQC_{May2006}$	✖	✔	1,206	01/05/06 → 31/05/06	$BBC_{May2006}$ $Guardian_{May2006}$ $Telegraph_{May2006}$ $Blogs_{May2006}$ $WikiNews_{May2006}$ $WikiUpdates_{May2006}$ $MSN_{May2006}$
$NQC_{Apr2012}$	✖	✖	2,935	11/04/12 → 23/04/12	$BlekkoNewsSnippets_{Apr2012}$ $BlekkoNewsArticles_{Apr2012}$ $BlekkoBlogSnippets_{Apr2012}$ $BlekkoBlogsFull_{Apr2012}$ $DiggSnippets_{Apr2012}$ $DiggsFull_{Apr2012}$ $Tweets_{Apr2012}$ $WikiUpdates_{Apr2012}$

Table 5.4: News Query Classification datasets used in this thesis. Datasets that were produced by TREC or that contain crowdsourced relevance assessments are denoted with a ✔ in the associated column.

Table 5.4 reports the datasets that we use later in Chapter 7 to evaluate the NQC component of our news search framework. From Table 5.4, observe that we use two different datasets, one from May 2006 and one from April 2012. We denote these datasets as $NQC_{May2006}$ and $NQC_{Apr2012}$, respectively. The $NQC_{May2006}$ dataset contains newswire article, blog post, Wikipedia and query log corpora. However, it does not contain Digg or Twitter corpora, since at that time Twitter has not yet been launched and Digg was still in its infancy. Meanwhile, the later $NQC_{Apr2012}$ dataset contains newswire article, blog post, Digg, Twitter and Wikipedia corpora. However, it does not contain an aligned query log, since no Web search engine query logs have been made available to academia for that period.

The $NQC_{May2006}$ dataset contains 1,206 queries to be classified. These queries were sampled using a Possion sampling strategy (Ozmutlu *et al.*, 2004) from the $MSN_{May2006}$ query log corpus. For each of these queries we create ground truth labels identifying each as news-related or not. To create these classification labels, we employ crowdsourced workers to label each. This is the second of the tasks that we use crowdsourcing to achieve. We describe our methodology for creating these labels, the validation strategies that we employ and evaluate the quality of the assessments later in Section 5.5.

The $NQC_{Apr2012}$ dataset contains 2,935 queries from the 11th to the 23rd of April 2012. Unlike the queries used in the $NQC_{May2006}$ dataset, these queries are not a sample from a Web search engine query log from the period, since no such log is available to us from that period (see Figure 5.1). Instead, we build a simulated query log sample based upon the expected proportion of news to non-news queries from prior query log studies (see Section 3.5.2) and our own analysis (see Section 4.3). In particular, we use the Google Trends[1] analytics tool to collect 199 news-related queries. These 199 queries were initially chosen since they were unusually popular during the period. For verification, we manually matched each of these queries to an event, confirming that they were in fact news-related. These 199 queries are event-related, i.e. encompass the breaking, recent and long-running categories of news-related query from our taxonomy (see Section 4.3). For use as the fourth category — generic news queries — the author manually selected 26 such queries (with reference to generic news-related queries identified in Section 4.3). The 199 event-related and 26 generic news queries form the news-related portion of our simulated query log sample. To create the (larger) news-unrelated portion of the query log sample (between 89% and 93% from Section 4.3) we randomly sample 2,710 queries from the $MSN_{May2006}$ query log. Since these queries come from an older query log, the queries are very unlikely to be related to events from the time of the $NQC_{Apr2012}$ dataset. Combining the 199 event-related queries, 26 generic news queries and 2,710 news-unrelated queries forms our simulated query log sample. Note that we do not need to create additional news/non-news assessments for these queries,

[1]`http://google.com/trends/`

Dataset	TREC Dataset?	Crowdsourced Assessments?	# Queries	# of Assessments	Time Range	Corpus
$BlogTrack_{2010}^{TopNews-Phase2}$	✔	✔*	68	7,975	14/01/08 → 10/02/09	$Blogs08$
$MicroblogTrack_{2011}$	✔	✖	50	60,129	23/01/11 → 08/02/11	$Tweets2011$

Table 5.5: Ranking News-Related Content datasets used in this thesis. Datasets that were produced by TREC or that contain crowdsourced relevance assessments are denoted with a ✔ in the associated column. Where crowdsourced assessments is marked with a ✔* the dataset assessments were crowdsourced by ourselves on behalf of TREC.

as we have done so implicitly during the selection process, e.g. we know that the 199 Google Trends queries are news-related.

5.2.3 Ranking News-Related Content Datasets

In Chapter 8, we investigate the third component of our news search framework, i.e. the Ranking News-Related Content (RNRC) component. The aim of this component is to identify and rank content from different news and user-generated content sources for the user query. The top ranked of this content will later be integrated into the Web search ranking. In Section 4.7, we identified two types of content that can be challenging to effectively rank, namely blog posts and Twitter tweets. Naturally, we need to be able to effectively rank these documents, otherwise we risk retrieving and then integrating irrelevant content into the Web search ranking. Hence, we examine how to effectively rank both blog posts and tweets later in Chapter 8.

To evaluate these more typical document ranking tasks, we use two publicly available TREC datasets. Table 5.5 summarises the contents of each of these two datasets. In particular, to evaluate blog post ranking, we use the dataset from the blog post ranking phase (phase 2) of the TREC 2010 Blog track top news stories identification task (Ounis *et al.*, 2010). We denote this dataset as $BlogTrack_{2010}^{TopNews-Phase2}$. The $BlogTrack_{2010}^{TopNews-Phase2}$ dataset contains the $Blogs08$ blog post corpus, comprised of 28.5 million blog posts from the period of the 14 of January 2008 to the 10th of February 2009. To evaluate tweet ranking, we use the dataset from the TREC 2011 Microblog track (Ounis *et al.*, 2011), denoted $MicroblogTrack_{2011}$. This dataset contains the $Tweets2011$ twitter corpus that contains approximately 16 million tweets from the 23rd of January 2011 to the 8th of February 2011.

$BlogTrack_{2010}^{TopNews-Phase2}$ is a TREC dataset. It provides 68 news article headlines to rank blog posts for. We apply stopword removal and stemming to these news article headlines and use them as surrogates for news-related user queries (see Section 8.3.1). The dataset also provides 8,000 blog post ranking assessments for the 68 news article headlines, against which blog post ranking effectiveness can be measured. Importantly, from Table 5.5, observe that although $BlogTrack_{2010}^{TopNews-Phase2}$ is a TREC dataset, it uses relevance assessments crowdsourced by ourselves on behalf of TREC. Relevance assessment for the 8,000 blog posts in this dataset is the third of the tasks that we achieved using crowdsourcing. A description of the methodology we use for assessing blog posts for this dataset, the validation strategies that we employ and an evaluation the quality of the assessments produced are detailed later in Section 5.6.

The $MicroblogTrack_{2011}$ dataset comes with 49 tweet search queries developed by TREC. For each of these 49 queries, TREC assessors created a tweet ranking ground truth by judging tweets pooled (see Section 2.4.2) for that query (Ounis *et al.*, 2011; Soboroff *et al.*, 2012).

5.2.4 News-Related Content Integration Datasets

The final component of our news search framework is the News-Related Content Integration (NRCI) component. The aim of this component is to integrate news and user-generated content into the Web search ranking such that user satisfaction is increased over displaying the unaltered Web search ranking. To evaluate the NRCI component, we create a dataset containing rankings of documents from different news and user-generated corpora for a series of news-related queries. In our NRCI evaluation, the top ranked documents from each of the news and user-generated content rankings for a news-related query are integrated into a Web search ranking for that query, creating an enhanced ranking. The Web search ranking and enhanced ranking are shown to different users, who state their preference for one of the two rankings, facilitating evaluation.

We develop a single dataset to facilitate this evaluation, denoted $NRCI_{Apr2012}$. Table 5.6 provides a summary of the contents of this dataset.[1] From Table 5.6, we see that the dataset contains five corpora of different types, namely, newswire articles, blog posts, Digg posts, Twitter tweets and Wikipedia pages, from the month of April 2012. We reuse the news event-related portion of the queries contained within the $NQC_{Apr2012}$ dataset (see Section 5.2.2) as the news-related queries that we rank news and user-generated content for. This event-related portion contains 199 news queries from Google Trends, as described in Section 5.2.2.

For the 199 event-related news queries, documents from each of the five news and user-generated corpora within the dataset are ranked. However, these rankings may contain irrelevant documents, that, in turn, might influence our evaluation. To minimise the risk that irrelevant documents might be ranked and therefore remove a source of potential error in our evaluation, we filter out non-relevant documents from each of the rankings produced. In particular, for each document within the top ten results ranked from a source for one of the news-related queries, we had crowdsourced workers label each as relevant

[1]The number of documents ranked for each corpus that are reported in Table 5.6 are the counts after we filter the ranking using crowdsourcing.

Dataset	TREC Dataset?	Crowdsourced Assessments?	# Event-Related News Queries	Time-Range	Corpora Used	# Documents Ranked
$NRCI_{Apr2012}$	✖	✔	199	11/04/12 → 23/04/12	$BlekkoNewsSnippets_{Apr2012}$	1,553
					$BlekkoBlogSnippets_{Apr2012}$	1,857
					$DiggSnippets_{Apr2012}$	1,888
					$Tweets_{Apr2012}$	1,990
					$WikiUpdates_{Apr2012}$	902

Table 5.6: Summary of the $NRCI_{Apr2012}$ News-Related Content Integration dataset used in this thesis. Datasets that were produced by TREC or that contain crowdsourced relevance assessments are denoted with a ✔ in the associated column.

or not for that query. Any documents marked as irrelevant were removed from the ranking. After this filtering was applied, the remaining documents were added to the $NRCI_{Apr2012}$ dataset. It is the top documents from these rankings that we integrate into the Web search rankings for evaluation of the NRCI component later in Chapter 9. Notably, the filtering of each ranking that we just described is the fourth of the crowdsourced tasks that we tackled in this thesis. We describe the methodology that we use for assessing the relevance of each document, the validation strategies that we employ and an evaluation the quality of the assessments produced in Section 5.7.

5.2.5 Summary

In this section, we have provided an overview of the datasets that we use in the later experimental chapters of this thesis. During our discussion of these datasets, we noted four datasets where we used crowdsourcing to create assessments, namely: $BlogTrack_{2010}^{TopNews-Phase1}$; $NQC_{May2006}$; $BlogTrack_{2010}^{TopNews-Phase2}$; and $NRCI_{Apr2012}$. In fact, as a new tool to the field of information retrieval, a secondary contribution of this thesis is the development of crowdsourcing methodologies for dataset development. In the remainder of this chapter we provide a background into the field of crowdsourcing and describe the creation of assessments for each of these four datasets. In particular, in Section 5.3, we provide a short background into the field of crowdsourcing. Sections 5.4 to 5.7 detail how we crowdsourced assessments for each of the aforementioned datasets. In Section 5.8, we conclude on the findings of this chapter.

5.3 Crowdsourcing

We use crowdsourcing to facilitate the creation of new datasets with which to evaluate whether the use of user-generated content can aid in satisfying news-related queries submitted to universal Web search engines. Crowdsourcing in general is the act of outsourcing tasks, traditionally performed by a specialist person or group, to a large undefined group of people or community (referred to as the "crowd"), through an open call (Howe, 2010). There are many motivations for crowdsourcing tasks.

Dataset	CrowdFlower Analytics	Gold Judgement Validation	Work Summary Validation	Interface Validation	Redundant Judging	Task Description
$BlogTrack_{2010}^{TopNews-Phase1}$	✖	✖	✔	✔	✔	Identify newswire articles relating to important events.
$NQC_{May2006}$	✔	✔	✖	✔	✔	Classify queries as news-related or not.
$BlogTrack_{2010}^{TopNews-Phase2}$	✖	✔	✖	✖	✖	Assess blog posts as relevant or not to a newswire article.
$NRCI_{Apr2012}$	✔	✔	✖	✖	✔	Assess documents as relevant or not to a query.

Table 5.7: Summary of the tasks crowdsourced for each dataset.

For example, simple tasks can be completed at a relatively small cost, and often very quickly (Alonso *et al.*, 2008). Moreover, by employing a crowd of 'users' to do assessments as opposed to a few 'experts', a wider range of talent can be accessed and expert bias avoided (Howe, 2009). Indeed, in a TREC setting (see Section 2.4.3), it has been observed that relevance assessments by TREC assessors often differ from those produced by external assessors (Bailey *et al.*, 2008). However, crowdsourcing has also been the subject of much controversy as to its effectiveness. In particular, the work produced by crowdsourced workers is known to often be of low quality (Atwood, 2010) and results can be rendered meaningless by random or malicious work (Downs *et al.*, 2010).

Crowdsourcing is facilitated by a number of online marketplaces. These marketplaces allow individuals or organisations to post tasks to be crowdsourced and provide means for the 'crowd' to find and complete these tasks. There is a growing number of competing marketplaces available upon which tasks can be posted. Currently, the largest marketplace is Amazon's Mechanical Turk (MTurk)[1].

On MTurk, the individual or organisation who has work to be performed is referred to as the requester. People who sign up to perform tasks on MTurk are known as workers. A single task is comprised of smaller units, called Human Intelligence Tasks, or HITs. A HIT represents a small sub-task to be completed by one or more workers. A HIT tackled by one worker is known as an assignment. It has a small payment associated to it and an allotted completion time. Workers can see an example of any HIT that they are considering, along with the amount they will be paid for completing it and the time allotted.

MTurk provides requesters with a simple web service API. This API facilitates the submitting of tasks to the MTurk, the approval of completed tasks, and access to the answers generated (Alonso *et al.*, 2008). Notably, poor quality work need not be accepted. In particular, HITs can be rejected without payment and workers can be blocked from completing tasks.

An important feature of MTurk workers is that they are anonymous to the requester. However, requesters can specify filters regarding the geographical location of workers and their prior acceptance

[1]`http://www.mturk.com/`

rate on other HITs. It is noteworthy that the usefulness of worker prior acceptance rate as an indicator of work quality/commitment is contested as high acceptance rates can be generated artificially (Eickhoff & de Vries, 2011).

Other crowdsourcing marketplaces provide either similar functionality to MTurk, like CloudCrowd[1] or implement a wrapper around MTurk to provide enhanced services, e.g. CrowdFlower[2] for a cost. Still other crowdsourcing platforms use known workers instead of anonymous ones, for example ODesk[3]. In this thesis, we use Amazon's Mechanical Turk to produce assessments. However, for two of the four datasets that we use crowdsourcing to develop assessments, we employ the analytical tools provided by CrowdFlower.

As described in Section 5.2, we use crowdsourcing during the development of four datasets. Table 5.7 lists each of these datasets and states the task that is crowdsourced for each. In the remainder of this section, we provide a short overview of prior work in the field of crowdsourcing that are relevant to these four crowdsourced tasks. In particular, we discuss validation strategies for overcoming poor quality assessments by workers in Section 5.3.1, while we describe agreement measures that are used to assess the quality of crowdsourced assessments in Section 5.3.2. Section 5.3.3 describes how we structure our discussion of each crowdsourced task.

5.3.1 Work Validation Strategies

As noted earlier, one of the criticisms of crowdsourcing is that the resultant assessments can be of low quality. To counteract this, *validation* approaches have been proposed in the literature to identify and discard poor quality work and increase overall accuracy. Snow *et al.* (2008) and Callison-Burch (2009) investigated the accuracy of crowdsourced labels generated for natural language processing tasks. They concluded that 'expert' levels of labelling quality can be achieved by having three or five workers complete each crowdsourcing job and taking a majority (most commonly selected) label. Kittur *et al.* (2008) highlighted the need to validate the output produced by each worker, showing how label quality could be markedly improved by introducing questions with verifiable, quantitative answers, often referred to as 'gold judgements' or a 'honey-pot'. The aim of these questions is to validate a workers' output in an online manner. In this way, poorly performing workers can be detected and then ejected from tasks early on in the evaluation, saving money and hopefully improving the quality of the final labels produced. Subsequent research by Ipeirotis (2011) has indicated that within a Web page classification context, gold judgements are unnecessary when larger numbers of assessors work on each

[1]`http://www.cloudcrowd.com/`
[2]`http://crowdflower.com`
[3]`http://odesk.com`

task. In particular, their results indicate that when 10 assessors work on each task and a majority result is taken, then no appreciable gains in accuracy are observed when further adding a gold judgement validation.

In this thesis, when crowdsourcing assessments for our evaluation datasets, we use these validation approaches to assure their quality. For each of the datasets for which we crowdsource assessments, Table 5.7 lists the validation strategies employed. From prior works, gold judgement validation refers to the use of a honey pot, while redundant judging denotes where we had multiple workers perform each assessment and then aggregate the results produced (by majority vote). Interface validation refers to cases where the assessment interface itself contains an inbuilt validation mechanism, e.g. for news query classification we ask workers to provide a URL to a news article to support their assessment when identifying a query as news-related. Work summary validation refers to a validation technique we developed where the output from multiple workers for the same assessments are summarised and then manually validated (see Section 5.4.3).

5.3.2 Agreement Measures

When crowdsourcing, it is common to have multiple workers attempt each task. When doing so, one measure of resulting assessment quality is worker agreement, i.e. how often our workers made the same assessments for the same queries. The higher the agreement between workers, the better quality the assessments are thought to be (Ipeirotis, 2011). In this thesis, we employ two well-known measures for evaluating agreement in user evaluations: Free-Marginal Multirater Kappa (Randolph, 2005), denoted κ_{free}; and Fleiss Multirater Kappa (Fleiss, 1971), denoted κ_{fleiss}, as shown below:

$$\kappa_{free} = \frac{\frac{1}{Nn(n-1)}\left(\sum_{i=1}^{N}\sum_{j=1}^{k} n_{ij}^2 - Nn\right) - \frac{1}{k}}{1-\frac{1}{k}} \tag{5.1}$$

where N is the number of queries, n is the number of assessments per query k is the number of classes and n_{ij} is the current label. The second Kappa value is Fleiss Multirater Kappa (Fleiss, 1971), as shown below:

$$\kappa_{fleiss} =$$

$$\frac{\frac{1}{Nn(n-1)}\left(\sum_{i=1}^{N}\sum_{j=1}^{k} n_{ij}^2 - Nn\right) - \sum_{j=1^k}\left(\frac{1}{Nn}\sum_{i=1}^{N} n_{ij}\right)^2}{1-\sum_{j=1^k}\left(\frac{1}{Nn}\sum_{i=1}^{N} n_{ij}\right)^2} \tag{5.2}$$

where N is the number of queries, n is the number of assessments per query k is the number of classes and n_{ij} is the current label. Notably, the two Kappa agreement measures differ in the way each calculates the probability of agreement occurring by chance. Free-Marginal Multirater Kappa assumes that the chance of selecting a class is equal to one over the number of classes, while Fleiss Multirater Kappa takes into account the relative size of the classes. We report agreement as a measure of assessment quality for the three datasets for which we use redundant judging (see Table 5.7).

5.3.3 Structured Crowdsourced Evaluation

In the following four sections, we describe how we crowdsource assessments for the four datasets listed in Table 5.7. The primary aim of each of these sections is to show that the assessments produced for each dataset are of good quality. The secondary aim is to describe how we crowdsourced each of the assessments as a contribution in the field of crowdsourcing for IR.

A crowdsourced assessment task can be divided into five stages, namely: define the task; develop an assessment interface; prepare assessment validation; crowdsource the assessments; and quality assure those assessments. As such, we structure each of the following four sections in line with these stages. In particular, each section is comprised of five subsections. The first describes the task to be crowdsourced. The second details the assessment interface developed. The third describes how we validate the assessments. The fourth defines how we configured each crowdsourcing task. The fifth reports the quality of the assessments produced. Note that there is a difference between assessment validation and quality assurance. Assessment validation is employed to increase the quality of the resulting assessments, while quality assurance measures the quality of assessments after validation has been applied (normally in terms of agreement between workers, see Section 5.3.2).

5.4 Crowdsourced Task: Identifying Top Events

In this section, we describe how we crowdsource importance assessments for the newswire articles contained within the $BlogTrack_{2009}^{TopNews}$ dataset. We structure this section as described in Section 5.3.3. In particular, in Section 5.4.1, we describe the crowdsourcing task. Section 5.4.2 details the interface designed to facilitate the judging of newswire articles by workers. In Section 5.4.3, we describe how we validate the assessments produced by the workers. Section 5.4.4 lists the configuration of the crowdsourcing task. In Section 5.4.5, we discuss how accurate the resultant assessments are in terms of inter-worker agreement (see Section 5.3.2).

5.4.1 Task Description

The crowdsourcing task that we examine in this section is to judge whether a set of newswire articles describe important events on specific day, referred to as a topic day[1]. In particular, for a given newswire article, a day of interest and a news category (U.S./World/Sport/Business/Technology news), each crowdsourced worker assesses whether that newswire article is newsworthy or not for that day and category. Only stories that belong to the named category can be newsworthy. We use these news categories to reduce the assessment load on the workers. For example, it is easier to assess whether a newswire article is related to an important political story than to assess whether it is important in general. Each news story is assessed by workers as belonging to one of three classes:

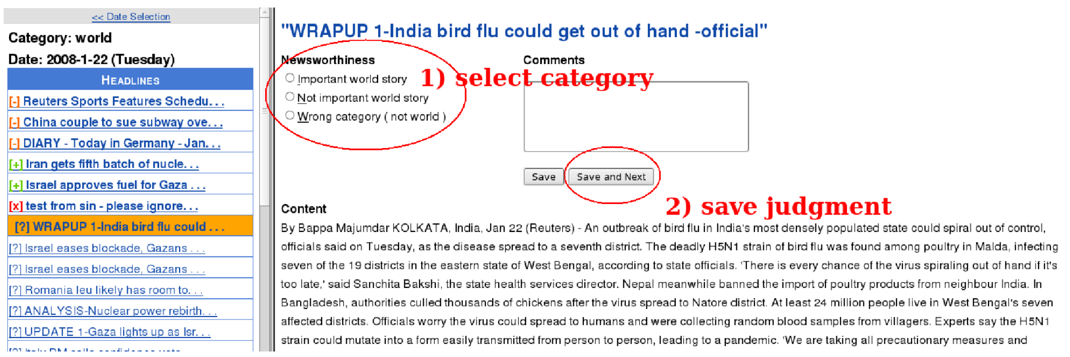
- Newsworthy and correct category: The story is newsworthy for the topic day and news category.
- Not newsworthy but correct category: The story is not particularly newsworthy for the topic day, but does match the news category.
- Incorrect category: The story belongs to a different news category.

Notably, for the purposes of the final assessments produced, the 'Not newsworthy but correct category' and 'Incorrect category' are both considered non-newsworthy. As shown earlier in Table 5.3, there are 8,000 newswire articles to be assessed, spread over 50 topic days. These articles were selected using a pooling strategy (see Section 2.4.2) over multiple news article rankings from different systems. In this case, statMAP sampling (Aslam & Pavlu, 2007) is used to a depth of 32 stories per day and category, resulting in 160 stories per day to be judged, with 8,000 stories in total (50 topics * 5 new categories * 32 newswire articles) (Ounis *et al.*, 2010).

5.4.2 Interface Design for News Article Assessment

To enable workers to assess each newswire article, we develop a new assessment interface. Figure 5.3 illustrates an example of the interface that we designed for the task. From Figure 5.3, we see that the current news category and day of interest is shown at top left. Meanwhile, down the left hand side we provide a listing of the newswire articles that are to be assessed. This summary is colour coded. A blue question mark in square brackets indicates that the newswire article is yet to be assessed, a green plus symbol in square brackets indicates that the newswire article has been judged as newsworthy for the stated category, an orange minus symbol in square brackets denotes that the newswire article was

[1]Authors Note: The day-centric nature of the assessments produced was due to a lack of granularity in the timestamps available from the TREC corpora.



Category: world

Date: 2008-1-22 (Tuesday)

- [-] Reuters Sports Features Schedu. . .
- [-] China couple to sue subway ove. . .
- [-] DIARY - Today in Germany - Jan. . .
- [+] Iran gets fifth batch of nucle. . .
- [+] Israel approves fuel for Gaza . . .
- [x] test from sin - please ignore. . .
- [?] WRAPUP 1-India bird flu could . . .
- [?] Israel eases blockade, Gazans . . .
- [?] Israel eases blockade, Gazans . . .
- [?] Romania leu likely has room to. . .
- [?] ANALYSIS-Nuclear power rebirth. . .
- [?] UPDATE 1-Gaza lights up as Isr. . .

"WRAPUP 1-India bird flu could get out of hand -official"

Newsworthiness

- ○ Important world story
- ○ Not important world story
- ○ Wrong category (not world)

Comments

Content

By Bappa Majumdar KOLKATA, India, Jan 22 (Reuters) - An outbreak of bird flu in India's most densely populated state could spiral out of control, officials said on Tuesday, as the disease spread to a seventh district. The deadly H5N1 strain of bird flu was found among poultry in Malda, infecting seven of the 19 districts in the eastern state of West Bengal, according to state officials. 'There is every chance of the virus spiraling out of hand if it's too late,' said Sanchita Bakshi, the state health services director. Nepal meanwhile banned the import of poultry products from neighbour India. In Bangladesh, authorities culled thousands of chickens after the virus spread to Natore district. At least 24 million people live in West Bengal's seven affected districts. Officials worry the virus could spread to humans and were collecting random blood samples from villagers. Experts say the H5N1 strain could mutate into a form easily transmitted from person to person, leading to a pandemic. 'We are taking all precautionary measures and

Figure 5.3: A screenshot of the external judging interface shown to workers within the instructions.

judged as not important but part of the correct category, while a red x in square brackets indicates that the newswire article does not belong to this category. On the right hand side, the current newswire article to be assessed is shown, including the headline and content of the article. One worker assesses all 32 of the newswire articles listed.

Notably, we have workers assess 32 newswire articles from a single day to provide them with some context regarding the day for which they are assessing. This is critical, since newswire article importance is to some extent relative in nature. For instance, a news story that would be considered ‘front-page’ news on most days can be buried by unexpected and/or high impact events, such as a celebrity death. To this end, we asked that workers make two passes over the newswire articles. During the first and longer pass, the worker would assess each newswire article based on the headline and content of that article and the previous articles assessed, while upon the second pass, the worker can change their assessment for any article now that they have knowledge of more newswire articles from that day. We use workers from the Amazon’s Mechanical Turk (MTurk) marketplace, hence each task instance is an MTurk HIT containing 32 newswire articles to be assessed.

5.4.3 Validation of Worker Assessments

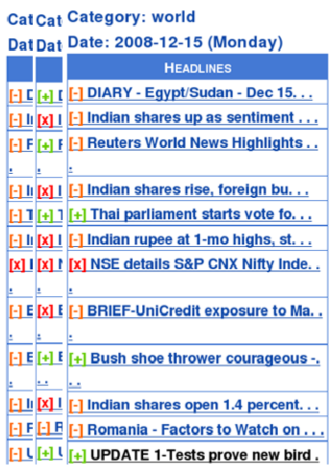
One of the criticisms of crowdsourcing is that it is susceptible to poor quality or malicious work. Best practises in crowdsourcing indicate that one or more validation strategies should be used to counteract this (Snow *et al.*, 2008). As such, we have three individual workers perform each HIT. From these three judgements we take the majority vote for each story to create the final newsworthiness assessment for that news story. Furthermore, to assure the quality of the resulting judgements, we manually validate the HITs produced using the same colour coded summaries of the stories and the judgements that each worker produced that were described in the previous section. In particular, the summary of the judgements for the three workers were displayed side-by-side, as illustrated in Figure 5.4 (a). We examined each set of 32 assessments produced by the workers based upon 3 criteria:

1. Are all 32 stories judged?

2. Are the judgements similar across the 3 redundant judgements?

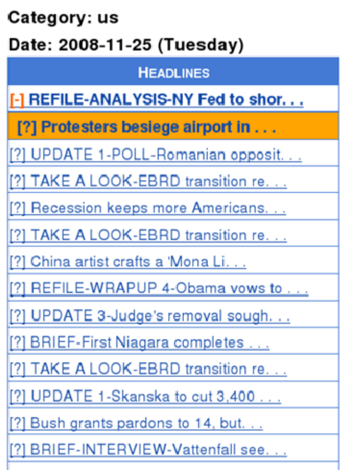
3. Are the stories marked important sensible?

Figure 5.4 (a) illustrates how the assessments produced by three workers for a single set of 32 newswire articles for the ‘world news’ category can be viewed as a colour-coded summary. We observe that in clear cut cases, like ‘NSE details S&P CNX Nifty Inde...’ shown in Figure 5.4 (a), which is clearly from the incorrect Business/Finance category, there is strong agreement between workers. In less clear cases, such as ‘Indian shares up as settlement...’ where the story could belong to either the World or Business/Finance categories, we observe some levels of disagreement. However, in this case, it is clear that the workers were completing the task in ‘good faith’ and hence the work was approved and paid for. On the other hand, Figure 5.5 (b) shows an example that we believe has been attempted by a bot, as only the first judgement was made before the HIT was submitted.



Cat Dat	Cat Dat	Category: world Date: 2008-12-15 (Monday)
		HEADLINES
[-] D	[+] D	[-] DIARY - Egypt/Sudan - Dec 15. . .
[-] In	[x] I	[-] Indian shares up as sentiment . . .
[-] F	[+] F	[-] Reuters World News Highlights . . .
[-] In	[x] I	[-] Indian shares rise, foreign bu. . .
[-] T	[+] T	[+] Thai parliament starts vote fo. . .
[-] In	[x] I	[-] Indian rupee at 1-mo highs, st. . .
[x] I	[x] N	[x] NSE details S&P CNX Nifty Inde. . .
[-] E	[x] E	[-] BRIEF-UniCredit exposure to Ma. . .
[-] E	[+] E	[+] Bush shoe thrower courageous -. . .
[-] In	[x] I	[-] Indian shares open 1.4 percent. . .
[-] F	[-] R	[-] Romania - Factors to Watch on . . .
[-] U	[+] U	[+] UPDATE 1-Tests prove new bird .

Figure 5.4: Displayed summary of three workers judgements for a single task instance.



Category: us

Date: 2008-11-25 (Tuesday)

HEADLINES
[-] REFILE-ANALYSIS-NY Fed to shor. . .
[?] Protesters besiege airport in . . .
[?] UPDATE 1-POLL-Romanian opposit. . .
[?] TAKE A LOOK-EBRD transition re. . .
[?] Recession keeps more Americans. . .
[?] TAKE A LOOK-EBRD transition re. . .
[?] China artist crafts a 'Mona Li. . .
[?] REFILE-WRAPUP 4-Obama vows to . . .
[?] UPDATE 3-Judge's removal sough. . .
[?] BRIEF-First Niagara completes . . .
[?] TAKE A LOOK-EBRD transition re. . .
[?] UPDATE 1-Skanska to cut 3,400 . . .
[?] Bush grants pardons to 14, but. . .
[?] BRIEF-INTERVIEW-Vattenfall see. . .

Figure 5.5: Task instance possibly completed by a bot.

Furthermore, we also note that our assessment interface has a passive protection against spammers and automatic bots that might attempt our task. In particular, our task requires the worker to select each news story in turn, assess that story, and then submit those assessments only when all 32 have been completed. It is unlikely that an automatic bot would be able to complete this successfully, since they would likely select submit before all of the stories had been assessed.

Category	Important	Not Important	Wrong Category	Agreement (Kappa Fleiss)
U.S. News	21%	39%	40%	63.53%
World News	24%	38%	38%	51.69%
Sport	21%	29%	49%	77.67%
Business/Finance News	24%	43%	33%	66.88%
Science/Technology	4%	10%	86%	82.97%
Average	19%	31%	49%	68.55%

Table 5.8: Judgement distribution and agreement on a per category basis.

5.4.4 Crowdsourcing Configuration

The crowdsourcing task that have described totals 24,000 story judgements (8,000 newswire articles * 3 workers per HIT) spread over 750 HIT instances. We paid our workers $0.50 (US dollars) per HIT (32 judgements), totalling $412.50 (including Amazon's 10% fees). For this task, we only used workers from the U.S.. Our reasoning is that international workers would likely not be able to accurately judge the newswire articles since the come from the $TRC2$ Reuters newswire corpus (see Table 5.3). Any incomplete HITs were rejected. As such, we collect exactly 3 judgements per story.

Following an iterative design methodology (Alonso *et al.*, 2008), we submitted our HITs in 6 distinct batches, allowing for feedback to be accumulated and HIT improvements to be made. We made minor modifications to the judging interface and updated the instructions based upon feedback from the workers. In the next section, we empirically evaluate the story ranking judgements produced. Screenshots of the instructions given to each worker are provided in Appendix B.

5.4.5 Assessment Accuracy

Recall that we have three individual workers assess each newswire article. To determine the quality of our assessments, we measure the agreement between our workers. Table 5.8 reports the percentage of judgements for each relevance label and the between-worker agreement in terms of Fleiss Kappa (Fleiss, 1971), on average, as well as for each of the five news categories. From Table 5.8, we observe that agreement on average is reasonable (69%). Meanwhile, we also observe that agreement varies markedly over news categories. For instance, the Science/Technology and Sport categories exhibit the highest agreement with 83% and 78% respectively, while the U.S. and World categories show less agreement. Based upon the class distribution for these categories, the disparity in agreement indicates that distinguishing science from non-science stories is easier than for the U.S. or World categories. This is intuitive, as the U.S. and World categories suffer from a much higher story overlap. For example, for the story "President meets world leaders regarding climate change", it is unclear whether it is a World and/or U.S. story. Hence, workers may disagree whether it should receive the 'important' or 'wrong category' label.

Overall, the reasonably high level of agreement between our workers, the US worker restriction that we employed and the manual validation strategy used all lend confidence to the quality of the judgements produced. Indeed, our agreement is greater than that observed in many studies of TREC assessments (Al-Maskari *et al.*, 2008). We therefore conclude that the assessments produced are of sufficient accuracy to be usable to evaluate top news identification approaches later in Chapter 6.

5.5 Crowdsourcing Task: Classifying User Queries

This section details how we use crowdsourcing to generate news query classification labels for the $NQC_{May2006}$ dataset. We structure this section as follows. In Section 5.5.1, we describe the crowdsourcing task of news query classification as tackled in this section. Section 5.5.2 describes the interfaces that we develop to enable workers to classify queries as news-related or not. In Section 5.5.3, we detail the steps we take to validate the assessments produced by the workers. Section 5.5.4 describes the configuration of our crowdsourcing tasks, while in Section 5.5.5, we report the accuracy of the news query classification labels produced.

5.5.1 Task Description

We aim to label each of a set of queries as news-related or not for a point in time. Each query in the set has a timestamp to the nearest second denoting when that query was submitted to a Web search engine. We use real queries sampled from American users of the MSN Search engine (now Microsoft Bing) query log from 2006 (Craswell *et al.*, 2009).

In particular, we sample two sets of queries: a small query set, which we refer to as NQC^{test}_{2006}, as well as the larger set of queries to be included in our final dataset, denoted NQC^{full}_{2006}. NQC^{test}_{2006}, is a small set of queries that we use later in this section for iterative development of the assessment interface. Iterative interface development is advantageous, since it allows different interfaces to be prototyped and tested at a low cost, with the aim of finding the interface that results in the highest quality assessments. NQC^{full}_{2006} are the target queries to be labelled from the $NQC_{May2006}$ dataset (see Table 5.4). We use a Poisson sampling strategy (Ozmutlu *et al.*, 2004) to create both of the NQC^{test}_{2006} and NQC^{full}_{2006} query sets from the MSN query log sample. Statistics for both the query-sets created are shown in Table 5.9. As we can see, the NQC^{test}_{2006} set contain 91 queries, while the NQC^{full}_{2006} set contains a further 1,206 queries.

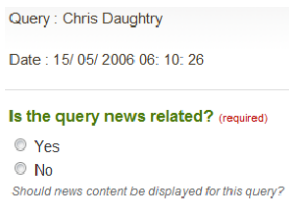
	MSN query log	NQC_{2006}^{full}	NQC_{2006}^{test}
Time Range	01/05 to 31/05	01/05 to 31/05	15/05
Number of Queries	14,921,286	1206	91
Mean Queries per Day	481,331	38.9	91
Mean Query Length	2.29	2.39	2.49

Table 5.9: Statistics for the MSN query log from 2006, as well as the sampled NQC_{2006}^{test} and NQC_{2006}^{full}.

5.5.2 Interface Design for News Query Classification

When classifying news-related queries, it is important to consider the temporal aspects of the task. In particular, the news-relatedness of a query can change over time. For example, the query 'ash' might be news-related if there has been a recent volcano eruption, but at other times might not be considered news-related. Hence, to enable users to accurately assess the news-relatedness of a query, we need to design an interface that enables queries to be labelled and provides information about the important stories at the time. However, there are different ways in which we can expose information about important stories to the user. Hence, we propose four potential labelling interfaces and test each later on the NQC_{2006}^{test} queries.

The four labelling interfaces are illustrated in Figure 5.6. Figure 5.6 (a) illustrates the first of the four interfaces, denoted *Basic*. It simply shows the query in addition to the date and time it was made, and asks the worker whether that query is news-related or not. This acts as our baseline approach that provides the worker with no additional information. Our second interface, denoted *Headline_Inline* and illustrated by Figure 5.6 (b), additionally incorporates news headlines published by the New York Times from the day of the query into our interface design. These are all of the headlines that were listed on the homepage for the New York Times on the day of each query, 12 in this example. By providing the workers with information about the news stories from around the time of the query, we aim to better inform our workers when completing the task and hence increase labelling accuracy. Our third interface, illustrated by Figure 5.6 (c), further adds a snippet describing the news article for each of the news headlines. We denote this interface as *Headline+Summary*. The aim is to provide more detailed information about each story such that the workers can better recognise queries relating to those stories. Our fourth interface, illustrated by Figure 5.6 (d) incorporates automatically generated search links to the three top search engines, i.e. Bing, Google and Yahoo!, for each query and time. We denote this interface *Link-Supported*. Each link initiates a search by the associated search engine. The query for each search constitutes the original user query and the date when the query was submitted, e.g. '4th May 2006 Moussaoui Verdict'. In this way, we cause the Web search engine to return results from the day of the query. Furthermore, for the *Link-Supported* interface, when labelling a query as news-related, we



Query : Chris Daughtry

Date : 15/ 05/ 2006 06: 10: 26

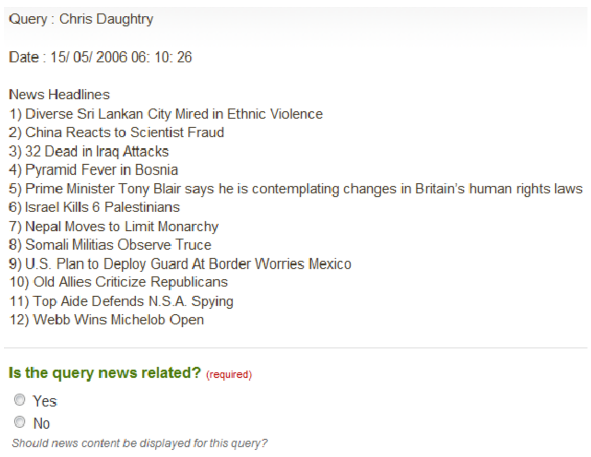
Is the query news related? (required)

Yes

No

Should news content be displayed for this query?

(a) Basic



Query : Chris Daughtry

Date : 15/ 05/ 2006 06: 10: 26

News Headlines
1) Diverse Sri Lankan City Mired in Ethnic Violence
2) China Reacts to Scientist Fraud
3) 32 Dead in Iraq Attacks
4) Pyramid Fever in Bosnia
5) Prime Minister Tony Blair says he is contemplating changes in Britain's human rights laws
6) Israel Kills 6 Palestinians
7) Nepal Moves to Limit Monarchy
8) Somali Militias Observe Truce
9) U.S. Plan to Deploy Guard At Border Worries Mexico
10) Old Allies Criticize Republicans
11) Top Aide Defends N.S.A. Spying
12) Webb Wins Michelob Open

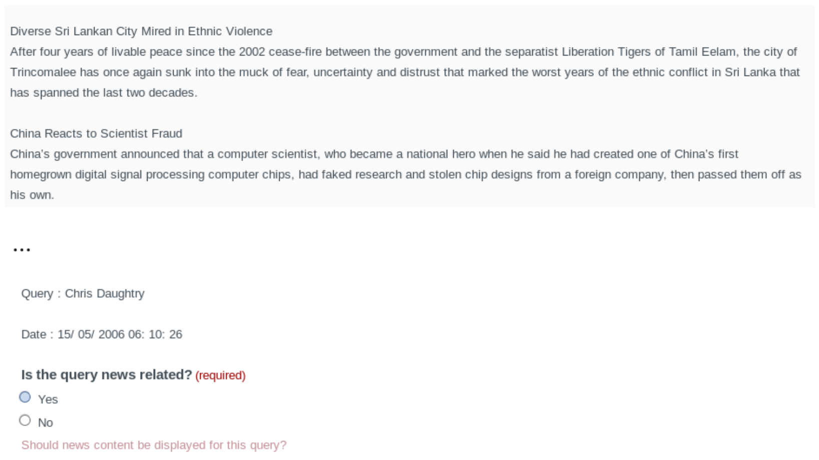
Is the query news related? (required)

Yes

No

Should news content be displayed for this query?

(b) Headline_Inline



Diverse Sri Lankan City Mired in Ethnic Violence
After four years of livable peace since the 2002 cease-fire between the government and the separatist Liberation Tigers of Tamil Eelam, the city of Trincomalee has once again sunk into the muck of fear, uncertainty and distrust that marked the worst years of the ethnic conflict in Sri Lanka that has spanned the last two decades.

China Reacts to Scientist Fraud
China's government announced that a computer scientist, who became a national hero when he said he had created one of China's first homegrown digital signal processing computer chips, had faked research and stolen chip designs from a foreign company, then passed them off as his own.

...

Query : Chris Daughtry

Date : 15/ 05/ 2006 06: 10: 26

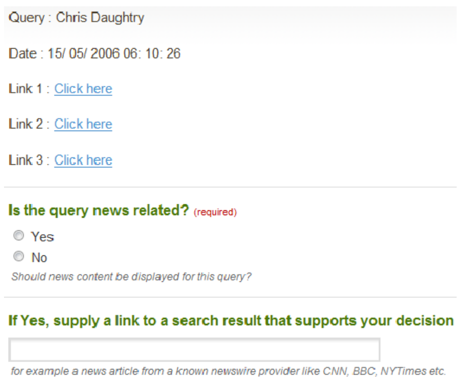
Is the query news related? (required)

Yes

No

Should news content be displayed for this query?

(c) Headline+Summary



Query : Chris Daughtry

Date : 15/ 05/ 2006 06: 10: 26

Link 1 : Click here

Link 2 : Click here

Link 3 : Click here

Is the query news related? (required)

Yes

No

Should news content be displayed for this query?

If Yes, supply a link to a search result that supports your decision

for example a news article from a known newswire provider like CNN, BBC, NYTimes etc.

(d) Link-Supported

Figure 5.6: News query classification instances used in this thesis. Not all of the news story summaries are shown in the Headline+Summary interface to save space.

	NQC_{2006}^{full}	NQC_{2006}^{test}
Time Range	01/05 to 31/05	15/05
Number of Queries	61	9
Mean Queries per Day	1.97	9
% of Target Query-set	5%	10%
News-Related Queries	47	5
Non-News-Related Queries	14	4

Table 5.10: Statistics for the validation sets created for the NQC_{2006}^{test} and NQC_{2006}^{full}.

also ask our workers to provide the URL of a supporting result from the search engine rankings that they view. Collecting this information enables us to perform an additional layer of validation as described in the next sub-section.

5.5.3 Validation of Worker Labelling

Earlier in Section 5.3.1, we raised the possibility that workers might label our queries in a random or malicious manner (Kittur *et al.*, 2008). To address this, we perform online validation of our worker labels against gold standard query-label pairs, i.e. a 'honey pot' (Kittur *et al.*, 2008). If a worker tries to 'game' our system by randomly labelling queries, they can be identified and ejected from the evaluation based upon a low accuracy on the ground-truth queries. Statistics for our two validation sets are shown in Table 5.10. When using the *Link-Supported* interface, we also use the URL supplied by the worker to check whether that worker has indeed clicked out and viewed related pages for the query. Assessments where no URL or random text was entered into the URL box were automatically rejected. When validating on the NQC_{2006}^{full} query set, 2.1% of assessments were rejected in this manner, indicating that workers were attempting the task in good faith.

5.5.4 Crowdsourcing Configuration

We use Amazon's Mechanical Turk (MTurk) to crowdsource our assessments. When submitting a job, there are multiple variables that determine how it is handled by MTurk. Firstly, the requester must specify the amount paid to each worker on a per job basis. In our case, we paid workers based on the estimated amount of time it would take to judge each query at a rate of \$2 (US dollars) per hour. The amount paid per HIT is dependant upon the interface design used, since some interfaces present more information that the user needs to read before making the assessment. We pay the least for the *Basic* interface at \$0.01, while the *Link-Supported* interface costs the most, at \$0.08. The total cost for iterative development on the NQC_{2006}^{test} query set was \$24.62. Meanwhile, the total cost for the NQC_{2006}^{full} query set was \$320.58. Secondly, the requester has the option to limit the worker pool based on geographical location. Since the MSN query log is predominantly American, we limited our worker pool to the USA only. To control validation, we set the validation cut-off, i.e. the level at which a worker is ejected from the evaluation, to 70% (should they get more than 30% of the validation queries wrong they are ejected from the evaluation without remuneration). Lastly, we set the level of redundancy for judging our queries at three, whereby each query will be judged by three unique workers. We take a majority vote to determine the final label.

5.5.5 Label Quality

The aim of this section is to determine whether the news query classification labels produced by our workers are of sufficient quality to be used in our later experiments in Chapter 7. Recall that we previously proposed four different labelling interfaces. We begin by evaluating each of the interfaces on the smaller test query set (NQC^{test}_{2006}). Once we have concluded which of the interfaces results in the most accurate labels, we then use that interface to label the larger NQC^{full}_{2006} query set and report the accuracy for it.

We first report the quality of the labels produced on the NQC^{test}_{2006} set for each interface in terms of worker agreement and accuracy against a small ground truth query set labelled by the author. Importantly, there is no de facto standard for defining acceptable or significant levels of agreement. However, Landis and Koch (Landis & Koch, 1977) state that Kappa values over 0.61 indicate substantial levels of agreement, while values over 0.81 represent almost perfect agreement. Table 5.11 reports two Kappa agreement measures, κ_{free} and κ_{fleiss} for labelling the NQC^{test}_{2006} queries with each of the four interfaces. From Table 5.11, providing our workers with either news headlines (*Headline_Inline*), news summaries (*Headline+Summary*) or a ranking of search results (*Link-Supported*) from the time of the query, we can markedly increase worker agreement over using the *Basic* interface. In general, we observe that our *Link-Supported* interface obtained the highest worker agreement, i.e. $\kappa_{free} = 0.8367$ and $\kappa_{fleiss} = 0.5341$. It also accrued a high labelling accuracy of 0.9684. As such, we chose this interface to crowdsource our larger NQC^{full}_{2006} query set.

The final row in Table 5.11 reports the worker agreement and label accuracy for the NQC^{full}_{2006} query set using our *Link-Supported* interface. From Table 5.11, we see that the accuracy of our crowdsourced labels is high, i.e. over 90%. Additionally, agreement between our workers is also high, with $\kappa_{free} = 0.8358$ and $\kappa_{fleiss} = 0.7677$. Overall, from the high agreement between the workers and accuracy against the ground truth assessments, we conclude that the news query classification labels produced are of sufficient quality to be used to evaluate the news query classification approaches. Indeed, we later use these assessments in Chapter 7 of this thesis.

Interface	Query Set	Accuracy	κ_{free}	κ_{fleiss}
Basic	NQC_{2006}^{test}	**0.9684**	0.7373	0.3525
Headline_Inline	NQC_{2006}^{test}	0.9474	0.7866	0.3018
Headline+Summary	NQC_{2006}^{test}	**0.9684**	0.83	0.5327
Link-Supported	NQC_{2006}^{test}	**0.9684**	**0.8367**	**0.5341**
Link-Supported	NQC_{2006}^{full}	**0.9748**	**0.8358**	**0.7677**

Table 5.11: Quality measures for news query classification on the approximately 100 query NQC_{2006}^{test} with varying interfaces, in addition to the over 1000 query NQC_{2006}^{full} using the *Link-Supported* interface.

5.6 Crowdsourcing Task: Ranking Blog Posts for a News Story

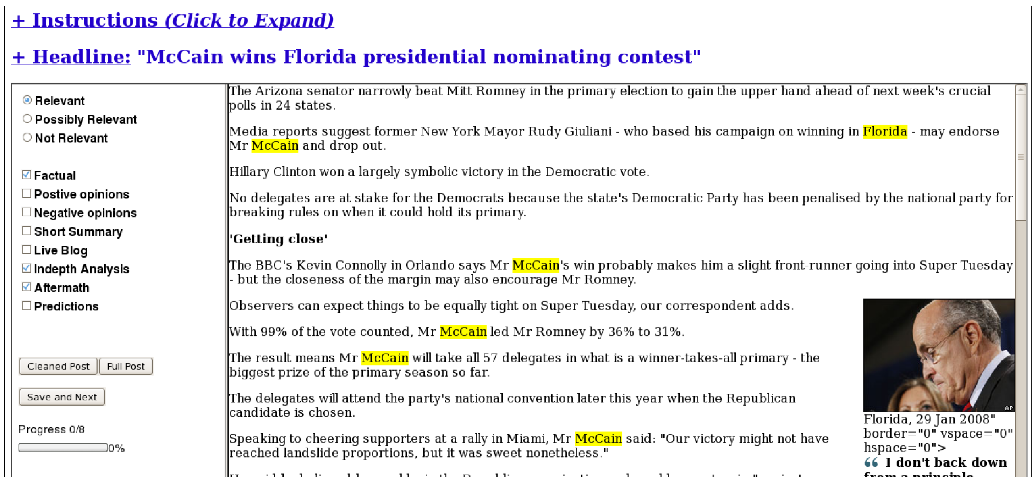
In this section, we describe how we crowdsource the blog post ranking assessments as part of the $BlogTrack_{2010}^{TopNews-Phase2}$ dataset. As before, we structure the section as follows. We begin by describing the task to be crowdsourced in Section 5.6.1. Section 5.6.2 details the interface that we use to collect blog post ranking assessments. In Section 5.6.3, we describe how we validate the assessments produced by our workers. Section 5.6.4 discusses the configuration of our crowdsourced tasks. Finally, in Section 5.6.5, we report on the quality of the assessments produced.

5.6.1 Task Description

Recall from Table 5.5 that $BlogTrack_{2010}^{TopNews-Phase2}$ is a TREC dataset, developed for the 2010 top news stories identification task, rather than specifically developed to evaluate a component of our framework. The TREC task investigated both blog post ranking and diversification for news article headlines (Ounis *et al.*, 2010). Hence, the crowdsourced assessment task is to judge each of the pooled blog posts as *relevant*, *possibly relevant* or *not relevant* to a newswire article (facilitating relevance evaluation), and also to suggest perspectives that describe each blog post (facilitating diversity evaluation)[1]. Here, perspectives come from a set of 9 categories that a blog post might be considered to belong to, e.g. a blog post might be considered to contain republican or democrat viewpoints for political stories. The nine perspectives that were defined by TREC for the task are: Factual Account; Opinionated Positive; Opinionated Negative; Opinionated Mixed; Short summary/Quick bites; Live Blog; In-depth analysis; Aftermath; and Predictions.

In total, 7,975 blog posts from the $Blogs08$ corpus are to be assessed (see Table 5.5). These blog posts were selected via a pooling strategy for 68 newswire articles from the TRC2 Thomson Reuters newswire corpus (see Section 5.2.3). In particular, for all 68 newswire articles, a series of blog post

[1]The approaches for blog post ranking that we investigate in Chapter 8 focus on relevancy and do not attempt diversification, unlike the TREC task.



+ Instructions *(Click to Expand)*

+ Headline: "McCain wins Florida presidential nominating contest"

[✓ Relevant]
Possibly Relevant
Not Relevant

[✓ Factual]
Postive opinions
Negative opinions
Short Summary
Live Blog
[✓ Indepth Analysis]
[✓ Aftermath]
Predictions

Progress 0/8

0%

The Arizona senator narrowly beat Mitt Romney in the primary election to gain the upper hand ahead of next week's crucial polls in 24 states.

Media reports suggest former New York Mayor Rudy Giuliani - who based his campaign on winning in Florida - may endorse Mr McCain and drop out.

Hillary Clinton won a largely symbolic victory in the Democratic vote.

No delegates are at stake for the Democrats because the state's Democratic Party has been penalised by the national party for breaking rules on when it could hold its primary.

'Getting close'

The BBC's Kevin Connolly in Orlando says Mr McCain's win probably makes him a slight front-runner going into Super Tuesday - but the closeness of the margin may also encourage Mr Romney.

Observers can expect things to be equally tight on Super Tuesday, our correspondent adds.

With 99% of the vote counted, Mr McCain led Mr Romney by 36% to 31%.

The result means Mr McCain will take all 57 delegates in what is a winner-takes-all primary - the biggest prize of the primary season so far.

The delegates will attend the party's national convention later this year when the Republican candidate is chosen.

Speaking to cheering supporters at a rally in Miami, Mr McCain said: "Our victory might not have reached landslide proportions, but it was sweet nonetheless."

Florida, 29 Jan 2008" border="0" vspace="0" hspace="0">

“ I don't back down

Figure 5.7: A screenshot of the external judging interface shown to workers within the instructions.

ranking systems (submitted to TREC 2010) produced three rankings of blog posts using a representation of those newswire articles as queries. One ranking containing posts published before each newswire article, one containing blog posts from the following day and before and one containing posts from a week following and before. This represents a system ranking for a newswire article at different times, i.e. as the story that the newswire article refers to matures. The top 20 posts from each of these three rankings were combined to form the 7,975 blog post pool to be assessed.

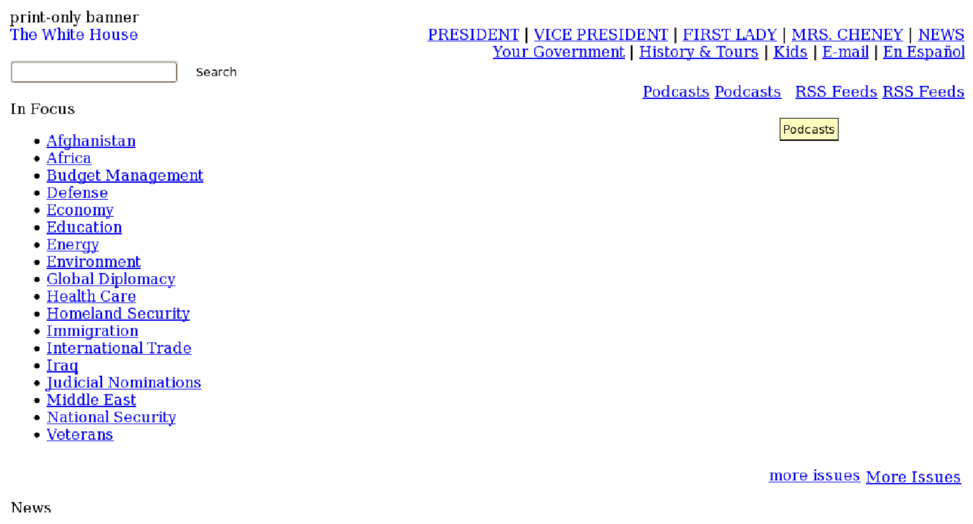
5.6.2 Interface Design for Blog Post Ranking

To crowdsource assessments for each of the 7,975 blog posts, we develop an assessment interface that both renders the blog posts to the user and records that user's assessment. To develop this interface, we follow the iterative design methodology proposed by Alonso & Baeza-Yates (2011). In particular, we first create a small test set comprised of 200 blog posts returned for two news stories that we manually assess. We iteratively develop different prototype interfaces, evaluate worker performance and subsequently made improvements. For brevity, we omit the intervening iterations of the interface. Figure 5.7 illustrates the final interface produced. From Figure 5.7, we see that the interface is divided into three components: the instructions and newswire article headline at the top; the assessment options down the left hand side and the rendering of the post to be assessed on the right hand side. In this example, at the top of the interface, the instructions are hidden. Clicking on the instructions within the interface reveals them to the user. See Appendix B.3 for the full instruction set provided to each worker.

Notably, one outcome of the iterative design process was that we increased the number of assessments that each worker makes in one sitting to 20. Indeed, when assessing the 200 blog posts in our small test set, we found that the completion time when assessing 20 posts was reduced by 79.2% over assessing a single post at a time. We also observed that the rendering of blog posts from the $Blogs08$ corpus can be difficult. In particular, the blog posts within that corpus only contain the raw HTML. When rendering, additional content such as images and CSS files need to be loaded from the original website, which may have changed or might have been removed. For example, Figure 5.8 (a) illustrates an example blog post when presented in original HTML form. As we can see, the page template has changed markedly, such that the main page content is no longer visible (one would need to scroll down to get to the main article content). To counteract this, as well as to decrease loading delays, by default we show workers a cleaned version of the Web page, created by extracting only the text within $< h >$ or $< p >$ tags. Indeed, Figure 5.8 (b) shows the cleaned version of the same page. However, we do provide a full HTML rendering that can be loaded by pressing the 'Full Post' button on the left component of the interface (see Figure 5.7), in case the text cleaning mistakenly removes the main content of the article.

5.6.3 Validation of Worker Assessments

As before, to ensure the quality of the relevance assessments produced, we employ validation strategies. In this case, we use a form of gold judgement validation. In particular, typical gold standard validation involves the prior creation of a gold-standard judgement set, with which to test workers, known as a 'honey pot' (see Section 5.3.2). Our approach is similar, except that we wait until all of the work is completed, such that we can better identify those documents that workers found difficult, i.e. disagreed on. For each set of 20 blog posts, we select three posts to be validated against a gold standard, i.e. one judged relevant, one judged possibly relevant and one judged not relevant. For each of these selected posts, the author assessed their relevance to the associated newswire article, forming the gold standard. If more than one of these did not match this gold standard, then all 20 assessments were rejected and re-posted for another worker to complete on the grounds that the work was not of sufficient quality. Overall, this resulted in a gold standard set of roughly 15% of all blog posts to be assessed (1197/7975), and took within the region of 8 hours to create. This is naturally longer than it would take to create a normal 5% gold standard set. However, by using a larger and more evenly distributed gold standard, we have greater confidence in the reliability of the gold standard, and hence the quality of the resulting assessments.



print-only banner
The White House

PRESIDENT | VICE PRESIDENT | FIRST LADY | MRS. CHENEY | NEWS
Your Government | History & Tours | Kids | E-mail | En Español

Search

Podcasts Podcasts RSS Feeds RSS Feeds

Podcasts

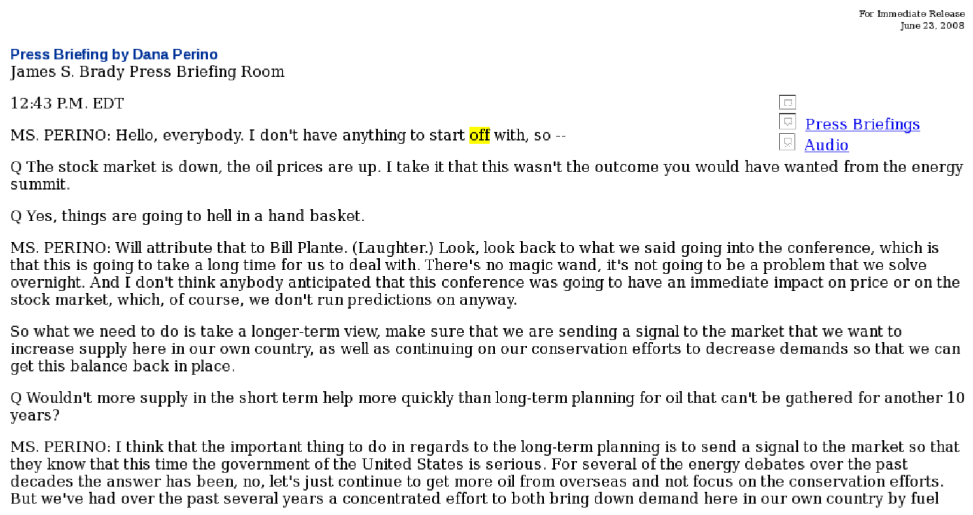
In Focus

- Afghanistan
- Africa
- Budget Management
- Defense
- Economy
- Education
- Energy
- Environment
- Global Diplomacy
- Health Care
- Homeland Security
- Immigration
- International Trade
- Iraq
- Judicial Nominations
- Middle East
- National Security
- Veterans

more issues More Issues

News

(a) HTML.



For Immediate Release
June 23, 2008

Press Briefing by Dana Perino
James S. Brady Press Briefing Room

12:43 P.M. EDT

Press Briefings
Audio

MS. PERINO: Hello, everybody. I don't have anything to start off with, so --

Q The stock market is down, the oil prices are up. I take it that this wasn't the outcome you would have wanted from the energy summit.

Q Yes, things are going to hell in a hand basket.

MS. PERINO: Will attribute that to Bill Plante. (Laughter.) Look, look back to what we said going into the conference, which is that this is going to take a long time for us to deal with. There's no magic wand, it's not going to be a problem that we solve overnight. And I don't think anybody anticipated that this conference was going to have an immediate impact on price or on the stock market, which, of course, we don't run predictions on anyway.

So what we need to do is take a longer-term view, make sure that we are sending a signal to the market that we want to increase supply here in our own country, as well as continuing on our conservation efforts to decrease demands so that we can get this balance back in place.

Q Wouldn't more supply in the short term help more quickly than long-term planning for oil that can't be gathered for another 10 years?

MS. PERINO: I think that the important thing to do in regards to the long-term planning is to send a signal to the market so that they know that this time the government of the United States is serious. For several of the energy debates over the past decades the answer has been, no, let's just continue to get more oil from overseas and not focus on the conservation efforts. But we've had over the past several years a concentrated effort to both bring down demand here in our own country by fuel

(b) Cleaned.

Figure 5.8: An example blog post rendering both when cleaned and as HTML. The HTML rendering contains the same content as the cleaned version, however, due to a missing CSS template one would need to scroll down to get to it.

Batch	# Stories	# HITs	# Judgements	Pay Per HIT	Hourly-rate ($)
Batch 1	1→5	27	540	0.50	2.27
Batch 2	6→15	66	1320	0.50	3.85
Batch 3	16→28	78	1560	0.50	3.90
Batch 4	29→50	137	2740	0.50	3.58
Batch 5	51→68	102	2040	0.50	2.27
Batch 6	68	68	80	0.25	6.07

Table 5.12: Average amount paid per hour to workers and work composition for each batch of HITs.

5.6.4 Crowdsourcing Configuration

For crowdsourcing, we use the Amazon Mechanical Turk (MTurk) marketplace. Recall that in total we assess 7,975 blog posts from 68 newswire articles. We spread these over 433 MTurk HIT instances, each containing approximately 20 posts. We pay our workers $0.50 (US dollars) per HIT for the 20 assessments. The total cost is $238.70 (including Amazon's 10% fees). Due to the larger and more thorough gold standard validation set employed, we have only a single worker assess each HIT. We did not restrict worker selection based on geography, however only workers with a prior 95% acceptance rate were accepted for this task.

Continuing with an iterative methodology (Alonso *et al.*, 2008), we submitted our HITs in 6 distinct batches, allowing for feedback to be accumulated and HIT improvements to be made. The first five batches were comprised of HITs containing 20 blog posts to be judged. The sixth batch contained all of the remaining blog posts for each news story. Table 5.12 reports the statistics and per-hourly rate paid to workers during each of the six batches.

5.6.5 Assessment Quality

For the blog post ranking task, we had a single worker judge each blog post. As such, we cannot use inter-worker agreement to estimate final quality (at least three workers per blog post would be required). Instead, to determine the quality of our resulting relevance assessments, we compare them against assessments produced by the author as a ground truth. In particular, we randomly sampled 5% of the blog post set judged (360/7975) and manually assessed each post in terms of its relevancy to the associated news story. Note that this is not the same as the gold standard used to validate the workers during the production of the relevance assessment (see Section 5.6.3), but is a different set used to evaluate the quality of the final relevance assessments produced. Table 5.13 reports the accuracy of the crowdsourced blog post relevance judgements in comparison to the aforementioned ground truth, both overall (All), and in terms of each relevance grade.

Relevance Grade	Accuracy
Relevant	62.43%
Probably Relevant	50.00%
Not Relevant	74.64%
All	66.63%

Table 5.13: Accuracy of blog post assessments in comparison to the track organiser judgements.

We observe a reasonable overall accuracy of 66.63%, i.e. 2/3 crowdsourced judgements exactly matched our ground truth. This indicates that by using a single crowdsourced worker, we can still generate assessments of reasonable quality. Furthermore, we see that a disproportionate number of incorrect judgements belonged to 'Probably Relevant' category. This is to be expected, as blog posts that fall into this category were difficult to assess for our track organisers. To resolve this, we collapse these three-way graded assessments into a binary form, by combing the 'Probably Relevant' and 'Not Relevant' categories together. By doing so, we observe a markedly higher worker accuracy of 76.73%. Indeed, based upon this higher level of accuracy and the resilience of Cranfield-style evaluation when using many topics to variances in assessment quality (Bailey *et al.*, 2008), we conclude that the relevance labels produced are of sufficient quality to use in our subsequent evaluation of blog post ranking approaches in Chapter 8.

5.7 Crowdsourced Task: Document Relevance Assessment

In this section, we describe how we crowdsource relevance assessments for a series of documents ranked from different sources as part of the $NRCI_{Apr2012}$ dataset. Recall from Section 5.2.4 that for this dataset, we use crowdsourcing to filter out irrelevant documents for rankings of news and user-generated content that we integrate into the Web search ranking in our final experimental chapter (Chapter 9). We structure this section into five sections as follows. In Section 5.7.1, we describe the crowdsourcing task. Section 5.7.2 details the assessment interface developed. In Section 5.7.3 we describe how we validate the assessments produced. Section 5.7.4 details the setup of the crowdsourcing task, while Section 5.7.5 reports the quality of the result assessments.

5.7.1 Task Description

The crowdsourcing task that we tackle in this section is to assess documents as *relevant*, *possibly relevant* or *not relevant* for a news-related user query. In this case, the documents can come from any one of five news and user-generated content sources, namely newswire, the blogosphere, Twitter, Digg or Wikipedia. Documents are uniformly represented as title and snippet pairs, with the exception of Twitter

Corpus	Document Type	# Documents
$BlekkoNewsSnippets_{Apr2012}$	Newswire Articles	1,553
$BlekkoBlogSnippets_{Apr2012}$	Blog Posts	1,857
$DiggSnippets_{Apr2012}$	Digg Posts	1,888
$Tweets_{Apr2012}$	Twitter Tweets	1,990
$WikiUpdates_{Apr2012}$	Wikipedia Pages	902
Total		8,190

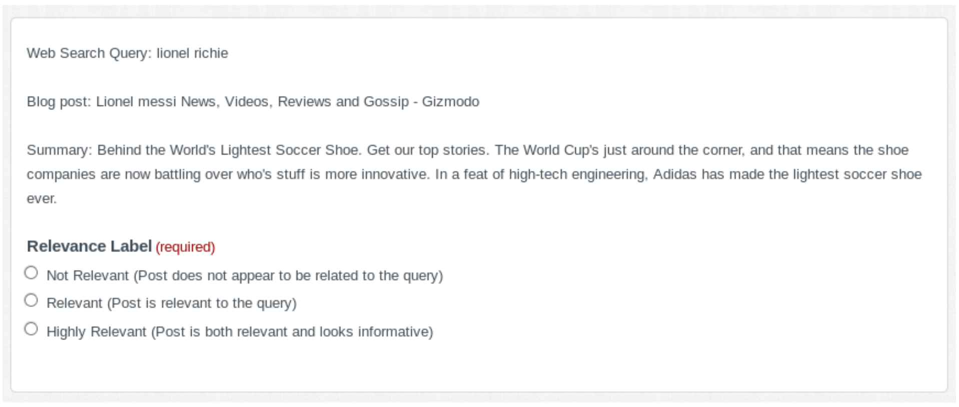
Table 5.14: Statistics of the documents to be assessed from each of the five news and user-generated content sources.

tweets that only contain a 'title' representing the text of the tweet. The news-related queries for which the documents are to be assessed come from the Google Trends analysis tool (see Section 5.2.4).

There are 8,190 documents to be assessed in total over the 199 queries (see Table 5.6). These documents come from pooling the top 10 rankings produced for each of the 199 queries by a single search engine for each of the five news and user-generated content rankings (the ranking approaches employed are described later in Section 9.5.2). Note that not all of the queries retrieved 10 documents since some news and user-generated content sources may contain fewer than ten related documents within the time window considered, hence the uneven total number of documents. Table 5.14 lists the corpora that the documents come from and the number of documents from each of those sources. From Table 5.14, we see that of the five sources, the blog post, Digg and Twitter corpora returned between 1,850 and the maximum 1,990 documents over the 199 news-related queries. The newswire article corpus returned fewer documents (1,553), while Wikipedia returned the fewest with only 902 documents.

5.7.2 Interface Design for Document Relevance Assessment

To enable workers to assess each of the 8,190 documents we develop a common interface that we vary slightly for each of the five different type of documents that we have assessed. This interface is comprised of two components. Firstly, an instruction block describing how the worker is to complete the task is provided at the top. An illustration of the instructions can be found in Appendix B.4. Below the instruction block are a series or assessment blocks. Each assessment block represents a single assessment that the worker is to make. Figure 5.9 illustrates how documents from three of the five sources are displayed to the user as assessment blocks. From Figure 5.9, we see that in each interface the news-related query is displayed at the top. Below the query, the document title and snippet pair is displayed. For Figure 5.9 (a) and (b) representing blog and Digg posts respectively, these take the form of an actual title and document snippet from the original document, while for Twitter (Figure 5.9 c) the



Web Search Query: lionel richie

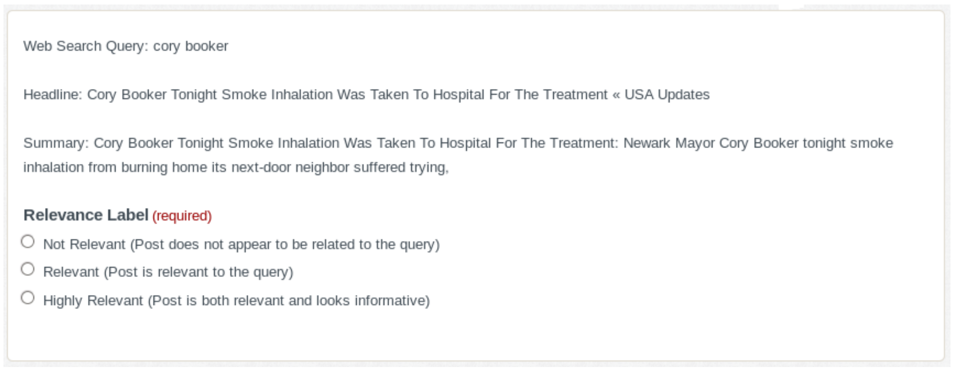
Blog post: Lionel messi News, Videos, Reviews and Gossip - Gizmodo

Summary: Behind the World's Lightest Soccer Shoe. Get our top stories. The World Cup's just around the corner, and that means the shoe companies are now battling over who's stuff is more innovative. In a feat of high-tech engineering, Adidas has made the lightest soccer shoe ever.

Relevance Label (required)

- ○ Not Relevant (Post does not appear to be related to the query)
- ○ Relevant (Post is relevant to the query)
- ○ Highly Relevant (Post is both relevant and looks informative)

(a) Blog Post



Web Search Query: cory booker

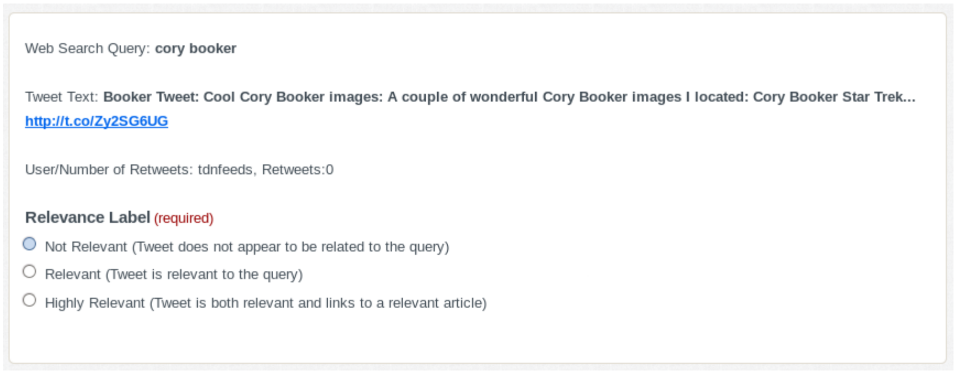
Headline: Cory Booker Tonight Smoke Inhalation Was Taken To Hospital For The Treatment « USA Updates

Summary: Cory Booker Tonight Smoke Inhalation Was Taken To Hospital For The Treatment: Newark Mayor Cory Booker tonight smoke inhalation from burning home its next-door neighbor suffered trying,

Relevance Label (required)

- ○ Not Relevant (Post does not appear to be related to the query)
- ○ Relevant (Post is relevant to the query)
- ○ Highly Relevant (Post is both relevant and looks informative)

(b) Digg



Web Search Query: **cory booker**

Tweet Text: **Booker Tweet: Cool Cory Booker images: A couple of wonderful Cory Booker images I located: Cory Booker Star Trek... http://t.co/Zy2SG6UG**

User/Number of Retweets: tdnfeeds, Retweets:0

Relevance Label (required)

- [✓ Not Relevant (Tweet does not appear to be related to the query)]
- ○ Relevant (Tweet is relevant to the query)
- ○ Highly Relevant (Tweet is both relevant and links to a relevant article)

(c) Twitter

Figure 5.9: The basic interface with which our workers label each query.

tweet text is displayed in the ‘title’ slot and the username and the number of retweets is displayed below it. Underneath the document representation, a common evaluation block is displayed enabling the user to record their assessment for that document and query.

5.7.3 Validation of Worker Assessments

As with previous crowdsourcing studies, we use gold judgement validation to identify workers who produce random or low quality assessments. In particular, for each of the five types of document that we rank, we create randomly select a small subset to manually assess, forming our gold judgement set. In particular, for newswire articles, we created 77 gold documents (4.9% of the total), while for blog posts we created 70 (3.7% of total). For the Digg source we created 88 (4.5% of the total), while we assessed 91 for Twitter (4.6% of the total). Finally, for Wikipedia, we created 50 gold documents (5.1% of the total). These ‘gold’ documents are interspersed with the non-gold documents that each worker assesses. Workers that fail more than 20% of the gold documents are rejected from the evaluation unpaid.

5.7.4 Crowdsourcing Configuration

To crowdsource our relevance labels, we used the CrowdFlower microtask crowdsourcing platform on top of Amazon’s Mechanical Turk (MTurk). We have the documents from each of the five sources assessed separately. Documents from each source are grouped into sets of ten (including a single gold document) forming an MTurk HIT. We pay workers \$0.01 (US dollars) per assessment made, hence \$0.10 for each HIT. Each document is assessed by three workers. The total cost for crowdsourcing all 24,570 assignments (8190 documents * 3 workers) was \$446.53. We do not restrict workers by region, however, we do have three individual workers attempt each. The final assessments produced are the majority vote across the three assessments.

5.7.5 Assessment Quality

We evaluate the quality of the assessments produced by the workers in terms of inter-worker agreement across the three workers that attempted each HIT. A high level of agreement indicates that the final labels are of good quality (see Section 5.3.2). In this case, we report the agreement measure that CrowdFlower provides, namely *observed agreement*. In this case, the probability of random agreement is the probability of all three workers selecting the same label by chance, i.e. 33%. Table 5.15 reports on a per source basis the number of assessments accrued for each in addition to the average worker agreement across all 199 queries for each source.

Corpus	# Documents	# Gold	# Assessments	# Agreement
$BlekkoNewsSnippets_{Apr2012}$	1,553	77	5,245	82.62%
$BlekkoBlogSnippets_{Apr2012}$	1,857	70	6,432	71.53%
$DiggSnippets_{Apr2012}$	1,888	88	6,027	77.29%
$Tweets_{Apr2012}$	1,990	91	7,722	73.43%
$WikiUpdates_{Apr2012}$	902	50	3,695	77.39%
Total	8,190	376	29,121	76.07%

Table 5.15: Statistics of the document relevance assessments produced by MTurk workers.

From Table 5.15, we observe that for all five sources, agreement is higher than 71% across the three relevancy labels. This is a high level of agreement that indicates that the resultant assessments are of good quality in comparison to random levels of agreement. The lowest agreement reported is 71.53% for the blog corpus, while the highest agreement was when assessing newswire articles with 82.62% agreement. The higher level of agreement when assessing newswire articles in comparison to tweets or blogs indicates that news articles are easier to assess as relevant or not to a query. Overall observed agreement across sources is 76.07%. Since this is much higher than random agreement, we conclude that the resultant relevance assessments are of sufficient quality to use later in Chapter 9.

5.8 Conclusions

In this chapter, we described ten different datasets from time periods between 2006 and 2012 that we use in our subsequent experimental chapters to evaluate whether the use of user-generated content can aid in satisfying news-related queries submitted to universal Web search engines. In Section 5.2, we listed each of the ten datasets that we use later (see Table 5.2) divided into the four components of our proposed news search framework.

Out of the ten datasets, four contain document assessments that we developed using the medium of crowdsourcing. In Section 5.3, we defined what crowdsourcing is and why it is a valuable resource for relevance assessment and detailed the validation strategies the we employ to improve the quality of the assessments produced. Agreement measures that were used to evaluate the quality of those assessments were also described.

In Sections 5.4, 5.5, 5.6 and 5.7, we described how we used crowdsourcing to generate assessments for each of the four datasets, totalling over 60,000 individual assessments and a total cost of $1,442.93 (US dollars). In particular, in each section, we described in detail how we developed effective new interfaces for crowdsourcing assessments and then validated the resulting assessments in real-time, enabling the rejection of poor quality work. Furthermore, we closed each of these four sections by evaluating the quality of the final assessments produced, showing through high levels of inter-worker agreement or accuracy against a ground truth, that the resultant assessments were of sufficiently good quality.

In the next chapter of thesis, we examine the Top Events Identification (TEI) component of our news search framework. We use the five datasets that we described in Section 5.2.1 (see Table 5.3), including the $BlogTrack_{2011}^{TopNews-Phase1}$ dataset that contains newswire article assessments that we crowdsourced (see Section 5.4), to evaluate whether important events can be identified accurately using live streams of blog posts or tweets.

Chapter 6

Identifying Top News

6.1 Introduction

Previously, in Chapter 4, we proposed a news search framework to describe the process of news search within a universal Web search engine. Within that framework, we identified four components, each responsible for a different task. In this chapter, we investigate the first of these components, namely Top News Identification (see Section 4.5). The aim of this component is to estimate what the most important news stories of the moment are in a fully automatic and real-time manner. This can be seen as a ranking task, whereby a set of news stories are ranked by their current importance. The Top News Identification component is unusual within our news search framework, in that it is not driven by a user-query. Instead, the component maintains an up-to-date ranking of news stories, ranked by their current importance, such that it can be used by other components within the framework. In particular, the rankings produced here are subsequently used within the News Query Classification and News Content Integration components that we examine later in Chapters 7 and 9, respectively.

However, 'current importance' is challenging to measure, as there are many factors that can influence it, e.g. story prominence, significance and proximity to the reader (Wavelength Media, 2009). As per our thesis statement (see Section 1.3), we propose to leverage real-time discussion in user-generated content sources to estimate the importance of individual news stories. Indeed, there are two key advantages to using user-generated content sources to measure news story importance. Firstly, the fast paced nature of some user-generated content sources, e.g Twitter, may enable breaking news stories to be promoted early in their lifetime, as discussion builds in social media. Secondly, by using the ebb and flow of user discussion, we can help avoid editorial bias observed in newspaper reporting (Sear, 2012).

In this chapter, we examine how to effectively rank news stories by their current importance or newsworthiness using discussion about those stories expressed in user-generated content and evaluate how these rankings compare to those produced by newspaper editors. In particular, we propose two types of approach to news story ranking – unsupervised and learning to rank – and then evaluate them on blog and tweet datasets. It is of note that although we motivate this work within the context of a universal search engine, the techniques proposed and evaluated here are directly applicable to news aggregator services and indeed e-newspapers that want to automatically rank their articles by importance (see Section 4.5). The remainder of this chapter is structured as follows.

- Section 6.2 details the first of our two types of approach to news story ranking, specifically unsupervised approaches. We propose an adaptation of the Voting Model to measure the recent discussion within user-generated content sources and enhancements to that model.
- In Section 6.3, we describe the second of our two types of approach to news story ranking, i.e. learning to rank news stories. In particular, we propose a new news story ranking framework that uses training data to build a news story ranking model combining multiple story ranking approaches with temporal evidence to better estimate the importance of a news story from a user-generated content stream.
- Section 6.4 defines our methodology for evaluating each of the two types of news story ranking approach described in the prior two sections over blog and tweet datasets (see Section 5.2.1).
- In Section 6.5, we describe the structure of our evaluation and enumerate the research questions investigated in the following two evaluation sections (Section 6.6 and 6.7).
- Section 6.6, the first of the two evaluation sections, evaluates the effectiveness of the news story ranking approaches proposed in Sections 6.2 and 6.3 on a stream of blog data. In particular, we use the TREC 2009 and TREC 2010 top news stories identification task datasets (see Section 5.2.1).
- Section 6.7 evaluates the effectiveness of our news story ranking approaches on real-time tweet data. In particular, we use a live stream of tweets from late 2011 and early 2012, in addition to news story rankings from newswire providers (see Section 5.2.1). We evaluate whether our story ranking approaches are effective at estimating the importance of news stories when leveraging large volumes of short tweets.
- In Section 6.8, we summarise the findings of this chapter.

6.2 Unlearned Story Ranking Approaches

The aim of this chapter is to investigate approaches to rank a set of pre-provided news stories by their importance, where importance is estimated using discussion in user-generated content streams (see Section 4.5). Stories are represented as newspaper articles from one or more news providers. One approach to estimate the importance of a news story is by measuring the volume and intensity of discussion about that story. The more a story is discussed, the more important it is considered to be. For example, in the case of the blogosphere, this can be estimated using the number of blogs or individual blog posts that discuss a news story. The first approaches to automatic ranking of news stories using the blogosphere were developed for the TREC 2009 and 2010 Blog track top news stories identification (TNSI) task (Ounis *et al.*, 2010). More information about the TREC Blog track TNSI task and approaches to tackle it (including baselines that we compare to in this chapter) can be found in Section 3.2.3.

In this section, we propose our own approaches for automatic news story ranking derived from voting theory (Macdonald, 2009). In particular, in Section 6.2.1, we propose our baseline approach, which we refer to as Votes. We then propose four enhancements to our approach in Sections 6.2.2 to 6.2.5, which take into account the degree of relatedness between posts and each news story (RWA); the length of the news story representation (RWAN); the distribution of related posts to each news story over time (GaussBoost), and the maximum importance observed (MaxBurst), respectively.

6.2.1 Voting for Important Stories (Votes)

To achieve automatic news story ranking, we propose to model a story's newsworthiness as a voting process (Macdonald & Ounis, 2009). The intuition underpinning our approach is that a blog post related to a news story can be seen as a vote for that story to be important at the time the blog post was published. Hence, by measuring the *volume* of posts sharing content with a given article around the time of ranking, i.e. the votes, we can estimate the importance of the story.

As mentioned earlier, a news story to rank for is described by a news article representation, denoted a, e.g. a headline. To measure the response to such a news article at and before a point in time t, we count the number of related posts to that article that were published during the period of $t - r \rightarrow t$. r is a window size, e.g 1 day, that represents recent posts. To operationalise this, an information retrieval (IR) system first selects a fixed number[1] of posts $R(a, S_{t-w\rightarrow t})$ which are topically related to news article a during the period of $t - w \rightarrow t$. w is a time period greater than r and represents the background of posts that are available. Let $R(a, S_{t-r\rightarrow t}) \subset R(a, S_{t-w\rightarrow t})$ denote the subset of posts

[1]In our later experiments we retrieve 1000 posts per article. An additional analysis of the ranking depth can be found in Appendix A.1.2

	d_1	d_2	d_3
a_1	4	4	2
a_2	1	8	1

Table 6.1: A sample assignment of votes for two articles (a_1 and a_2) over 3 days (d_1 to d_3).

in $R(a, S_{t-w\rightarrow t})$ that were published during the smaller time window $t - r \rightarrow t$. We consider that $R(a, S_{t-w\rightarrow t})$ can be seen as a set of votes for article a as important during for the period $t - w \rightarrow t$ (the background), while $R(a, S_{t-r\rightarrow t})$ can be seen as a set of votes for article a as important during the more recent period $t - r \rightarrow t$. By counting the number of votes for article a in the most recent subset (i.e. $|R(a, S_{t-r\rightarrow t})|$) we can estimate the article's importance for the period r. Consequently, using this voting-like approach (Macdonald, 2009), which we refer to as *Votes*, the score for each article a is calculated as:

$$score(a, S, t, w, r) = |R(a, S_{t-r\rightarrow t}) \subset R(a, S_{t-w\rightarrow t})| \qquad (6.1)$$

where a is the newswire article to rank for, S is the source to rank documents from, t is the ranking time, w is the size of the time window from which to rank and r is size of the smaller window considered recent. To build the final ranking of articles for time t, denoted I (see Section 4.4), we compare the number of votes for all articles $a \in A$, i.e. we rank by $score(a, S, t, w, r)$. We refer to this story ranking approach throughout this thesis as *Votes*. Note that this is a three-stage process. Firstly, for each news article a, we retrieve related posts from the background window w in stream S. Next, we find the subset of those posts that we consider recent, as defined by the period $t - r \rightarrow t$, and count them to form a score. Lastly, we rank all articles $a \in A$ by the number of votes (score) they received.

To illustrate our approach, we give a short example of Votes for two news articles (a_1 and a_2) and 3 days (d_1 to d_3), i.e. $r = 1day$ and three values for t are defined corresponding to three days. For each article, a ranking of the top 10 posts is analysed, and then votes are assigned to the various days. Table 6.1 shows, for each article (row), the number of votes for each of the 3 days. From this example, we can see that a_1 received 4 votes on days d_1 and d_2, in contrast to only 2 votes on d_3. Article a_2 obtained 8 votes on d_2 but only 1 on days d_1 and d_3. Hence, to find the most important news article on d_1 (first column), a_1 would be ranked higher than a_2, since a_1 received more votes. Similarly, for d_2 (second column), a_2 would be the most important news story.

6.2.2 Relevance Weighted Aggregation (RWA)

Our Votes approach described above, considers all posts returned in $R(a, S_{t-r\rightarrow t})$ to be equally relevant to the news article a. However, in a fast paced real-time setting, where a news article has just broken, there may be few relevant posts to return. In cases where there is little or no related discussion (so far), the majority of posts returned in $R(a, S_{t-r\rightarrow t})$ will be irrelevant to the news story and retrieved only due to chance matching of terms from that story. In these cases, under Votes, the score for a story can be highly misleading, as it assumes that the posts in $R(a, S_{t-r\rightarrow t})$ discuss the news story.

To counteract this, we propose an enhanced variant of Votes, which incorporates the relevance of a post to the news story. In particular, we propose to weight each post by the degree to which it is related to the story. In this way, for cases where there are few relevant posts, and hence the story should be considered unimportant due to a lack of related discussion, the low retrieval scores for non-relevant posts assure that the story will receive an overall low score.

In particular, the weight for each post becomes equal to its relevance score, while the newsworthiness for the story becomes the sum of the scores of the posts returned. We refer to this enhanced story ranking approach as Relevance Weighted Aggregation (RWA), which is defined as follows:

$$score(a, S, t, w, r) = \frac{\sum_{p \in R(a,S_{t-r\rightarrow t}) \subset R(a,S_{t-w\rightarrow t})} score(a, p)}{|R(a, S_{t-w\rightarrow t})|} \tag{6.2}$$

where a is a story, $R(a, S_{t-r\rightarrow t})$ is the set of posts returned for a and $score(a, p)$ is the retrieval score (e.g. using a document weighting model – see Section 2.3) for post $p \in R(a, S_{t-r\rightarrow t}) \subset R(a, S_{t-w\rightarrow t})$. We hypothesise that this aggregation approach should be more effective than Votes, because it incorporates not only the volume of discussion about a news story, but also the likelihood that the post is about article a. This should avoid overemphasising the importance of a news story in cases where many non-relevant or weakly relevant documents are returned in $R(a, S_{t-r\rightarrow t})$.

6.2.3 Story Length Normalisation (RWAN)

Next, we propose an enhancement to the RWA, denoted $RWAN$, that takes into account the length of the story representation. The intuition is that the length of each news story should act as a normalising factor on the score for each news story, as when comparing across news stories, the longer news story representations will naturally receive higher scores under RWA as they are based upon the SUM of the retrieval scores of each individual document. This results because longer news story representations typically lead to larger document (post) scores, because they in turn are dependant upon the SUM of the scores for each individual term in the document retrieved. The aim is to avoid favouring stories with longer news story representations. In particular, we add a story length component to RWA to normalise out the differing lengths of individual headlines as follows:

$$score(a, S, t, w, r) = \frac{1}{|a|} \cdot \frac{\sum_{p \in R(a,S_{t-r\rightarrow t}) \subset R(a,S_{t-w\rightarrow t})} score(a, p)}{|R(a, S_{t-w\rightarrow t})|} \tag{6.3}$$

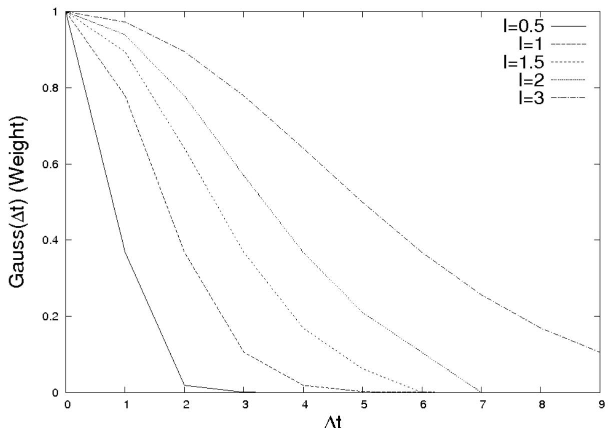


Figure 6.1: Example of Gaussian curves with varying values of l (the Gaussian curve width). Note that for illustration clarity, weights have been normalised so that $\Delta t = 0$ will always be assigned a weight of 1.

where $\frac{1}{|a|}$ is the added normalisation component and $|a|$ is the number of terms in the newswire article representation a.

6.2.4 Temporal Promotion (GaussBoost)

Thus far, we have made use of evidence from only the time period preceding t defined by r. However, historical evidence may also help to improve the accuracy of Votes, RWA or RWAN. In particular, news stories may be discussed beforehand for predictable events, e.g. speculation about election results, and/or discussed afterwards for long running, controversial or important unpredictable stories, e.g. the aftermath of a terrorist bombing. By taking this evidence into account, we may be able to identify those stories that maintain their interest over time, and as such can be deemed more important. Kleinberg (2002) suggested that bursts in term distributions could last for a period of time. Hence, in the following, we define an alternative technique for calculating $score(a, S, t, w, r)$, which leverages the *temporal distribution* of each article a over time. This technique additionally accumulates vote evidence from the period preceding $t - r$, to 'boost' the score of articles which retain their importance over time.

In our proposed temporal distribution boosting technique, denoted $GaussBoost$, we consider the background related document set $R(a, S_{t-w \rightarrow t})$ to be comprised of n equal parts covering periods of size r, starting at time t. These parts are referred to as temporal units (of size r). Votes or RWA work exclusively on the most recent of these temporal units before t. In contrast, $GaussBoost$ weights documents (posts) based on the time elapsed from t, using a Gaussian curve to define the magnitude of emphasis. In this way, we state a preference for stories that were important during periods close to t, rather than stories which peaked some time before t:

$$score(a,S,t,w,r) = \sum_{0<\Delta t<\frac{w}{r}} Gauss(\Delta t) \cdot |R(a,S_{t-r*(\Delta t+1)\rightarrow t-r*\Delta t}) \subset R(a,S_{t-w\rightarrow t})| \quad (6.4)$$

Δt is the number of temporal units since t. $Gauss(\Delta t)$ is the Gaussian curve value for a difference of temporal units Δt, as given by:

$$Gauss(\Delta t) = \frac{1}{l.\sqrt{2\pi}} \cdot exp\frac{-(\Delta t)^2}{(2l)^2} \quad (6.5)$$

where the parameter l defines the width of the Gaussian curve. A small l will emphasise stories that were important during temporal units very close to t, while a larger l will take into account stories highly scored for temporal units more distant to t. In general, the larger the distance from t (Δt), the lower the weight assigned to the evidence from the corresponding temporal unit. Figure 6.1 shows five sample Gaussian curves with different values for l. As we can see, when we increase l, evidence from days further from the t are taken into account. In particular, if $l = 0.5$ then only stories from the first temporal unit after t receive additional importance, while a l value of 3 promotes stories from the 9 prior temporal units in a diminishing fashion.

The advantage of this approach over Votes and RWA is two-fold. Firstly, it allows us to account for evidence preceding the period $t - r$. Secondly, we can control the weight placed on temporal units other than $t - r \rightarrow t$, thereby avoiding over-emphasising documents posted on days which are unlikely to be useful. However, we are assuming that the Gaussian distribution is a good model of the way the usefulness of the evidence will diminish, which may not be the case for all stories.

It is important to note that $GaussBoost$ is an enhancement to Votes, RWA and RWAN approaches. Indeed, in Equation 6.4, $|R(a,S_{t-r*(\Delta t+1)\rightarrow t-r*\Delta t}) \subset R(a,S_{t-w\rightarrow t})|$ is simply Votes, where the Δt'th temporal unit is scored rather than the most recent one. These enhancements can be similarly applied to RWA or RWAN by replacing $|R(a,S_{t-r*(\Delta t+1)\rightarrow t-r*\Delta t}|$ with the RWA/RWAN scoring functions (Equations 6.2 and 6.3), corrected for the Δt'th temporal unit. For example, RWA and $GaussBoost$ together can be expressed as:

$$score(a,S,t,w,r) = \sum_{0<\Delta t<\frac{w}{r}} Gauss(\Delta t) \cdot \frac{\sum_{p\in R(a,S_{t-r*(\Delta t+1)\rightarrow t-r*\Delta t})\subset R(a,S_{t-w\rightarrow t})} score(a,p)}{|R(a,S_{t-w\rightarrow t})|} \quad (6.6)$$

In our later experiments, we use the $GaussBoost$ subscript to denote runs where this enhancement has been added to either Votes, RWA or RWAN. For example, RWA and $GaussBoost$ is denoted RWA$_{GaussBoost}$.

6.2.5 Maximum Burst Magnitude (MaxBurst)

In the previous sub-section, we introduced a temporal promotion technique, referred to as $GaussBoost$, that takes into account discussion preceding the period $t \rightarrow t - r$. This approach is based upon the assumption that if a news story is important, then there will be a rise in discussion during or just before $t \rightarrow t - r$. However, in a real-time streaming setting, particularly for fast-paced user-generated streams like Twitter, discussion of news stories may follow short term bursty distributions, i.e. there is a sharp rise in discussion as the story breaks, which subsequently dies out. Over time, our approaches described above will promote and then demote news stories in-line with this bursty distribution, i.e. the ranking of stories will change. However, if the burst is short, then the time period for which the news story is promoted will also be short. For example, if discussion only lasts for 30 minutes in Twitter, then once that 30 minute window of discussion exits the period defined by $t \rightarrow t - r$, the news story will be deemed as unimportant from that point on. When ranking news stories, both old and new stories are ranked. It would be advantageous to maintain knowledge of the historical importance of a news story, such that its importance is not underestimated as soon as discussion dies out.

To this end, we propose a second enhancement for our Votes, RWA and RWAN approaches. In particular, instead of focusing on only the most recent period defined by $t \rightarrow t - r$, we instead focus on the time period where the news story was most discussed. To operationalise this, we use a similar formulation as $GaussBoost$, representing the background time window of size w to be comprised of fixed time periods (temporal units) of size r. $t \rightarrow t - r$ is the most recent of these periods. For each of our prior news story ranking approaches, we calculate the score for each temporal unit as per the approach specified, i.e. $t \rightarrow t - r$ is replaced by $t - r * (\Delta t + 1) \rightarrow t - r * \Delta t$ representing the Δtth temporal unit preceding t. The maximum of these scores is then considered to be the score for the news story. In effect, this approach measures the maximum importance observed for the news story over the time period covered by w, hence we refer to it as the maximum burst magnitude or $MaxBurst$. For example, $MaxBurst$ is calculated using RWA as follows:

$$score(a, S, t, w, r) = \max_{0<\Delta t<\frac{w}{r}} \frac{\sum_{p \in R(a, S_{t-r*(\Delta t+1) \rightarrow t-r*\Delta t}) \subset R(a, S_{t-w \rightarrow t})} score(a, p)}{|R(a, S_{t-w \rightarrow t})|} \tag{6.7}$$

where Δt is the number of temporal units since t and p is a single post retrieved from the temporal unit defined by $t - r * (\Delta t + 1) \rightarrow t - r * \Delta t$.

Of course, the disadvantage of $MaxBurst$ is that it does not take into account the age of each document retrieved for the story when estimating that story's importance, i.e. the score for a news story will always be the maximum importance observed for that story (within the background window considered). Until the burst of activity for a news story leaves the background time window of size

w, the score for a news story will be constant, unless a new larger burst of discussion is detected. To counteract this, we also combine $MaxBurst$ with $GaussBoost$, using the Gaussian curve value (see Equation 6.5) to reduce the score for stories the further from t the burst occurs. For example, $MaxBurst + GaussBoost$ using RWA is calculated as follows:

$$score(a, S, t, w, r) = \max_{0<\Delta t<\frac{w}{r}} Gauss(\Delta t) \cdot \frac{\sum_{p \in R(a, S_{t-r*(\Delta t+1) \rightarrow t-r*\Delta t}) \subset R(a, S_{t-w \rightarrow t})} score(a, p)}{|R(a, S_{t-w \rightarrow t})|} \tag{6.8}$$

where $Gauss(\Delta t)$ is the Gaussian curve value for the Δtth temporal unit defined in Equation 6.5. This approach is denoted RWA$_{GaussBoost+MaxBurst}$.

6.3 Learning To Rank Stories (LTRS)

In the previous section, we proposed a story ranking approach based upon voting theory and four extensions to it. All of these approaches define an untrained model via which stories can be ranked given some user-generated content stream. In contrast, in this section, we propose a new learned approach to news story ranking that leverages pre-provided training examples to learn a news story ranking model. In particular, we propose a new learning to rank framework comprised of four elements to generate features, these features are then combined via a learning to rank technique into a story ranking model. We refer to this framework as LTRS (Learning to Rank Stories).

Learning to rank techniques are machine learning algorithms that take as input a set of document features and learn a combination of those features within an information retrieval (IR) system (see Section 2.5.2). The aim is to find the combination of these features that results in the most effective document ranking. The advantages of a LTR approach in comparison to existing story ranking strategies are two-fold. Firstly, LTR provides a principled means for combining multiple sources of story ranking evidence as features. For example, in the previous section, we proposed both RWA that considers discussion just before the time of ranking t, and RWA$_{MaxBurst}$ that considers the period during which most of the discussion about that story happened. Using LTR, we can express both of these approaches as features and then combine them within a single news story ranking model. Secondly, as LTR is extensible, should a new possible feature become available, e.g. the number of clicks on a specific story, then this can be easily integrated to further increase story ranking effectiveness.

Traditional LTR techniques define features on the object that is to be ranked, i.e. the news story in this case. However, news story ranking is an aggregate ranking task, i.e. newsworthiness is defined in terms of a collection of related objects, i.e. posts relating to the story, rather than the news story itself. Inspired by prior work in the field of applying learning to rank techniques to aggregate problems (Macdonald & Ounis, 2011), we propose an novel approach that generates LTR features by combining different elements from multiple story ranking strategies. In particular, under LTRS, a single feature is comprised of four elements:

- **Story Representation:** Textual representation of each story to be ranked, e.g. its headline.
- **Document Ranking Approach:** Generates a ranking of documents from the stream using the story representation as a query.
- **Document Sub-Feature:** A numeric feature that we extract from each document to be ranked.
- **Aggregation Model:** Defines how to aggregate the document features extracted from each document together into a final score for each news story.

These elements represent a real-time story ranking strategy in a generic manner, i.e. a document ranking approach takes as input a textual representation of a story (story representation) to be ranked, retrieving a set of relevant posts about that story. Features about each of the posts retrieved (document sub-features) are then combined into a final score for the story using an aggregation model. For example, Xu *et al.* (2010)'s real-time approach (see Section 3.2.3) estimates a story's newsworthiness by summing the BM25 scores for each post published during the last 24 hours using the headline from a related article as a query. In this case, the story is the headline, the document weighting model BM25 is used to rank documents, the document feature is the retrieval score from BM25, and the aggregation model is a summation of the BM25 scores. Formally, a single feature is comprised of the four elements: a story representation a, a document ranker R, sub-feature to extract f and the aggregation model M. Hence, in simple terms, one feature can be expressed as follows:

$$feature(a, f, M) = M(f(R(a))) \tag{6.9}$$

where $R(a)$ retrieves the top documents for the story representation a, function f returns a score for each of the documents returned by $R(a)$ which are subsequently aggregated together into a final score for a using M. In line with the real-time nature of the task, story ranking is performed with respect to a specific point in time t and using a window of time w, like our untrained story rankers described in Section 6.2. Hence, each feature is expressed as:

$$feature(a, S, t, w, f, M) = M(f(R(a, S_{t-w \to t}))) \tag{6.10}$$

Figure 6.2 illustrates a example of feature extraction for a Twitter stream. 12 unique features are generated by combining two story representations (the news story headline and news story summary), two

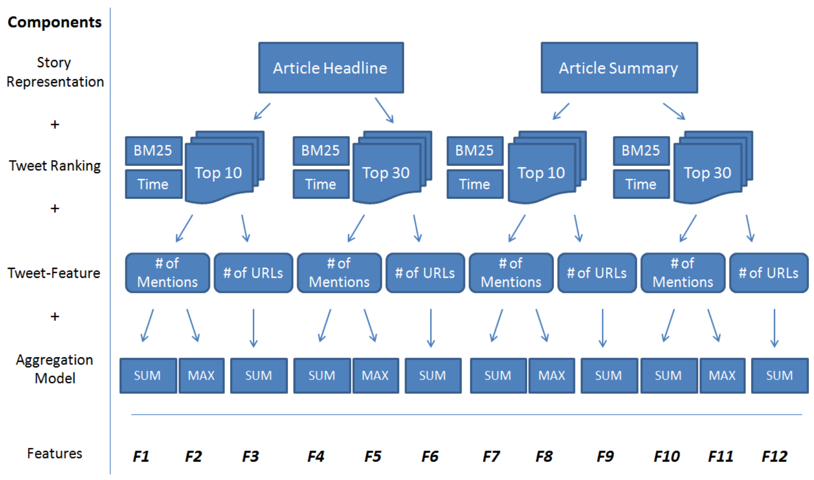


Figure 6.2: An example of 12 story ranking features generated from two story representations, two tweet rankers, two extracted tweet-features and two aggregation models.

document (tweet) rankers (BM25 retrieving 10 tweets and BM25 retrieving 30 tweets), two document (tweet) features (the number of mentions and URLs within each tweet) and two aggregation models (SUM the tweet features or return only the MAX value). For example, feature 6 (**F6**) is generated by using a news story article headline as a query to a tweet ranking system. This system uses the BM25 retrieval model to retrieve the top 30 tweets. From each of these 30 retrieved tweets, the number of URLs contained within each tweet is extracted to use as a feature, and then these features are aggregated via a summation to form a score for the story (a feature). As can be seen from Figure 6.2, many features are generated by varying the four elements of the LTRS framework. To further illustrate this using our own unsupervised approaches described in the previous section, consider RWA. It takes a news story representation a (Story Representation), retrieves posts using a document weighting model (see Section 2.3) to form a ranking $R(a, S_{t-w \rightarrow t})$ (Document Ranking Approach) and then combines the retrieval score (Document Sub-Feature) for each post made during the period $t - r \rightarrow t$ using a mathematical summation (Aggregation Model). We could, for example, change the underlying document ranking approach by changing the size of w, to generate other features. In the next section, we describe our experimental setup, including the features that we extract from this framework and that we use to learn a news story ranking model.

6.4 Experimental Methodology

We evaluate our news story ranking approaches in two distinct settings, each leveraging a different user-generated content stream. Firstly, we evaluate our news story ranking approaches within the context of the TREC Blog track 2009 and 2010 top news stories identification tasks, i.e. using a blog stream. The datasets associated with these tasks are the $BlogTrack_{2009}^{TopNews}$ and $BlogTrack_{2010}^{TopNews-Phase1}$, described in Section 5.2.1. Secondly, we evaluate our proposed ranking approaches on a real-time tweet stream provided by Twitter from two points in time and spanning three datasets. The datasets associated with this stream are the $Twitter_{Dec2011}^{TopNews-NYT}$, $Twitter_{Jan2012}^{TopNews-NYT}$ and $Twitter_{Jan2012}^{TopNews-Reuters}$ datasets. Notably due to the disparate time-frames for which user-generated content streams are available – as discussed in Section 5.1 – the blog stream covers the year of 2008, while the Twitter stream is from the period from late 2011 to early 2012. For brevity, in the remainder of this chapter we refer to the $BlogTrack_{2009}^{TopNews}$ and $BlogTrack_{2010}^{TopNews-Phase1}$ datasets as Blogs$_{09}$ and Blogs$_{10}$, respectively. Similarly, we denote the three Twitter datasets as Twitter$_{11-NYT}$, Twitter$_{12-NYT}$ and Twitter$_{12-TR}$, respectively. In Section 6.4.1, we describe our evaluation methodology on the Blog track datasets, while Section 6.4.2 details our methodology for evaluation on the Twitter datasets.

6.4.1 Blog Stream Methodology

The Blogs$_{09}$ dataset spans the period of January 2008 to February 2009. It is comprised of the TREC *Blogs08* blog post corpus (Macdonald & Ounis, 2006*b*) and the $NYT08$ news story corpus, which span the same period. With the dataset, news story importance assessments for 50 'topic days' are provided. Each topic day refers to exactly one day from the year long corpus. For these topic days, news stories from the $NYT08$ corpus were manually judged as important or not for that day by human assessors. For evaluation, we have each of our news story ranking approaches assess the importance of each news story published on the 50 topic days. As this ranking task is defined in terms of days, the time of ranking t is considered to be 23:59 on each topic day. For our unsupervised approaches and story ranking features that define a small window describing the period to rank for r, r equals 1 day. The story rankings produced by each of our approaches are evaluated against the stories that the assessors marked as important. Notably, as 2009 was the first year that top news stories identification was run at TREC, the task was formulated as a retrospective ranking task. Here systems were allowed to use evidence from after the time of ranking. In effect, this means that the original TREC runs on the Blogs$_{09}$ dataset were not truly real-time in nature. However, we are interested in using user-generated content sources for their real-time nature. Hence, in this work we maintain our real-time constraints for runs on this dataset, i.e. we do not make use of any future evidence, unlike what the TREC systems could do.

The Blogs$_{10}$ dataset similarly spans the period of January 2008 to February 2009. The same *Blogs08* blog post corpus (Macdonald & Ounis, 2006*b*) is used as the user-generated content stream, however a different news story corpus from the Reuters news agency, denoted $TRC2$ (Leidner, 2010) was used. Again, 'topic days' with associated assessments for stories published on those days are provided in terms of their newsworthiness, against which we evaluate the rankings produced by our approaches. The ranking task is once again defined in terms of days, hence the time of ranking t is considered to be 23:59 on each topic day. However, TREC 2010 task followed a strict real-time setting, so approaches on the Blogs$_{10}$ dataset can only use posts from before the time of ranking t.

Due to slight differences in setting between the TREC 2009/2010 task formulations, we make the following changes to create a consistent setting and make cross-corpus training possible. Firstly, the TREC 2009 task ($NYT08$ topics) considered that stories both after and before each topic day might still be relevant due to differences in time-zone, which the 2010 task ($TRC2$ topics) did not. We follow the TREC 2010 setting and only rank the stories published on each topic day. Secondly, the 2010 task introduced category classification of articles, i.e. each article was judged as to the degree to which it is important on the topic day with regard to one of five news categories. Importantly, these categories can introduce a confounding variable into the evaluation, as even a perfect article ranking system will be heavily penalised should it use a poor classifier. Hence, in this chapter, we report news story ranking performance only and leave category classification to future work.

Of note is that the $TRC2$ news corpus provides both an article headline for each story, as well as the full article content, whilst the $NYT08$ corpus provides only the article headline. To make these corpora comparable, we independently crawled the missing article content for the $NYT08$ corpus, cleaning the resulting text with the BoilerPipe article extractor (see Section 2.2.6).

We use the Terrier information retrieval platform (Ounis, Amati, Plachouras, He, Macdonald & Lioma, 2006) to index the Blogs08 corpus of blog posts, removing standard stopwords, and applying Porter's English stemmer. To generate the ranking of blog posts with respect to a news article $R(a, S_{t-w \rightarrow t})$, we use a w setting (background window size) of 10 days, i.e. 10 days worth of blog posts prior to t. For our unsupervised approaches, we use the DPH ranking model from the Divergence from Randomness framework (Amati *et al.*, 2007) (see Section 2.3.3) to rank blog posts for a news story a. In contrast, for our learned approach, to show its generality, we use two effective weighting models, namely: the probabilistic BM25 (Robertson *et al.*, 1992) (Section 2.3.2); and the aforementioned DPH model. To make our results comparable with other systems participating in the TREC Blog track 2009 top news stories identification task, we use the default parameters for the weighting models employed, as no training data was available for Blogs08 at the time the task was run. In particular, we use the default parameters for BM25 of $k1 = 1.2, k_3 = 1000, b = 0.75$ (Robertson *et al.*, 1994). DPH on the other-hand, is a 'parameter-free' model, where all parameter values are derived from the collection statistics.

Notably, our story ranking approaches have a variety of parameters that may effect the story ranking effectiveness. Unless otherwise stated, for the following experiments, the default size of $|R(a, S_{t-w \to t})|$ (ranking depth or $|R()|$) is 1000, based upon recommendations by Macdonald (2009). The default blog representation is a blog post, and the default story representation (a) is the headline of the article for each story. In Appendix A can be found a detailed analysis of the effect that these parameters have on news story ranking[1]. For a state-of-the-art baseline, we compare our story ranking approaches to the best systems participating in TREC 2009 and 2010. For comparison to various approaches, statistical significance is measured using the t-test at p<0.01 and p<0.05. A single ▲ or ▼ indicates a statistically significant increase or decrease at p<0.05, while two such symbols indicate significance at p<0.01.

The LTRS framework described previously, uses four components to generate story ranking features, namely: a story representation, document ranking approach, document sub-feature and an aggregation model. We sampled a subset of the possible components that we considered a-priori might be effective for news story ranking and combined them to create 160 news story ranking features. Table 6.2 lists the instances of each of the four components that are used for feature extraction on the blog stream in our later experiments. Importantly, not all possible components are used here, e.g. we do not make features leveraging the GaussBoost extension to Votes. Our reasoning is two-fold. Firstly, not all components are applicable for the day-centric setting of the TNSI task, e.g. $MaxBurst$. Secondly, the number of possible features is multiplicative with the number of components, hence the feature space increases very quickly with the number of components, increasing training time. To train a learned model using these features, we experiment with two different training regimes, namely *Cross-Corpus* and *Per-Corpus*. In particular, under Cross-Corpus training, we train using the topics from one corpus (either $NYT08$ or $TRC2$) and then test upon the topics from the other corpus and vice-versa. Under Per-Corpus training, we train and test on the same topic set using a 5-fold cross validation. We train our models using the automatic feature selection (AFS) algorithm (see Section 2.5.2).

[1]Author Note: We do not include the analysis in this chapter as they do not impact the conclusions that we draw, but are provided later for completeness.

Component	Name	Description
Story Representation	Headline	The story headline.
	QE_Blogs08	The headline expanded using the Blogs06 blog post corpus (Macdonald & Ounis, 2006*b*).
	QE_NYT06	The headline expanded using 2000 news articles from the New York Times during May 2006.
	QE_TRC2	The headline expanded using 13 days of news stories from the TRC2 corpus but before the start of Blogs08 (Leidner, 2010).
	Content	The article content.
	Entities	Named entities from the article content identified by a Wikipedia-based dictionary (Santos, Macdonald & Ounis, 2010*b*).
	Noun-Phrases	Noun Phrases extracted from the article content (Schmid, 1995).
	Summary	Story summary generated using part-of-speech tagged article content (Lioma *et al.*, 2006).
Document Ranking Approach	BM25$_{1000}$	The BM25 (Robertson *et al.*, 1992) document ranking model retrieving 1000 blog posts. r is varied using values from 1 day to 10 days.
	DPH$_{1000}$	The DPH (Amati *et al.*, 2007) document ranking model retrieving 1000 blog posts. r is varied using values from 1 day to 10 days.
Document Sub-Features	Relevance	The retrieval score for the blog post.
Aggregation Model	BM25$_{SUM}$	Aggregated relevance-based story ranking model (Xu *et al.*, 2010).
	Votes	Voting-based story ranking model (McCreadie *et al.*, 2010*c*).
	$RWA^{C6.2.2}$	Voting-based story ranking model (McCreadie *et al.*, 2010*c*).

Table 6.2: Blog stream LTRS feature instances used.

6.4.2 Tweet Stream Methodology

We also evaluate our story ranking approaches on three different tweet datasets, namely: Twitter$_{11-NYT}$; Twitter$_{12-NYT}$; and Twitter$_{12-TR}$ (see Section 5.2.1). In particular, these datasets contain tweet corpora that cover the time periods of the 17th of December 2011 to the 31st of December 2011 and from January the 5th 2012 to January the 12th 2012.

To evaluate the effectiveness of our proposed approaches at identifying the top stories for a point in time, we compare against the rankings observed on the website homepages of major e-newspapers. Note that this differs from our blog stream evaluation, where human assessors were used to judge the importance of each news story. Instead, we use the story ranking from a news website as the ground truth, on the assumption that the editor of the e-newspaper knows what the important stories of the moment are. In particular, in parallel with the creation of the tweet corpora described above, we also downloaded the homepages of two major news providers, namely the New York Times and Reuters each hour. Homepages were only downloaded during the latter half of each tweet corpus. i.e. we only evaluate on the latter half. This is to allow for a reasonable background of tweets to be collected, avoiding issues regarding unstable background term statistics when few tweets have yet been seen. Duplicate story rankings, i.e. when the homepage has not changed since the last download, are removed. From each homepage download, we extract the set of current news headlines for our system to rank and also the ground truth ranking of stories against which we will compare. We evaluate story ranking effectiveness in terms of Normalised Discounted Accumulative Gain (NDCG) (Järvelin & Kekäläinen, 2002). In this case, the importance score assigned to each newswire headline is used as the relevance label. In this way, NDCG measures the ability of our approaches to promote higher ranked stories. We evaluate the story ranking as a whole ($NDCG$) and where our approach correctly identifies the top story in rank 1 ($Success@1$). Notably, unlike our blog stream evaluation, the ranking task is defined in terms of time points (to the nearest second) rather than days, hence each times at which we rank t is the moment that we crawled the newspaper homepages.

We use Terrier (Ounis, Amati, Plachouras, He, Macdonald & Lioma, 2006) to index the tweet corpora, removing stopwords, and applying Porter's English stemmer. To generate the ranking of tweets with respect to a news article $R(a, S_{t-w\rightarrow t})$, we use the DFReeKLIM (Amati *et al.*, 2011) document weighting model specifically designed for ranking short texts. Further information regarding DFReeKLIM can be found in Section 2.3.3.

Note that our tweet datasets are not TREC derived, but were rather built by ourselves specifically for this task, as detailed in Section 5.2.1. Hence, we cannot compare to other systems on these datasets. Indeed, to the best of our knowledge, news story ranking using tweets has never been investigated before. Instead, we use our Votes approach as a baseline and examine how the rankings produced by our approach differ from those observed on the source newspaper websites. Furthermore, as a sanity check, we compare our approach to the mean NDCG performance of ten random rankings of news stories, denoted $Random$.

LTRS uses four components to generate story ranking features, namely: a story representation, document ranking approach, document sub-feature and an aggregation model. Notably, the instances of each component used for feature generation differ between the two blog and three tweet datasets upon which we evaluate. The reason for this is that tweets differ markedly from blogs, hence different features need to be selected. For example, unlike for blogs, tweets have metadata like the number of retweets, that act as different document sub-features. As a result, we use a slightly larger feature set comprised of 177 tweet features. Table 6.3 lists the instances of each component used to extract features from the tweet stream. To train the weights for each of these features, we train on the Twitter$_{11-NYT}$ dataset and test on both the Twitter$_{12-NYT}$; and Twitter$_{12-TR}$ datasets. One fifth of the training dataset was reserved for validation. As before, we train our models using the automatic feature selection (AFS) algorithm (see Section 2.5.2).

6.5 Research Questions

Our overall aim is to determine whether user-generated content sources are useful for enriching the news vertical aspects of a Web search engine. Within the context of the Top Events Identification component evaluated in this section, we investigate whether news story ranking task can be tackled effectively by using discussion from user-generated content. To test this hypothesis, we developed both unsupervised and a learned story ranking approaches that leverage user-generated content (see

Component	Instance	Description
Story Representation	Headline	The article headline of each news story.
	PRF(10)	Headline expanded using the Bo1 (Amati., 2003) query expansion model with 10 terms from the top 10 tweets
	PRF(30)	Headline expanded using the Bo1 (Amati., 2003) query expansion model with 10 terms from the top 30 tweets
Document Ranking Approach	DFR_{10}	A DFR tweet ranking model (Amati *et al.*, 2011) retrieving 10 tweets.
	DFR_{30}	A DFR tweet ranking model (Amati *et al.*, 2011) retrieving 30 tweets.
	DFR_{1000}	A DFR tweet ranking model (Amati *et al.*, 2011) retrieving 1000 tweets.
Document Sub-Features	Relevance	The retrieval score for the tweet.
	URL	The number of URLs contained within the tweet.
	Mentions	The number of Mentions contained within the tweet.
	HashTags	The number of Hashtags contained within the tweet.
	Reach	The number of followers the tweet author has.
	ListReach	The number of times tweet author has been listed.
	Activity	The number of statuses the tweet author has made.
	Prominence	The number of times the tweet has been retweeted.
Aggregation Models	Votes	Counts the number of tweets both retrieved and that were posted within the last hour. (McCreadie *et al.*, 2010*c*)
	$RWA^{C6.2.2}$	As Votes, but sums over sub-feature scores rather than counting.
	$RWAN^{C6.2.3}$	As $RWA^{C6.2.2}$, but normalises across stories by the length of each news story.
	$RWA^{C6.2.2}_{Max}$	As $RWA^{C6.2.2}$ but returns the maximum sub-feature score from the retrieved tweets.

Table 6.3: Tweet stream LTRS feature instances used.

Section 6.2 and 6.3). We evaluate these approaches to determine whether they are able to effectively rank news stories (represented by newswire articles) in real-time. To this end, we investigate the following five research questions in the subsequent two evaluation sections (6.6 and 6.7):

1. Are our unsupervised news story ranking approaches (Votes, RWA and RWAN) effective at real-time news story ranking using user-generated content streams? (Section 6.6.1 and Section 6.7.1)

2. Can the application of our $GaussBoost$ and $MaxBurst$ enhancements lead to more effective real-time story ranking by accounting for older discussions in user-generated content streams? (Section 6.7.2 and Section 6.7.3)

3. Are the models produced using our LTRS framework effective at real-time news story ranking when leveraging features from user-generated content streams? (Section 6.6.3 and Section 6.7.4)

4. What story ranking features have the most impact on the ranking of each news story? (Section 6.6.4 and Section 6.7.5)

5. What sort of events does the models produced by our LTRS framework promote when using user-generated content streams? (Section 6.6.5 and Section 6.7.6)

Our evaluation is structured such that we investigate the same research questions on the two streams of data, as described in Section 6.4. In particular, Section 6.6 investigates our unsupervised and learned story ranking approaches on a blog stream, while Section 6.7 investigates the same questions for the tweet stream. We provide concluding remarks in Section 6.8.

Approach	Document Representation	Blogs$_{09}$ MAP	Blogs$_{10}$ statMAP
TREC Best System	Post	0.1862	0.1898
Votes	Post	0.1525	0.1407
RWA	Post	0.1836	0.1846
RWAN	Post	0.1455	0.1300
Votes	Feed	0.1562	0.1353
RWA	Feed	**0.2129**	**0.1949**
RWAN	Feed	0.1644	0.1420

Table 6.4: News story ranking performance of Votes, RWA and RWAN in comparison to the best participating system to the TREC top news stories identification task during 2009 and 2010.

6.6 Evaluation: News Story Ranking with Blogs

In this section, we evaluate our news story ranking approaches – proposed previously in Sections 6.2 and 6.3 – on a stream of blog data. If our news story ranking approaches are able to use discussion within the blogosphere to identify the top events of the moment, then this would illustrate one use where user-generated content can drive a real-time news-related application. Furthermore, the Top Events Identification component that these approaches enable indirectly aids the news search process by providing events rankings to the News Query Classification and News-Related Content Integration components of our framework. Therefore, effective user-generated content driven news story ranking approaches have the potential to aid when satisfying news-related queries within a universal Web search engine.

Note that due to the day-centric nature of the evaluation assessments for the top news stories identification track, the window-size parameter r is set to one day and the background window w is set to 10 days. Furthermore, our second $MaxBurst$ enhancement was developed with Twitter streams in mind, hence we do not apply it to the blog stream in this section.

This section is structured as follows. In Section 6.6.1, we evaluate the effectiveness of Votes, RWA and RWAN at ranking news stories using blogs. Section 6.6.2 evaluates whether the addition of our $GaussBoost$ enhancement to these approaches can increase story ranking effectiveness. In Section 6.6.3, we evaluate story ranking models built using our proposed news story ranking framework (LTRS). Section 6.6.4 investigates the most influential features when ranking using LTRS. Finally, in Section 6.6.5, we examine the types of events that the model produced by our LTRS framework promotes.

6.6.1 Votes, RWA and RWAN

We begin by evaluating the effectiveness of our Votes, RWA and RWAN unsupervised approaches for automatic news story ranking. In particular, we compare the performance of these two approaches on the Blogs$_{09}$ and Blogs$_{10}$ top news stories identification datasets to the best approaches submitted to the TREC 2009 and 2010 Blog track top news stories identification task (see Section 3.2.3). Note that these are strong baselines, representing state-of-the-art approaches to the task. Should any of our news story ranking approaches perform effectively in comparison to them, then we can conclude that by measuring the volume and/or relevance of recent blog postings for a news story, automated ranking of news stories can be achieved, in turn showing the value of user-generated content for this component of our news search framework.

Table 6.4 reports the story ranking effectiveness of Votes, RWA and RWAN when driven by a stream of blogs, in addition to the effectiveness of the best TREC 2009 and 2010 systems for the same task, which we use as baselines. Statistical significance over these baselines (t-test $p<0.05$) was tested, however none of the approaches exceeded the 95% confidence margin. Notably, the stream of blogs can be represented in two manners, either as individual blog posts, or as an aggregate of all of the posts for a blog, referred to as the blog feed. We report ranking effectiveness both when retrieving blog posts or blog feeds from the stream.

From Table 6.4, we note the following four points of interest. First, comparing our approaches (rows 2-7) to the TREC best systems (row 1), we observe that all of our approaches achieve an effectiveness within the range of +-5% absolute of the TREC best systems over both datasets. This shows the viability of modelling story importance as a voting process. Second, comparing the ranking performance of Votes, RWA and RWAN (rows 2-7), we observe that RWA is most effective across both datasets and stream representations. For example, on the Blogs$_{10}$ dataset and using blog posts, RWA outperforms Votes by a margin of 4.4% absolute (0.1846 statMAP to 0.1407 statMAP). This indicates that gains in story ranking effectivenenss can be achieved under our voting approach by considering relatedness of (voting) blog post retrieved for each story, rather than considering all posts equally. Thirdly, if we compare the performance of our approaches when using the two blog stream representations, i.e. blog posts (rows 2-4) and blog feeds (rows 5-7), we see that in all cases except Votes on Blogs$_{10}$, the feed representation leads to more effective story rankings than the post representation. This indicates that blogs that counting multiple blog posts from a single blog is less effective than considering each blog only once. Finally, if we compare our approaches to the TREC best system on each dataset, we observe that RWA on blog posts provides comparable effectiveness, while RWA on blog feeds outperforms the

TREC best systems (0.2129 MAP to 0.1862 MAP for Blogs$_{09}$ and 0.1949 statMAP to 0.1898 statMAP for Blogs$_{10}$).

In answer to our first research question, all three of our unsupervised news story ranking approaches are able to achieve similar effectiveness to the TREC best systems for this task, indeed outperforming them for some settings. This in turn indicates that our news story ranking approaches are state-of-the-art and therefore likely to be useful for ranking news stories in our news search framework. Of the approaches tested the most effective was RWA when applied on blog feeds.

6.6.2 GaussBoost

In Section 6.2.4, we proposed the $GaussBoost$ enhancement to our unsupervised approaches that considers older discussions about each story in addition to more recent discussions. The idea behind $GaussBoost$ is that news stories will often be discussed beforehand for predictable events and/or discussed afterwards for long running, controversial or important unpredictable stories. $GaussBoost$ may be able to better identify those important stories that maintain their interest over time, by taking prior blog post discussion into account.

This technique, has an additional parameter l, which controls the period of time considered and the magnitude of weight assigned to this evidence (see Figure 6.1). We compare the story ranking effectiveness of Votes and RWA unchanged, with their effectiveness when $GaussBoost$ is added (Votes$_{GaussBoost}$ and RWA$_{GaussBoost}$). Moreover, we examine multiple values of the Gaussian curve width l to see which leads to the best story ranking performance, and hence best models the distribution of discussion of important news stories.

Table 6.5 reports the ranking effectiveness of Votes and RWA both with and without $GaussBoost$ over the Blogs$_{09}$ and Blogs$_{10}$ datasets when using l values between 0.2 and 3.0. As baselines, we include the TREC best systems in addition to our Votes and RWA approaches. From these tables, we observe the following. Firstly, we see that over the different news story ranking approaches and datasets, adding $GaussBoost$ can result in increases in news story ranking effectiveness. For example, on the Blogs$_{09}$ dataset and headline representation, MAP increases from 0.1525 to 0.1733 for Votes, and 0.1836 to 0.1900 for RWA, for the best value of l. Secondly, we see that, in general, the story ranking effectiveness rises to a peak and then falls away, as l is increased. However, the optimal value of l differs between the datasets and ranking approaches. For Votes$_{GaussBoost}$, the optimal values of l cluster around 1.6, while for RWA$_{GaussBoost}$, the best values found are more widely distributed across the range of l values tested. With reference to Figure 6.1 from Section 6.2.4, under our blog stream setting, where the window size r is one day, an l value of 1.6 provides almost full weight to posts published on the day preceding the ranking time t, 65% weight on the second day proceeding t and 35% weight on the third day proceeding t. In this way, $GaussBoost$ aggregates evidence (discussion) from a longer period proceeding t than Votes or RWA (that only consider the window size r preceding t), while placing emphasis on the most recent of that evidence.

Overall, we conclude that our $GaussBoost$ enhancement shows promise, as it can increase story ranking effectiveness for both Votes$_{GaussBoost}$ and RWA$_{GaussBoost}$ approaches, answering our second research question. Indeed, for the vast majority of l values tested, adding $GaussBoost$ leads to effectiveness increases over the baseline without $GaussBoost$. Furthermore, using good settings for l, RWA$_{GaussBoost}$ outperforms the TREC best systems. This indicates that using past discussions within the blogosphere can lead to a more accurate estimation of a news story's importance.

6.6.3 Learning to Rank Stories Effectiveness

Having shown that our unsupervised news story ranking models are effective in isolation, we next examine whether by combining different versions of these models can lead to a better estimation of a news story's importance. In particular, we examine whether our LTRS framework (see Section 6.3) can produce a more effective story ranking model than our unsupervised approaches. Recall that LTRS considers each story ranking approach as a feature for ranking, where each feature is comprised of four elements, namely: a story representation; a document ranking approach; document sub-feature; and aggregation model. The elements that we combine to make our story ranking features were described previously in Table 6.2.

To evaluate whether LTRS can produce more effective story ranking models, we once again compare to the TREC best system for each of the Blogs$_{09}$ and Blogs$_{10}$ datasets, in addition to our unsupervised ranking approaches. We also compare against the best individual feature that LTRS uses, since LTRS builds many alternative features (story ranking models) using different elements. Table 6.6 reports the story ranking performance of the best TREC 2009 and 2010 systems as well as the performance of our best individual feature, in comparison to LTRS when trained under both Cross-Corpus and Per-Corpus regimes. ▲ denotes a statistically significant increase over the best individual feature (t-test $p < 0.05$).

From Table 6.6, we observe that under Cross-Corpus training, i.e. training on Blogs$_{09}$ and testing on Blogs$_{10}$, and vice versa, the performance of our approach is lower than the best TREC system. However, when moving to Per-Corpus training, the resulting trained model markedly outperforms the best individual feature used alone by a similar margin (0.2042 MAP to 0.1836 MAP and 0.2248 statMAP to 0.1949 statMAP). Indeed, under per-corpus training, the model produced by LTRS outperformed its best individual feature by a statistically significant margin. This shows that our proposed learning to

Approach	Representation	Gaussian Curve Width l	Blogs$_{09}$ MAP	Blogs$_{10}$ statMAP
TREC Best System	Headline	N/A	0.1862	0.1898
Votes	Headline	N/A	0.1525	0.1407
RWA	Headline	N/A	0.1836	0.1846
Votes$_{GaussBoost}$	Headline	0.2	0.1525	0.1407
Votes$_{GaussBoost}$	Headline	0.4	0.1525	0.1407
Votes$_{GaussBoost}$	Headline	0.6	0.1588	0.1420
Votes$_{GaussBoost}$	Headline	0.8	0.1571	0.1424
Votes$_{GaussBoost}$	Headline	1	0.1568	0.1416
Votes$_{GaussBoost}$	Headline	1.2	0.1630	0.1425
Votes$_{GaussBoost}$	Headline	1.4	0.1636	0.1415
Votes$_{GaussBoost}$	Headline	1.6	**0.1733**	0.1418
Votes$_{GaussBoost}$	Headline	1.8	0.1713	0.1420
Votes$_{GaussBoost}$	Headline	2	0.1709	0.1424
Votes$_{GaussBoost}$	Headline	2.2	0.1674	**0.1435**
Votes$_{GaussBoost}$	Headline	2.4	0.1644	0.1420
Votes$_{GaussBoost}$	Headline	2.6	0.1628	0.1412
Votes$_{GaussBoost}$	Headline	2.8	0.1616	0.1395
Votes$_{GaussBoost}$	Headline	3	0.1593	0.1367
RWA$_{GaussBoost}$	Headline	0.2	0.1836	0.1845
RWA$_{GaussBoost}$	Headline	0.4	0.1836	0.1845
RWA$_{GaussBoost}$	Headline	0.6	0.1810	0.1876
RWA$_{GaussBoost}$	Headline	0.8	**0.1900**	0.1882
RWA$_{GaussBoost}$	Headline	1	0.1844	0.1878
RWA$_{GaussBoost}$	Headline	1.2	0.1741	**0.1890**
RWA$_{GaussBoost}$	Headline	1.4	0.1754	0.1888
RWA$_{GaussBoost}$	Headline	1.6	0.1712	0.1864
RWA$_{GaussBoost}$	Headline	1.8	0.1708	0.1862
RWA$_{GaussBoost}$	Headline	2	0.1698	0.1857
RWA$_{GaussBoost}$	Headline	2.2	0.1667	0.1850
RWA$_{GaussBoost}$	Headline	2.4	0.1668	0.1846
RWA$_{GaussBoost}$	Headline	2.6	0.1680	0.1797
RWA$_{GaussBoost}$	Headline	2.8	0.1674	0.1773
RWA$_{GaussBoost}$	Headline	3	0.1663	0.1757

Table 6.5: News story ranking performance of Votes$_{GaussBoost}$ and RWA$_{GaussBoost}$ when the Gaussian curve width l is varied.

Model	Training	Blogs_{09} (TREC 2009)	Blogs_{10} (TREC 2010)
TREC Best System	N/A	0.1862	0.1898
Votes	N/A	0.1525	0.1407
RWA	N/A	0.1836	0.1846
RWAN	N/A	0.1455	0.1300
$\text{RWA}_{GaussBoost}$ (Best)	N/A	0.1900	0.1890
Best Individual Feature	N/A	0.1836	0.1949
LTRS	Cross-Corpus	0.1165	0.1689
	Per-Corpus	**0.2042▲**	**0.2248▲**

Table 6.6: Comparison between our learning to rank approach when trained under both Cross-Corpus and Per-Corpus training with the best TREC systems in terms of overall story ranking performance under the Blogs_{09} and Blogs_{10} story ranking topics. ▲ denotes a statistically significant increase over the best individual feature (t-test $p < 0.05$).

rank approach can indeed be effective for real-time story ranking (under Per-Corpus training). On the other hand, the lesser performance observed when using Cross-Corpus training indicates that the best features for story ranking are different for the two news corpora and topic sets. This is somewhat to be expected, as the news corpora underlying the Blogs_{09} and Blogs_{10} datasets differ markedly in both the story writing style as well as the level of noise contained. Hence, we would expect the effective features to vary across datasets.

Overall, based upon LTRS's effective performance in comparison to our baselines, we conclude that LTRS is more effective than our unsupervised approaches, answering our third research question. Indeed, this makes it more suitable to use as the Top Events Identification component.

6.6.4 Influential Story Ranking Features

To answer our final research question, we investigate which of the 160 features generated using LTRS contribute most to the ranking of news stories. In particular, we examine the features selected in terms of the four elements that comprise each feature, namely: the story representation; the document ranking approach; the document sub-feature; and the aggregation model. We draw conclusions regarding which of these component combinations and hence elements are most heavily influencing our news story ranking. Table 6.7 reports the five most influential positive and negative features and their associated weights that were selected by LTRS when trained on all of the instances in each dataset. Recall that the features used under LTRS are normalised such that the weights reported are comparable.

From Table 6.7, we observe that of the aggregation models tested (see Table 6.2), RWA is preferred across both topic sets, as it is the aggregation model used within the majority of top features. This indicates that RWA produces features better able to distinguish between newsworthy and non-newsworthy

Feature Type	Blogs$_{09}$ Elements				Blogs$_{10}$ Elements			
	Aggregation Model	Story Representation	Document Ranking Approach	Weight	Aggregation Model	Story Representation	Document Ranking Approach	Weight
Positive	RWA	Headline	$DPH_{1000,r=1day}$	**0.9703**	RWA	Headline	$DPH_{1000,r=1day}$	**0.7719**
Positive	RWA	Summary	$DPH_{1000,r=1day}$	**0.2762**	RWA	Content	$DPH_{1000,r=2days}$	**0.1406**
Positive	RWA	Content	$DPH_{1000,r=1day}$	**0.2646**	RWA	Noun-Phrases	$DPH_{1000,r=5days}$	0.0468
Positive	$BM25_{SUM}$	Summary	$DPH_{1000,r=2days}$	**0.1213**	$BM25_{SUM}$	Summary	$DPH_{1000,r=3days}$	0.0196
Positive	RWA	QE_Blogs06	$DPH_{1000,r=3days}$	0.0982	RWA	QE_TRC2	$DPH_{1000,r=3days}$	0.0173
Negative	RWA	Entities	$DPH_{1000,r=8days}$	-0.0063	RWA	Headline	$DPH_{1000,r=6days}$	-0.0035
Negative	RWA	Content	$DPH_{1000,r=8days}$	-0.0286	RWA	Content	$DPH_{1000,r=6days}$	-0.0038
Negative	RWA	QE_TRC2	$DPH_{1000,r=7days}$	**-0.1032**	RWA	QE_NYT06	$DPH_{1000,r=7days}$	-0.0063
Negative	$BM25_{SUM}$	Noun-Phrases	$DPH_{1000,r=1days}$	**-0.1143**	RWA	Noun-Phrases	$DPH_{1000,r=10days}$	-0.0077
Negative	RWA	Headline	$DPH_{1000,r=7days}$	**-0.5215**	RWA	Summary	$DPH_{1000,r=6days}$	-0.0101

Table 6.7: Strongest 5 positive and negative features on the Blogs$_{09}$ and Blogs$_{10}$ datasets. Boldened feature weights indicate features with a high impact on the story ranking. The document feature component is not reported, as only document relevance is used in this experiment.

stories than our Votes approach or indeed the $BM25_{SUM}$ alternative proposed by Xu *et al.* (2010) (see Section 3.2.3) that we used as a feature. Of the eight story representations listed previously in Table 6.2, we see that the headline alone is used by the most influential features, indicating that it is the most useful. However, we also observe that features using the news article content and summary representations were also selected. This indicates that the news article content is more noisy, but still provides valuable ranking evidence. In terms of the time window (of size r) that we impose upon the document ranking, we make the following two observations. Firstly, only features that use blog posts from the one or two days before the time of ranking appear to be useful. Secondly, as we relax the temporal restriction and use older blog posts, the story ranking features become negative, i.e. if a story has been discussed extensively beforehand then the story is less likely to be newsworthy. Indeed, for the Blogs$_{09}$ dataset, the strongest positive feature (RWA + Headline + $DPH_{1000,r=1day}$) becomes the strongest negative feature by changing the temporal restriction. Notably, in general, negative features appear not to be influential on the Blogs$_{10}$ dataset, i.e. they receive very low weight. This indicates that there is little to be gained by leveraging older discussions on this dataset.

Overall, we have shown through analysis of the features that LTRS selected to form our news story ranking model that RWA remained an effective story ranking feature. Furthermore, we have shown that LTRS is able to leverage different news story representations and time restrictions to produce a better story ranking model than our unsupervised approaches and the best TREC systems for this task.

6.6.5 Event Promotion in Blogs

When using user-generated content streams to rank news events, it is of interest the types of event that make the top ranks. In this section, we examine the types of events that LTRS favours. One might expect that the blogosphere, as a slower news reporting medium in comparison to Twitter, might favour predictable events over unpredictable ones. To test this, we select the top 5 most newsworthy stories as returned by our learned approach for each of the 50 days that we rank stories for in the Blogs$_{10}$ dataset, creating a set of 250 newsworthy stories. We manually annotated each of these as reporting about predictable or unpredictable events. Should the blogosphere lack sufficient freshness, then our story ranking approach will be more likely to identify predictable events over unpredictable ones, i.e. the vast majority of the 250 news stories would relate to predictable events.

However, from analysis of the annotated stories, we found that 46% of the top stories were unpredictable, while 54% were predictable (a close to even spread). This indicates that, at least for the simulated real-time setting introduced by TREC, there is no evidence to indicate that bloggers react too slowly for unpredictable stories to be effectively ranked.

6.6.6 Conclusions

In this section, we evaluated our unsupervised news story ranking approaches on a stream of blogs. We showed that these approaches can be effective at ranking news stories in a real-time manner when using recent discussion within blogs to estimate news story importance, under the day-centric constraints of the TREC top news stories identification task. Of the approaches tested, we showed that RWA was the most effective model and that our proposed $GaussBoost$ enhancement was able to increase story ranking effectiveness. Our results also showed that combining these approaches under our proposed LTRS framework can generate an even more effective model. We conclude that both our unsupervised approaches and LTRS framework appear to be effective and therefore have the potential to aid when satisfying news-related queries within a universal Web search engine. In the next section, we examine whether this result holds when using a stream of tweets rather than blogs/blog posts to drive story importance estimations.

6.7 Evaluation: News Story Ranking with Tweets

In the previous section (Section 6.6), we evaluated the story ranking effectiveness of our learned and unsupervised story ranking approaches on a stream of blogs. Experimental results showed that both our proposed unsupervised and learned approaches outperform the state-of-the-art approaches deployed at TREC for the task. We now shift our story ranking evaluation to use a real-time stream of tweets, i.e. we use recent discussions in Twitter to estimate the current importance of newswire articles. The aim is to determine whether our proposed approaches remain effective when using the higher-volume and more noisy tweet stream.

As described in Section 6.4.2, we evaluate our news story ranking approaches on three tweet datasets, namely: Twitter$_{11-NYT}$, Twitter$_{12-NYT}$ and Twitter$_{12-TR}$ (see Section 5.2.1). These datasets contain news story rankings from major news providers against which we compare the rankings produced my our approaches. The goal is for our approaches to produce similar rankings to those produced by said news providers, by leveraging a real-time tweet stream to identify the stories that are the most important. Indeed, our approaches are restricted to a fully-real-time setting, such that for a ranking downloaded from a news provider at time t, our approaches have only access to tweets made before t with which to determine importance.

The outline of our evaluation follows the same structure as Section 6.6, that evaluated our untrained story ranking approaches on the blog stream. First, we compare the effectiveness of Votes, RWA, RWAN at ranking news stories using tweets in Section 6.7.1, both when considering each individual tweet as a vote, and the aggregate the tweets from a single user. We evaluate the effect that adding $GaussBoost$ to our unsupervised approaches has on story ranking effectiveness in Section 6.7.2. In Section 6.7.3 we examine whether our $MaxBurst$ enhancement can improved news story ranking effectiveness on tweets. Section 6.7.4 evaluates story ranking models produced by our LTRS framework. In Section 6.7.5, we examine the most effective features selected. Finally, in Section 6.6.5, we examine the types of events that the model produced by our LTRS framework promotes when using real-time discussion from tweets.

6.7.1 Votes, RWA and RWAN

We begin our evaluation of news story ranking using tweets by evaluating the effectiveness of our Votes, RWA and RWAN unsupervised approaches. Should any of our approaches perform effectively, then we can conclude that both the volume and relevance of tweets related to a news story at a point in time is indicative of its importance at that time, and that these ranking approaches are generalisable enough to leverage evidence from both blog and tweet streams. Table 6.8 reports the story ranking effectiveness of Votes, RWA and RWAN when driven by a stream of tweets. Notably, the stream of tweets can be represented in two manners, either as individual tweets (Tweet), or as an aggregate of all of the posts from a single tweeter (Twitterer-Feed), similar to the feed representation from our blog stream experiments. We report ranking effectiveness both when retrieving single tweets and the aggregate of those tweets from each user from the stream. Unlike for our blog stream experiments, we cannot compare to the best TREC systems for this task. Instead, we use the performance of a random ranking of stories (averaged over ten of such rankings) as a baseline. Furthermore, we use our Votes approach that was shown to be effective on the blogs stream as a second baseline. We denote statistically significant

Approach	Document Representation	Effectiveness					
		Twitter$_{11-NYT}$		Twitter$_{12-NYT}$		Twitter$_{12-TR}$	
		NDCG	Success@1	NDCG	Success@1	NDCG	Success@1
Random	N/A	0.6674	-	0.6991	-	0.7363	-
Votes	Tweet	0.7443	0.0888	0.7313	0.0178	0.7650	**0.2380**
Votes	Twitterer-Feed	0.7418	0.0888	0.7342	0.0357	0.7674	0.2222
RWA	Tweet	**0.7666**▲ ▲	**0.1333**	**0.7820**▲ ▲	**0.2678**	0.7324▼ ▼	0.1111
RWA	Twitterer-Feed	0.7550	0.0444	0.7776▲ ▲	0.0714	0.6988▼ ▼	0.1587
RWAN	Tweet	0.7022▼ ▼	0.0444	0.7328	0.0357	0.7765▲	0.1904
RWAN	Twitterer-Feed	0.6708▼ ▼	0.0000	0.7296	0.0357	**0.7780**▲	0.0952
Topic Count		45	45	71	56	63	63

Table 6.8: News story ranking performance of Votes, RWA and RWAN in comparison to the random baseline. ▲ and ▼ indicate statistically significant increases/decreases over Votes using the Tweet representation under *t-test* $p < 0.05$. ▲▲ and ▼▼ indicate statistically significant increases/decreases over Votes using the Tweet representation under *t-test* $p < 0.01$.

increases and decreases using t-test at p<0.05 in comparison to Votes using single ▲ and ▼ symbols, two such symbols indicate significant increases and decreases using t-test at p<0.01.

From Table 6.8, we observe the following. Firstly, we observe that under most settings, our approaches markedly outperform the random baseline. This shows that news story ranking with tweets is feasible. Furthermore, we note that random effectiveness is high for these experiments because we are ranking a small set of stories for each topic and always return all stories, i.e. recall is 1. The only way to increase NDCG effectiveness is to rank important stories from the set in higher ranks.

Secondly, with regard to our news story ranking approaches, we see that different approaches are better for different datasets. In particular, RWA (rows 4-5) outperforms both Votes (rows 2-3) and RWAN (rows 6-7) on both Twitter$_{11-NYT}$ and Twitter$_{12-NYT}$ datasets, while RWAN was the most effective on Twitter$_{12-TR}$ dataset under NDCG. This indicates that varying story length – which RWAN accounts for but RWA does not – is much more of an issue when ranking Reuters news stories that those from the New York Times. Moreover, this difference leads to markedly different news story ranking performance. For the New York Times datasets, RWA is significantly more effective than Votes, but significantly less effective on the Reuters dataset.

Thirdly, in terms of Success@1, i.e. the proportion of times that we rank the top story in the top rank, we see that our best approach for each dataset correctly ranks the most important story between 13% (Twitter$_{11-NYT}$) and 26% (Twitter$_{12-NYT}$) of the time. This shows that while our approaches are able to rank news stories by their importance, there is scope for improvement in identifying the top story of the moment.

Finally, comparing the news story ranking approaches when using the Tweet and Twitterer-Feed representations, unlike for our blog stream experiments, we observe that there is little difference in effectiveness between them. Indeed, this is because few tweets that receive votes under this setting come from the same Twitterer. Hence, the post vs feed representation is not as influential in a Twitter setting than it was when using blogs/blog posts.

Overall, we conclude that our news story ranking approaches able to rank news stories using a high volume Twitter stream, in addition to the blog stream tested earlier. Indeed, with regard to our first research question, we conclude that Votes, RWA and RWAN can indeed be effective when ranking using tweets, although there is a higher variance in effectiveness between datasets than was observed for blogs. Furthermore, from the low Success@1 performances reported, we see that there is scope for better identification of the top story of the moment.

6.7.2 GaussBoost

Recall that in Section 6.2.4 we proposed $GaussBoost$, an extension to our approaches that takes into account the volume of discussion in the user-generated content stream in question from time periods more distant from the time of the query, i.e. using older discussions from the stream. Later, when we tested $GaussBoost$ on a stream of blogs, we observed some improvements in story ranking effectiveness. In this section, we examine the effectiveness of $GaussBoost$ on our tweet datasets. In particular, we aim to determine whether by explicitly taking into account the volume and relevance of discussion in Twitter before the period defined as recent, i.e. the day before the time of ranking, can lead to a better ranking of news stories. $GaussBoost$ divides the background window of tweet evidence into temporal units of size r. The magnitude of emphasis placed on each temporal unit can be controlled via the parameter l, i.e. the width of the Gaussian curve. We test l values within the range 0.2 to 3.0, in 0.2 increments.

Table 6.9 reports the news story ranking effectiveness of our Votes and RWA approaches with and without $GaussBoost$ for the aforementioned l values. Votes and RWA unaltered are our baselines. Statistically significant (t-test $p<0.01$ and $p<0.05$) increases and decreases over these baselines are denoted using the ▲ and ▼ symbols. From Table 6.9, we see that for the majority of parameter settings and datasets, adding $GaussBoost$ leads to significant decreases in story ranking effectiveness. Moreover, we observe that as we increase the value of l, i.e. we consider older evidence, story ranking effectiveness tends to decrease. In turn, this indicates that either older tweet evidence is not directly useful for the ranking of news stories by their importance, or that a Gaussian curve does not well model the distribution of tweets for important news stories. To answer our second research question, on this dataset $GaussBoost$ does not appear to be effective, unlike in the when used on the blog stream.

Approach	Gaussian Curve Width l	Effectiveness					
		Twitter$_{11-NYT}$		Twitter$_{12-NYT}$		Twitter$_{12-TR}$	
		NDCG	Success@1	NDCG	Success@1	NDCG	Success@1
Votes	None	**0.7443**	**0.0888**	0.7313	0.0178	**0.7650**	0.2380
Votes$_{GaussBoost}$	0.2	0.7442	**0.0888**	0.7313	0.0178	**0.7650**	**0.2698**
Votes$_{GaussBoost}$	0.4	**0.7443**	0.0666	0.7313	0.0178	**0.7650**	0.2222
Votes$_{GaussBoost}$	0.6	0.7305▼ ▼	0.0444	0.7333	0.0714	0.7620	**0.2698**
Votes$_{GaussBoost}$	0.8	0.7252▼ ▼	0.0444	0.7377	0.1071	0.7615	0.2539
Votes$_{GaussBoost}$	1.0	0.7204▼ ▼	0.0444	0.7371	0.1071	0.7605	0.2222
Votes$_{GaussBoost}$	1.2	0.7308▼	0.0444	0.7415	0.1071	0.7585	0.2380
Votes$_{GaussBoost}$	1.4	0.7287▼ ▼	0.0444	0.7427	**0.1250**	0.7551▼	0.2539
Votes$_{GaussBoost}$	1.6	0.7211▼ ▼	0.0444	0.7450▲	**0.1250**	0.7568	0.2222
Votes$_{GaussBoost}$	1.8	0.7180▼ ▼	0.0444	0.7420	**0.1250**	0.7553▼	0.2380
Votes$_{GaussBoost}$	2.0	0.7192▼ ▼	0.0444	0.7396	**0.1250**	0.7554	0.2380
Votes$_{GaussBoost}$	2.2	0.6960▼ ▼	0.0444	0.7415	0.1071	0.7541	0.1904
Votes$_{GaussBoost}$	2.4	0.6944▼ ▼	0.0444	0.7438	0.1071	0.7518▼	0.1746
Votes$_{GaussBoost}$	2.6	0.6851▼ ▼	0.0444	0.7455	0.0892	0.7542	0.1746
Votes$_{GaussBoost}$	2.8	0.6828▼ ▼	0.0444	0.7466▲	0.0892	0.7548	0.1746
Votes$_{GaussBoost}$	3.0	0.6792▼ ▼	0.0444	**0.7481▲**	0.0892	0.7546	0.1746
RWA	None	**0.7666**	**0.1333**	0.7820	**0.2678**	**0.7324**	0.1111
RWA$_{GaussBoost}$	0.2	0.7665	**0.1333**	0.7820	**0.2678**	**0.7324**	0.1428
RWA$_{GaussBoost}$	0.4	0.7665	**0.1333**	**0.7821**	**0.2678**	**0.7324**	0.1587
RWA$_{GaussBoost}$	0.6	0.7552▼ ▼	0.0444	0.7774	0.2142	0.7217▼ ▼	0.1111
RWA$_{GaussBoost}$	0.8	0.7550▼ ▼	0.0444	0.7781	0.1785	0.7184▼ ▼	0.1111
RWA$_{GaussBoost}$	1.0	0.7551▼ ▼	0.0444	0.7778	0.1607	0.7170▼ ▼	0.1269
RWA$_{GaussBoost}$	1.2	0.7533▼ ▼	0.0444	0.7732▼	0.0892	0.7136▼ ▼	0.1746
RWA$_{GaussBoost}$	1.4	0.7542▼	0.0444	0.7735▼	0.0892	0.7132▼ ▼	0.1111
RWA$_{GaussBoost}$	1.6	0.7544▼	0.0444	0.7723▼	0.0892	0.7094▼ ▼	**0.1904**
RWA$_{GaussBoost}$	1.8	0.7546▼	0.0444	0.7717▼	0.0892	0.7089▼ ▼	0.1428
RWA$_{GaussBoost}$	2.0	0.7560	0.0444	0.7724▼	0.0892	0.7087▼ ▼	0.1587
RWA$_{GaussBoost}$	2.2	0.7470▼ ▼	0.0444	0.7765	0.1607	0.7070▼ ▼	0.1428
RWA$_{GaussBoost}$	2.4	0.7455▼ ▼	0.0444	0.7765	0.1607	0.7067▼ ▼	0.1428
RWA$_{GaussBoost}$	2.6	0.7383▼ ▼	0.0444	0.7755	0.1607	0.7068▼ ▼	0.1269
RWA$_{GaussBoost}$	2.8	0.7381▼ ▼	0.0444	0.7758	0.1607	0.7068▼ ▼	**0.1904**
RWA$_{GaussBoost}$	3.0	0.7363▼ ▼	0.0444	0.7755	0.1607	0.7059▼ ▼	**0.1904**
Topic Count		45	45	71	56	63	63

Table 6.9: News story ranking performance of Votes$_{GaussBoost}$ and RWA$_{GaussBoost}$ in comparison to Votes and RWA. The most effective NDCG and Success@1 performance observed under each approach and dataset is highlighted. ▲ and ▼ indicate statistically significant increases/decreases over the baseline run without $GaussBoost$ under *t-test* $p < 0.05$. ▲▲ and ▼▼ indicate statistically significant increases/decreases under *t-test* $p < 0.01$.

Approach	Document Representation	Effectiveness					
		Twitter$_{11-NYT}$		Twitter$_{12-NYT}$		Twitter$_{12-TR}$	
		NDCG	Success@1	NDCG	Success@1	NDCG	Success@1
Votes	Tweet	0.7443	0.0888	0.7313	0.0178	0.7650	0.2380
Votes$_{MaxBurst}$	TweetMax	0.7500	**0.0444**	0.7523▲ ▲	0.0535	0.7516▼	**0.2539**
Votes$_{MaxBurst}$	Twitterer-FeedMax	0.7492	**0.0444**	0.7521▲ ▲	0.0357	0.7528▼	0.2380
RWA	None	0.7666	**0.1333**	0.7820	**0.2678**	**0.7324**	0.1111
RWA$_{MaxBurst}$	TweetMax	**0.7761**	0.0444	0.7898▲	**0.1964**	0.7149▼ ▼	0.1269
RWA$_{MaxBurst}$	Twitterer-FeedMax	0.7757	0.0444	**0.7907**▲	**0.1964**	0.7134▼ ▼	0.1111
RWAN	Tweet	0.7022	**0.0444**	0.7328	0.0357	0.7765	0.1904
RWAN$_{MaxBurst}$	TweetMax	0.7229▲ ▲	**0.0444**	0.7344	0.0357	0.7697	0.1587
RWAN$_{MaxBurst}$	Twitterer-FeedMax	0.7229▲ ▲	**0.0444**	0.7354	0.0357	**0.7704**	0.1746
Topic Count		45	45	71	56	63	63

Table 6.10: News story ranking performance of Votes$_{MaxBurst}$, RWA$_{MaxBurst}$ and RWAN$_{MaxBurst}$. The most effective NDCG and Success@1 performance observed under each approach and dataset is highlighted. ▲ and ▼ indicate statistically significant increases/decreases over the base news story ranking approach without $MaxBurst$ under *t-test* $p < 0.05$. ▲▲ and ▼▼ indicate statistically significant increases/decreases under *t-test* $p < 0.01$.

6.7.3 MaxBurst

In this section, we evaluate the effectiveness of ranking news stories by the maximum observed importance of a story from any point in the background time window, instead of focusing only on a period just before t. In particular, this approach, referred to as $MaxBurst$, was proposed specifically for the Twitter stream as we suspect that short bursts of discussion within Twitter will result in overly zealous demotion of older news stories. We evaluate the effectiveness of $MaxBurst$ when using Votes, RWA and RWAN to estimate news story importance over the time period of the background window. We compare $MaxBurst$ to Votes to see if news story ranking effectiveness increases. If so, then we can conclude that focusing only on a recent time period is not the best strategy, and that we need to consider its prior importance as well. In a similar fashion to Section 6.7.1, we report story ranking effectiveness for both Tweet and Twitterer-Feed document representations.

Table 6.10 reports the story ranking effectiveness of our approaches with and without $MaxBurst$. Statistically significant (t-test p<0.01 and p<0.05) increases and decreases over Votes, RWA and RWAN are denoted using the ▲ and ▼ symbols. From Table 6.10, we see that by adding $MaxBurst$, improvements in news story ranking performance are gained for the New York Times datasets. For instance, the addition of $MaxBurst$ to RWAN on the Twitter$_{11-NYT}$ dataset results in statistically significant increases in story ranking effectiveness (0.7022 to 0.7229) under NDCG. In contrast, for the Reuters dataset, we observe small improvements in performance for the most effective approach on that dataset (RWAN), but statistically significant decreases when adding $MaxBurst$ to the less effective approaches. From these results, we conclude that $MaxBurst$ shows promise, as it can improve story ranking for the majority of settings tested. Hence, focusing only on a recent time period is not the best strategy. However, while improvements are observed, these are not large in nature. This may be because $MaxBurst$ does not account the age of the news story at all, i.e. it measures the maximum observed importance for a background time window. Instead, it may be advantageous to reduce the maximum observed importance for a news story in line with the time since that maximum occurred. To test this, we combine $MaxBurst$ with $GaussBoost$, as described in Section 6.2.5.

Table 6.11 reports the story ranking effectiveness of Votes and RWA when both $MaxBurst$ and $GaussBoost$ are combined, for various values of l (that controls the degree that we decrease the voting power of older tweets). The larger the l value, the more emphasis we place on older tweets. Statistically significant (t-test p<0.01 and p<0.05) increases and decreases over Votes and RWA are denoted using the ▲ and ▼ symbols.

From Table 6.11, we observe that for the majority of settings tested, decreasing the scores of older tweets, even with the addition of $MaxBurst$ continues to degrade story ranking effectiveness. Indeed, for low values of l (0.2/0.4), story ranking is identical to the basic approach without $MaxBurst$ and $GaussBoost$, as all emphasis is placed on the most recent temporal unit. As we increase l, taking into account the $MaxBurst$ evidence with increasing emphasis, ranking effectiveness decreases for all settings except Twitter$_{12-NYT}$. For large values of l, where older stories receive close to full weight, story ranking increases once again, but not by a sufficient margin to outperform the baseline. From this result, we conclude that it is not effective to trade off emphasis given to older tweets by combining $GaussBoost$ and $MaxBurst$.

In summary, over the course of this section we have experimented with the unsupervised news story ranking approaches that we proposed in Section 6.2 on three tweet/news story datasets. We have shown that our unsupervised approaches remain effective on when deployed on tweets in addition to blogs. However, unlike when using blogs, leveraging older discussions about each story using $GaussBoost$ was not effective when using tweets. On the other hand, our proposed $MaxBurst$ enhancement significantly increased effectiveness over our basic approaches, although combining both $GaussBoost$ and $MaxBurst$ did not.

6.7.4 Learning to Rank Stories Effectiveness

In this section, we evaluate the effectiveness of models produced by our LTRS framework. Recall that LTRS considers each story ranking approach as a feature for ranking, where each feature is comprised of four elements. The elements that we combine to make our story ranking features were described previously in Table 6.3. LTRS uses training data to produce a story ranking model. However, due to

Approach	Gaussian Curve Width l	Effectiveness					
		Twitter$_{11-NYT}$		Twitter$_{12-NYT}$		Twitter$_{12-TR}$	
		NDCG	Success@1	NDCG	Success@1	NDCG	Success@1
Votes	None	**0.7443**	0.0888	0.7313	0.0178	**0.7650**	0.2380
Votes$_{MaxBurst+GaussBoost}$	0.2	**0.7442**	**0.0888**	0.7314	0.0178	0.7649	0.2222
Votes$_{MaxBurst+GaussBoost}$	0.4	**0.7442**	**0.0888**	0.7313	0.0178	**0.7650**	**0.2857**
Votes$_{MaxBurst+GaussBoost}$	0.6	0.7306▼▼	0.0444	0.7333	0.0714	0.7620	0.2698
Votes$_{MaxBurst+GaussBoost}$	0.8	0.7254▼▼	0.0444	0.7378	0.1071	0.7615	0.2222
Votes$_{MaxBurst+GaussBoost}$	1.0	0.7206▼▼	0.0444	0.7371	0.1071	0.7606	0.2222
Votes$_{MaxBurst+GaussBoost}$	1.2	0.7308▼	0.0444	0.7415	0.1071	0.7583	0.2539
Votes$_{MaxBurst+GaussBoost}$	1.4	0.7287▼▼	0.0444	0.7427	**0.1250**	0.7551▼	0.2063
Votes$_{MaxBurst+GaussBoost}$	1.6	0.7212▼▼	0.0444	0.7450▲	**0.1250**	0.7571	0.2539
Votes$_{MaxBurst+GaussBoost}$	1.8	0.7189▼▼	0.0444	0.7420	**0.1250**	0.7556	0.2380
Votes$_{MaxBurst+GaussBoost}$	2.0	0.7192▼▼	0.0444	0.7396	**0.1250**	0.7554	0.2222
Votes$_{MaxBurst+GaussBoost}$	2.2	0.6960▼▼	0.0444	0.7415	0.1071	0.7540	0.2222
Votes$_{MaxBurst+GaussBoost}$	2.4	0.6947▼▼	0.0444	0.7438	0.1071	0.7518▼	0.2063
Votes$_{MaxBurst+GaussBoost}$	2.6	0.6851▼▼	0.0444	0.7455	0.0892	0.7542	0.1746
Votes$_{MaxBurst+GaussBoost}$	2.8	0.6827▼▼	0.0444	0.7466▲	0.0892	0.7548	0.1746
Votes$_{MaxBurst+GaussBoost}$	3.0	0.6789▼▼	0.0444	**0.7482▲**	0.0892	0.7546	0.1746
Votes$_{MaxBurst+GaussBoost}$	4.0	0.6632▼▼	0.0444	0.7466▲	0.0892	0.7529▼	0.1746
Votes$_{MaxBurst+GaussBoost}$	5.0	0.6674▼▼	0.0444	0.7452	0.0892	0.7546	0.1746
Votes$_{MaxBurst+GaussBoost}$	6.0	0.6669▼▼	0.0444	0.7459	0.0892	0.7539	0.1746
Votes$_{MaxBurst+GaussBoost}$	7.0	0.6663▼▼	0.0444	0.7466▲	0.0892	0.7537	0.1746
Votes$_{MaxBurst+GaussBoost}$	8.0	0.6650▼▼	0.0444	0.7470▲	0.0892	0.7539	0.1746
RWA	None	**0.7666**	**0.1333**	**0.7820**	**0.2678**	**0.7324**	0.1111
RWA$_{MaxBurst+GaussBoost}$	0.2	**0.7665**	**0.1333**	**0.7820**	**0.2678**	**0.7324**	0.1587
RWA$_{MaxBurst+GaussBoost}$	0.4	**0.7665**	**0.1333**	**0.7820**	**0.2678**	**0.7324**	0.1428
RWA$_{MaxBurst+GaussBoost}$	0.6	0.7552▼▼	0.0444	0.7774	0.2142	0.7217▼▼	0.0952
RWA$_{MaxBurst+GaussBoost}$	0.8	0.7550▼▼	0.0444	0.7780	0.1785	0.7184▼▼	0.1587
RWA$_{MaxBurst+GaussBoost}$	1.0	0.7551▼▼	0.0444	0.7778	0.1607	0.7170▼▼	0.1269
RWA$_{MaxBurst+GaussBoost}$	1.2	0.7533▼▼	0.0444	0.7732▼	0.0892	0.7136▼▼	0.1428
RWA$_{MaxBurst+GaussBoost}$	1.4	0.7541▼	0.0444	0.7735▼	0.0892	0.7132▼▼	0.1269
RWA$_{MaxBurst+GaussBoost}$	1.6	0.7544▼	0.0444	0.7723▼	0.0892	0.7094▼▼	0.0952
RWA$_{MaxBurst+GaussBoost}$	1.8	0.7547▼	0.0444	0.7717▼	0.0892	0.7089▼▼	0.1428
RWA$_{MaxBurst+GaussBoost}$	2.0	0.7558	0.0444	0.7724▼	0.0892	0.7087▼▼	0.1428
RWA$_{MaxBurst+GaussBoost}$	2.2	0.7470▼▼	0.0444	0.7765	0.1607	0.7069▼▼	0.1269
RWA$_{MaxBurst+GaussBoost}$	2.4	0.7455▼▼	0.0444	0.7765	0.1607	0.7067▼▼	0.1269
RWA$_{MaxBurst+GaussBoost}$	2.6	0.7383▼▼	0.0444	0.7755	0.1607	0.7068▼▼	0.1587
RWA$_{MaxBurst+GaussBoost}$	2.8	0.7381▼▼	0.0444	0.7758	0.1607	0.7068▼▼	0.1111
RWA$_{MaxBurst+GaussBoost}$	3.0	0.7362▼▼	0.0444	0.7755	0.1607	0.7059▼▼	0.1428
RWA$_{MaxBurst+GaussBoost}$	4.0	0.7367▼▼	0.0444	0.7639▼▼	0.0892	0.7028▼▼	**0.1746**
RWA$_{MaxBurst+GaussBoost}$	5.0	0.7507	0.0444	0.7648▼▼	0.0892	0.7021▼▼	0.1587
RWA$_{MaxBurst+GaussBoost}$	6.0	0.7522	0.0444	0.7747	0.1071	0.7010▼▼	**0.1746**
RWA$_{MaxBurst+GaussBoost}$	7.0	0.7523	0.0444	0.7744	0.1071	0.6999▼▼	0.1587
RWA$_{MaxBurst+GaussBoost}$	8.0	0.7533	0.0444	0.7741	0.0892	0.7001▼▼	0.1111
Topic Count		45	45	71	56	63	63

Table 6.11: News story ranking performance of Votes$_{MaxBurst+GaussBoost}$ and RWA$_{MaxBurst+GaussBoost}$ in comparison to Votes and RWA. The most effective NDCG and Success@1 performance observed under each approach and dataset is highlighted. ▲ and ▼ indicate statistically significant increases/decreases over the base news story ranking approach without $MaxBurst$ and $GaussBoost$ under *t-test* $p < 0.05$. ▲▲ and ▼▼ indicate statistically significant increases/decreases under *t-test* $p < 0.01$.

Dataset	Approach	Training Dataset	NDCG	Success@1
Twitter$_{11-NYT}$	Random	None	0.6674	-
	Best Individual Feature	None	0.7666▲ ▲	0.1333
	LTRS	Twitter$_{12-NYT}$	0.6738	**0.2888**
	LTRS	Twitter$_{12-TR}$	**0.7718▲ ▲**	0.1111
Twitter$_{12-NYT}$	Random	None	0.6991	-
	Best Individual Feature	None	**0.7820▲ ▲**	**0.2678**
	LTRS	Twitter$_{11-NYT}$	0.7254	0.1250
Twitter$_{12-TR}$	Random	None	0.7363	-
	Best Individual Feature	None	**0.7780▲**	0.0952
	LTRS	Twitter$_{11-NYT}$	0.7743▲	**0.2352**

Table 6.12: Learned story ranking performance when training across story ranking datasets when using AFS. The most effective NDCG and Success@1 performance observed under each approach and dataset is highlighted. ▲ and ▼ indicate statistically significant increases/decreases over the random baseline under *t-test* $p < 0.05$. ▲▲ and ▼▼ indicate statistically significant increases/decreases under *t-test* $p < 0.01$.

the real-time nature of our evaluation, we need to avoid training and testing on two settings from the same timeframe, even if the news providers for each are different. This is because both providers may publish about the same stories; hence a story may appear in both training and testing sets. This means that we cannot train on Twitter$_{12-NYT}$ and test on Twitter$_{12-TR}$ and vice-versa. Hence, in this section, we report the news story ranking effectiveness of LTRS when training on the topics from one of our three datasets, while testing on another. This is equivalent to the Cross-Corpus training regime used during our experiments upon the blog stream (see Section 6.6.3). Effectiveness is reported over the three Twitter datasets in terms of NDCG and Success@1 measures.

Table 6.12 reports the story ranking effectiveness of the models produced by LTRS in comparison to the random baseline and the best individual feature that LTRS could have selected. Statistically significant (t-test p<0.01 and p<0.05) increases and decreases over the best individual feature is denoted using the ▲ and ▼ symbols. From Table 6.12, we observe the following. Firstly, as a sanity check, we observe that LTRS always outperforms the random baseline. This shows that the approach is able to learn something from news story rankings from a different point in time. Next, if we compare our learned approach to the best individual feature for each dataset, then we see that LTRS outperforms the best feature once (row 4), provides equivalent performance once (row 10), and underperforms twice (rows 3 and 7) under the NDCG measure. In particular, when testing on Twitter$_{11-NYT}$ and training on Twitter$_{12-TR}$, LTRS learns a better composite model than any of its individual features. This is a promising result, in that it shows that LTRS has the potential to produce better story rankings than the individual story rankers upon which it builds. However, when training on Twitter$_{12-NYT}$, NDCG

Top 5	Representation		Sub-feature	Document Ranking			Aggregation	Weight
	Story	Tweet		Ranking Depth	w	r		
1)	Headline	Tweet	Mentions	DFReeKLIM$_{1000}$	10 days	1 day	RWA	0.1720
2)	Headline	Twitterer-Feed	Relevance	DFReeKLIM$_{1000}$	2 days	1 day	RWAN	0.0860
3)	Headline	Tweet	Relevance	DFReeKLIM$_{1000}$	2 days	1 day	Votes	0.0860
4)	Headline	Tweet	Prominence	DFReeKLIM$_{1000}$	10 days	1 day	RWA	0.0860
5)	Headline	Tweet	Activity	DFReeKLIM$_{1000}$	10 days	1 day	RWA	0.0430

Table 6.13: Top 5 features (comprised of their four individual elements) selected by AFS on the Twitter$_{12-TR}$ dataset.

performance is lower than the best feature. Indeed, in this second case, performance is close to random. Furthermore, for the reverse case, i.e. where we test on Twitter$_{12-NYT}$ and train on Twitter$_{11-NYT}$, we again observe lower story ranking performance under NDCG. This indicates that the story ranking models produced by these two settings are quite different, and hence do not generalise. Finally, we see that when we test on Twitter$_{12-TR}$ and train on Twitter$_{11-NYT}$, LTRS provides equivalent performance to the best feature that it could have selected. To answer our third research question, we conclude that LTRS has the potential to produce better story ranking models than our unsupervised approaches. However, we do not see as large an increase in story ranking performance as when using LTRS as in the blog setting (see Section 6.6.3). This is likely due to insufficient training data from which to identify effective features, since our three twitter datasets cover a much shorter timeframe than the blog datasets (see Table 5.3). We would expect that training over story rankings from a longer time period will result in a more effective story ranking model using LTRS.

6.7.5 Influential Story Ranking Features

To examine which of our features were the most useful for news story ranking on tweets, we analyse the features that comprise our most generalisable story ranking model, i.e. the one that performed the most effectively when tested on another dataset, namely the model produced from the Twitter$_{12-TR}$ dataset. Table 6.13 lists the top 5 features selected by LTRS for the Twitter$_{12-TR}$ dataset. Recall that each feature is comprised of the four elements described in Table 6.3. All feature scores are normalised to between 0 and 1, while the AFS learning to rank approach (see Section 2.5.2) that we use combines features in a linear fashion. Hence, the feature weights that we report are indicative of their related feature's influence on the ranking.

From Table 6.13, we see the following trends regarding feature selection. Firstly, we see that the only story representation that was selected was the Headline. This shows that the other potential story representations that were used to produce features, most notably query expansion, does not appear to aid our story ranking models. Secondly, we observe that two features that use the document weighting

```
Thousands march in Mexico City against presumed presidential winner
@EPN - http://ow.ly/cq7ZH @youranonnews @RT_com @BBCWorld
```

Figure 6.3: Example tweet that would be promoted by the Mentions feature.

model (relevance) score are selected with equal weight, one that uses Votes upon the tweet ranking and one that uses RWAN. The latter was in-fact the most effective single feature on the dataset in question, hence it is interesting that it did not receive the highest weight. Indeed, the highest weighted feature does not consider the relevance scores assigned to each tweet, rather it leverages a Twitter specific sub-feature, i.e. whether or not each voting tweet mentioned another Twitter user. From further analysis, the reason that this feature was shown to be useful is that it allowed the learner to highly score tweets that mentioned a news provider. Figure 6.3 illustates a tweet that would be promoted by the Mentions feature.

This highlights how LTRS can combine different types of evidence specific to the user-generated content stream that it is using. Indeed, three of the five top selected features used Twitter specific sub-features, namely considering the most influential (retweeted) tweet in the ranking and the activity of the author of each tweet, in addition to mentions within the voting tweets.

In terms of the time windows that produced the most influential features, from Table 6.13, we see that the only very recent tweets ($r = 1$ day) were selected. Meanwhile, the background window sizes chosen were small for the Relevancy features ($w = 2$ days), while for the tweet-specific features a larger time window ($w = 10$ days) was selected. This indicates that comparing related tweets from today to yesterday's tweets provides a useful indicator of story importance. On the other hand, when considering features that are not based on relevance, it is more effective to compare related tweets from today to a longer background time period.

Overall, we have shown that LTRS has potential to learn an effective model for the automatic ranking of news stories by their importance, although there exists significant scope to improve identification of the top event of the moment. Furthermore, when leveraging tweets, we have shown that an effective such model combines tweet relevance information with Twitter specific sub-features in a way that is not possible for our unsupervised approaches, answering our fourth research question.

6.7.6 Event Promotion in Tweets

In Section 6.6.5, we examined whether the models learned by our LTRS framework tended to promote predictable or unpredictable events. In this section, we examine to what extent the model learned by our LTRS framework uses recent content to drive the story importance estimation. To test this, we

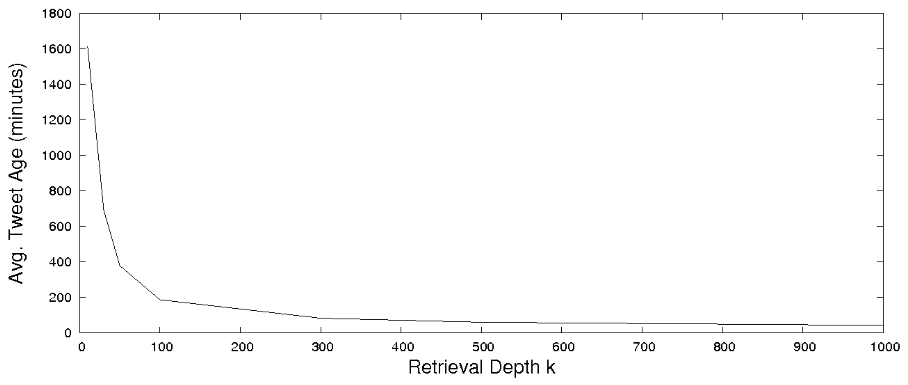


Figure 6.4: The average age of selected tweets as the retrieval depth is increased.

examine the recency of tweets that are used to build our features. We aim to determine whether our learned model is leveraging up-to-the-minute discussion. In particular, for the 45 homepages within the Twitter$_{12-NYT}$ dataset, we measure the mean age of the tweets retrieved for the stories contained within those homepages at different retrieval depths.

Figure 6.4 shows the dataset mean of the average tweet age in minutes when retrieving up to 1000 tweets. Importantly, earlier in Section 6.7.5 we showed that LTRS uses retrieval rankings to depth 1000. A low average tweet age at depth 1000 might indicate a focus on recent everging events, while a higher average tweet age at that same rank might indicate a focus longer running events (with a history of discussion).

From Figure 6.4, we observe that as retrieval rank increases, the more recent the average age of the tweets selected as evidence become. At a shallow retrieval depth of 10, the mean age is approximately 1,614 minutes, i.e. the tweets selected are predominantly from the previous day. On the other hand, at a depth of 1000 the mean age of tweets selected is only 42 minutes. The features selected by LTRS used rankings of 1000 tweets, hence the majority of tweets used will be recent. This would indicate that story rankings using evidence derived from Twitter might focus on recent events more than longer running ones.

6.7.7 Conclusions

In this section, we evaluated our unsupervised news story ranking approaches on a stream of Twitter tweets. We showed that these approaches can be effective at ranking news stories in a real-time manner when using recent discussion within Twitter to estimate news story importance in a real-time setting. However, in contrast to our results when applying our proposed unsupervised approaches on the blog stream, we observed that our approaches varied more widely in story ranking effectiveness between the three Twitter datasets used for evaluation and that leveraging older discussions using our $GaussBoost$ enhancement was not effective. On the other hand, leveraging the maximum observed importance for a story over a longer time period using our $MaxBurst$ enhancement was shown to significantly increase effectiveness. Furthermore, our experimental results also showed that combining these approaches under LTRS framework can generate a more effective model than our unsupervised approaches and also show that these models combine tweet relevance information with Twitter specific sub-features. We conclude that both our unsupervised approaches and LTRS framework appear to be effective and therefore have the potential to aid when satisfying news-related queries within a universal Web search engine. In the next section, we summarise the findings of this chapter as a whole.

6.8 Conclusions

In this chapter, we have investigated whether user-generated content can be used to identify the most important news stories of the moment. We proposed both learned and unsupervised news story ranking approaches that leverage user-generated content to test this. Across two evaluation settings covering both blog and tweet streams, we have evaluated our approaches to determine whether they are effective at identifying top news stories in real-time.

In particular, in Section 6.2, we proposed a novel adaptation of the Voting Model for automatic real-time news story ranking. This approach treats recent and related documents retrieved for an event (represented by a newswire article) from a user-generated content stream as votes for that event being important at that moment. We also proposed four extensions to this approach that take into account the degree of relatedness between the retrieved documents and each news story, the length of the news story representation, the distribution of related posts to each news story over time and the maximum importance observed, respectively.

In Section 6.3, we then proposed a new framework – referred to as Learning to Rank Stories (LTRS) – that uses training examples to learn effective news story ranking models. This new framework allows existing event ranking approaches to be expressed in terms of four elements, namely: the story representation; the document ranking approach; the document sub-feature; and the aggregation model. Combinations of these elements form features for real-time story ranking. Multiple effective story ranking approaches are expressed in this way and then combined to provide better estimations of the importance of each event.

To evaluate whether our proposed approaches were effective for the automatic identification important events in real-time using user-generated content, we examined their effectiveness when using both blog post and tweet streams. Section 6.4 defined our methodology for evaluating our approaches on two blog and three tweet datasets, while in Section 6.5, we specified the research questions that we examined in this Chapter.

In Section 6.6, we reported the news story ranking performance of our proposed approaches when using the two blog datasets ($BlogTrack_{2009}^{TopNews}$ and $BlogTrack_{2010}^{TopNews-Phase1}$). Our experimental results showed that these approaches can be effective at ranking news stories in a real-time manner when using recent discussion within blogs to estimate news story importance in comparison to state-of-the-art systems submitted to the TREC 2009 and 2010 Blog track top news stories identification tasks (see Table 6.4). In particular, of the approaches tested, we showed that relevance weighted aggregation (RWA) lead to the most effective news story rankings, indeed markedly outperforming the TREC best systems (see Table 6.4). Moreover, we showed that leveraging older discussions using our proposed $GaussBoost$ enhancement was able to increase story ranking effectiveness (see Table 6.5). Furthermore, through evaluation of our LTRS framework we have shown that it can generate significantly more effective models than our unsupervised approaches (see Table 6.6). Meanwhile, analysis of the features selected by LTRS indicate that those which consider only the last day's blog posts were the most effective (see Section 6.6.4).

Section 6.7 reported on the effectiveness of our proposed approaches over the three tweet datasets ($Twitter_{Dec2011}^{TopNews-NYT}$, $Twitter_{Jan2012}^{TopNews-NYT}$ and $Twitter_{Jan2012}^{TopNews-Reuters}$). Our results attested to the effectiveness of these approaches. In particular, all of our approaches outperformed a random event ranking baseline. Meanwhile, our RWA or RWAN approaches (see Section 6.2) that consider the relevance score of each tweet were shown to outperform our voting-based baseline by a statistically significant margin on one or more datasets (see Table 6.8). However, in contrast to our results when applying our proposed unsupervised approaches on the blog stream, we also observed that leveraging older discussions using our $GaussBoost$ enhancement was not effective when using tweets as opposed to blogs (see Table 6.9). On the other hand, leveraging the maximum observed importance for a story over a longer time period using our $MaxBurst$ enhancement was shown to significantly increase effectiveness (see Table 6.10). However, on the three datasets that use tweet streams, LTRS effectiveness varied depending upon the dataset used for training (see Section 6.7.4). As with the blog dataset, the most effective event ranking features selected by LTRS considered only recent documents from the 24 hour period prior to the ranking time (see Section 6.7.5).

Overall, our results showed that modelling event importance from user-generated content as a voting process is effective and that both blogs and tweets are good sources of evidence to use for real-time event ranking. Meanwhile, we conclude that our proposed LTRS framework is overall more effective than our voting-based approaches when estimating the importance of events in real-time using the blogosphere. This thesis postulated that the most important events at a point in time could be identified using user-generated content. From the experiments in this chapter, we have shown that this is indeed the case when using blogs and tweets.

In the next chapter (Chapter 7), we examine the second component of our news search framework – News Query Classification – that leverages the techniques described here to facilitate the accurate identification of news-related queries in real-time using user-generated content. Chapter 8 investigates the third component of our news search framework, i.e. Ranking News-Related Content, which generates rankings of news-related content for queries that are identified as news-related. Chapter 9 examines the final News-Related Content Integration component of our news search framework that combines news content into the Web search ranking for news-related queries – including the event rankings produced by the approaches described here.

Chapter 7

News Query Classification

7.1 Introduction

Within the News Search Framework described in Chapter 4, the second component determines whether the incoming user query has a news-related intent or not. If the query is determined to have a news-related intent, then relevant news content will be retrieved and subsequently merged into the final search results. If no news-intent is detected, then a traditional web-page ranked result list will be returned. In Section 4.4, we named this component News Query Classification (NQC). Effective news query classification is critical, since should the query-intent be wrongly classified, then either relevant news content will be omitted, or additional irrelevant news content will be added to the ranked results.

Importantly, we see a pivotal role for user-generated content within the news query classification process. Live streams of user-generated content provide up-to-the-minute discussion on topics of interest. Indeed, we discussed how users often publicly discuss current news as it happens and report breaking news in various user-generated content sources in Chapter 3. By monitoring user-generated content streams, it may be possible to identify emerging events – and hence user queries relating to them – faster than when using traditional newswire sources alone. Furthermore, by accounting for the discursive interest shown in social media for a particular newsworthy event, it may be possible to better estimate the background likelihood each query is about that event, especially in cases where the query alone is ambiguous.

In this chapter, we have three main aims, namely: to investigate novel approaches for achieving effective news query classification in a real-time setting; to determine whether user-generated content can aid the news search process by increasing news query classification accuracy; to examine whether user-generated content streams can be used to classify end-user queries in a more timely manner. The outline of this chapter is as follows:

- In Section 7.2, we describe a machine learning approach to news query classification that leverages both newswire and user-generated content streams to classify user queries regarding their news-related intent in a real-time manner. In particular, we define a framework for the extraction of news-related features from parallel content streams, enabling a model to be learned that can be used to classify incoming user queries.
- Section 7.3 describes our methodology for evaluating the proposed news query classification approach. In particular, we discuss the crawling and preparation of the news and user-generated content corpora from different time-frames that we subsequently extract features from and how we train our news query classifier.
- In Section 7.4, we summarise the features that we extract from each of our news and user-generated corpora that are used later under FANS for news query classification. Detailed descriptions of these features can be found in Appendix C.
- Section 7.5 lists the research questions investigated through experimentation in the following two sections.
- In Section 7.6, we experimentally evaluate our news query classification approach using our first experimental dataset from May 2006.
- Section 7.7 evaluates our news query classification approach using our second dataset from April 2012.
- In Section 7.8, we provide conclusions regarding the effectiveness of our proposed news query classification approach and whether the user-generated content can facilitate accurate real-time classification of user queries.

7.2 Feature Aggregation from News Streams (FANS)

To tackle the challenging task of identifying news-related queries in real-time, we propose a new classification approach that combines evidence from multiple news and user-generated sources. In particular, we propose to learn a classification model for distinguishing between news-related and non-news-related queries, which can then be deployed on a live stream. We use the incoming query to retrieve recent and related content from parallel news and user-generated content streams, from which we will derive features that either relate it to an ongoing news story or describe the current interest shown in the query topic. The classification model combines these features for a query into a probability score that it is

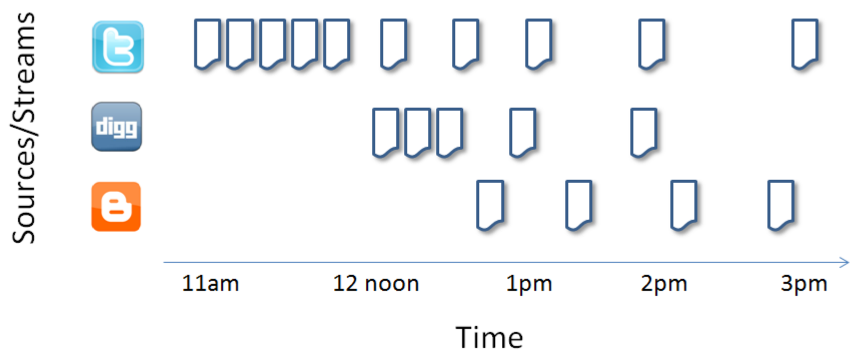


Figure 7.1: A hypothetical distribution of documents published over time that are about a newsworthy event in three streams.

a news-related query or not. We refer to this proposed approach as *Feature Aggregation from News Streams (FANS)*.

We begin by describing how FANS classifies queries over time. FANS classification has three stages:

- Features relating the query to be classified to recent publications in one or more document streams are extracted, referred to as *stream features*.
- To these initial features, additional *query-only features* are added.
- Finally, all of these features are expressed as a feature vector describing the query. A machine learned classification model uses the feature vector to estimate whether the query is news-related or not. For more information about machine learned classification approaches see Section 2.5.1.

Importantly, the stream features vary over time, as new documents are published within each source. This means that for a given query, the classification returned for that query can change over time, based upon whether the stream features indicate that there is a related event being currently discussed. To illustrate, Figure 7.1 shows a hypothetical distribution of documents published over a 5 hour period that are about a newsworthy event in three streams, namely: Twitter, Digg and the blogosphere. From Figure 7.1, we see that the distribution of documents is not the same across streams. In particular, at 11:30am, only relevant tweets have been published, while an hour later users start posting relevant documents to the Digg social news aggregator. Finally, around 12:45, the first blog post is published. Assume that FANS was to classify a query relating to the event that these documents discuss. If the query was submitted at 11:30am, then features extracted from Twitter would indicate that the query is news-related, while features tracking the Digg and blogosphere streams would not. Dependant upon how much weight the pre-prepared classification model assigns to Twitter-derived features, the query might not be considered to be news-related at this point. However, if the same query was submitted at 1pm, then features from all three of the sources will indicate that related discussions are being posted in each stream. Hence, at 1pm an effective classification model would therefore classify the query as news-related at that point in time.

In general, under our FANS approach, a query classifier learns a classification model comprised of multiple time-dependent features extracted from current newswire and user-generated content streams from the same time-frame, which are indicative of news-related queries. The classification model is then used to classify a test set of unseen queries (once those queries have had their associated features extracted), facilitating performance evaluation. When extracting features from multiple streams in a real-time setting, each feature can be expressed in terms of three components, namely: *the stream* from which it is extracted; *the time window* – representing the subset of the stream used to calculate the feature, and the *stream feature* – i.e. the notion is being measured from the stream. Hence, the score for one feature can be expressed formally as follows:

$$score_f(S, t_{start}, t_{end}) = \varpi(f, S_{t_{start} \rightarrow t_{end}}) \tag{7.1}$$

where f is the stream feature, S is the stream from which to extract the feature, t_{start} is the start of the time window and t_{end} is the end of the time window. $\varpi()$ calculates a score given a stream feature and stream (or subset of that stream). Figure 7.2 illustrates the feature score generation process when using three streams (BBC News Articles) , two time windows (1 hour and 24 hours preceding the time of the query) and two stream features (the DF and TF-IDF for the query in the stream) . Note that the number of possible features that can be generated is the multiplication of the three components. However, it is often not necessary to generate every feature combination because not every feature will provide unique and/or useful evidence for classification.

As we discussed earlier in Section 2.5, data-driven approaches like FANS are particularly advantageous when using features from multiple diverse corpora, because of their generality. In particular, unlike within a statistical text classification scenario (Nigam *et al.*, 2000), our features are not individual terms, but are rather dependant on recent historical news content provided by different newswire and user-generated content corpora. Data-driven learning provides a convenient framework by which multiple types of evidence from these corpora can be evaluated in terms of their effectiveness in distinguishing news-related queries. Furthermore, once a model has been learned for a particular set of features, that model can be used to classify unseen queries in real-time, subject to the extraction of those features for the new queries.

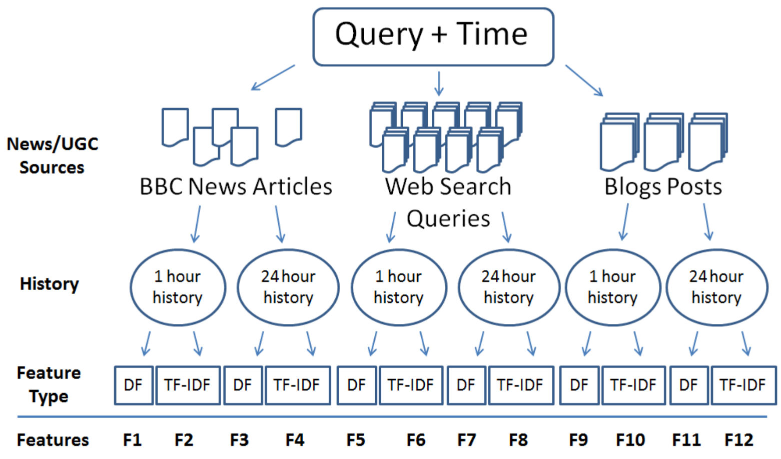


Figure 7.2: Illustration of feature generation for news query classification.

7.3 Experimental Methodology

In this section, we describe our experimental methodology for evaluating our proposed FANS approach to news query classification, as described previously (see Section 7.2). In particular, to evaluate FANS, we need both parallel news and user-generated content streams and both news-related and news-unrelated queries for FANS to classify. However, there exists no standard dataset of this form that we could use. Instead, as we described earlier in Section 5.2.2, we develop two new datasets from different time frames, namely $NQC_{May2006}$ that spans the month of May 2006 and $NQC_{Apr2012}$ that covers the period of the 11th to the 23rd of April 2012.

Recall from our dataset discussion in Section 5.2.2, that each news query classification dataset is comprised of three components. First, a series of corpora spanning a fixed time-frame that represent different content streams. The features that FANS uses to classify each query are extracted from these corpora. Secondly a set of queries that are to be classified. Each query has a timestamp from within the period of the dataset, corresponding to the time that query was made. The features from each corpora for a query are extracted with respect to that query's timestamp, i.e. only documents published before the timestamp are considered. Finally, for each query, a classification label is provided indicating whether that query was news-related or not for the time denoted by its timestamp.

Notably, May 2006 was chosen for our first dataset to match the time-period of the most recent available query log, i.e. $MSN_{May2006}$ (see Table 5.1). In contrast, April 2012 was chosen such that the Twitter and Digg user-generated content sources – that were not available in 2006 – could be used. A description of the corpora, queries and relevance assessments contained within each of these datasets

Streams	$NQC_{May2006}$	$NQC_{Apr2012}$
BBC Articles	✓	
Guardian Articles	✓	
Telegraph Articles	✓	
Blekko News Snippets		✓
Blekko News Articles		✓
Blog Posts	✓	✓
Blog Snippets		✓
Digg Snippets		✓
Digg Referred Articles		✓
MSN Queries	✓	
Tweets		✓
WikiNews Headlines	✓	
Wikipedia Updates	✓	✓

Table 7.1: News Streams available within each of the two news query classification datasets.

can be found in Section 5.2.2. Notably, the corpora used within each of the two news query classification datasets where collected in different manners due to the differing time-frames. In particular, $NQC_{May2006}$ was generated retroactively, by crawling archives of content from the time period of May 2006. Meanwhile, corpora for $NQC_{Apr2012}$ were live crawled. For this reason, the corpora contained within the two datasets are not identical. Table 7.1 provides a summary of the content streams that are available within each of the datasets.

To train the classification model used by FANS, we created a set of classification instances to train on (see Section 2.5.1). In particular, for a set of selected training queries, FANS extracts a fixed set of features about those queries using information from the streams within the dataset. Features for a query can only use documents from each stream that were published before the timestamp of that query, simulating a real-time streaming setting. Each of the features that we extract are detailed in the next section (Section 7.4). Each query, its features and the classification label (whether it is news-related or not) form a single classification instance. FANS builds a classification model by training upon a subset of all classification instances and then testing upon the other instances. Following standard practice in machine learning for classification, we use a 10-fold cross validation (Witten & Frank, 2005).

We measure the classification effectiveness of FANS in terms of the precision, recall and combined F_1 metrics described previously in Section 2.4.1.1. Furthermore, for the majority of the experiments, we down-sample our query dataset due to the high degree of class imbalance between news and non-news classes (Chawla *et al.*, 2004). This imbalance results from the naturally smaller ratio of news to non-news queries submitted to Web search engines (Bar-Ilan *et al.*, 2009). Specifically, on $NQC_{May2006}$, from the original 176 news and 2092 non-news queries, we randomly removed 1917 non-news queries resulting in an even split between news and non-news classes. On $NQC_{Apr2012}$, in a similar manner we randomly remove 2714 of the 2935 non news queries, leaving 221 non-news and 225 news queries. Performance is reported using the linear logistic regression trees (LogitBoost) classifier (Landwehr *et al.*, 2003) implemented within the Weka toolkit (Witten & Frank, 2005) to classify news queries. All of our features are normalised into the range [0,1] (based upon the maximum observed range of feature values) to enable comparison of those features in our later experiments.

We also evaluate how the evidence provided by each of our seven corpora changes over time. To do so, we create two additional simulated settings. In particular, in one simulated setting, each query in the evaluation dataset was 're-timed' to exactly 6 hours after the time recorded in the query log. In a second simulated setting, each query in the evaluation dataset was re-timed to exactly 24 hours after the time recorded in the query log. Importantly, for each of these two additional settings, all stream features are re-extracted for the re-timed queries, as new content will be available in each of the news and user-generated corpora at these later points in time.

7.4 News Query Classification Features

Under our FANS news query classification approach, we use features about each query to determine to what extent that query is news-related. Recall that in Section 7.2 we divided our features into two types, query-only features and stream features. Query-only features are derived from the query Q. In contrast, stream features are derived from a stream of content S bounded by a time window of size w that ends at time t for the query Q. Notably, some stream features can be applied to any news or user-generated content stream. An example of such a feature is the document frequency – the number of documents matching one or more query-terms. Naturally, this can be extracted from any stream of documents, i.e. by counting those documents from the stream and within the time window that contain each query term. Other features can be stream-specific. Stream specific features make use of some property of the stream that make it applicable to that stream only. For example, the Twitter stream might have a feature 'top tweet retweets' that counts the number of times the most relevant tweet to the query has been retweeted. Clearly, the number of retweets is applicable to only to the Twitter stream, hence is stream-specific.

We structure our features into six distinct feature sets, each encapsulating a different type of evidence. Query-only features form one feature set, while we separate stream features into five types, one of which represents our stream-specific features. Each feature set is described below:

- **Query-only features** encapsulate latent information from within the query. For example, a query-only feature might be the time the query was made, or whether it contains a celebrity name. These features are based upon the 'query-only' features proposed by König *et al.* (2009).

- **Frequency features** consider evidence from the usage of the query terms within each of the news and user-generated content streams prior to the query time. These features primarily measure how the usage of the query terms have changed over the short term. These features are an expansion of the 'corpus' features proposed by König *et al.* (2009).
- **Retrieval features** use the document weighting models described in Section 2.3 to find the most relevant documents from within each stream that were posted before the query time. The aggregate relevance of each set of documents retrieved for the query are used as features. These features are loosely related to those proposed by Diaz (2009), although we use document weighting models rather than counting matches and we employ them for sources other than query logs.
- **Burstiness features** use the burst detection algorithms described in Section 3.5.1 to identify query terms that are undergoing a burst in one or more of the document streams at the time of the query.
- **Importance features** measure the current importance of the query using the importance estimation approaches described previously in Chapter 6.
- **Stream-specific features** are similar to retrieval features, but rather than measuring the relevance of the top documents retrieved, we instead record stream-specific information about those documents.

Recall that we use two different datasets for evaluation in this chapter. These datasets contain different corpora. Hence, the features we extract from each similarly differ. In Table 7.2, we summarise all of the features – excepting the stream-specific features – extracted from the $NQC_{May2006}$ and $NQC_{Apr2012}$ datasets. Meanwhile, Table 7.3 summarises the remaining stream-specific features extracted from the two datasets. From Table 7.2, observe that we use 961 query-only and stream features from the $NQC_{May2006}$ dataset, while we use 755 query-only and stream features from the $NQC_{Apr2012}$ dataset. From Table 7.3, we see that there are no stream-specifc features from the corpora contained within the $NQC_{May2006}$ dataset, but 10 stream-specific features are extracted from the $NQC_{Apr2012}$ dataset. The lack of stream-specific features from the $NQC_{May2006}$ dataset is due to the 7 streams within that dataset providing only the text of each document.

A detailed description of all features used in this chapter can be found in Appendix C[1]. In the next section, we discuss the research questions that this chapter investigates. In Sections 7.6 and 7.7 we

[1]Author Note: Although these features and how they are extracted in a real-time streaming environment is a contriubution, we defer a detailed description of them here to avoid breaking the flow of the chapter.

$NQC_{May2006}$				
Source(s)	Feature Set	Feature	Description	# Features
Query	Query-only	# Tokens	Query length (tokens)	1
Query	Query-only	# Terms	Query length (terms)	1
Query	Query-only	# Entities	Number of named entities (Wikipedia)	1
Query	Query-only	# People	Number of person entities (Wikipedia)	1
Query	Query-only	# Places	Number of place entities (Wikipedia)	1
Query	Query-only	# Organisations	Number of named organisations (Wikipedia)	1
Query	Query-only	# Products	Number of product entities (Wikipedia)	1
Query	Query-only	Contains URL	Does the query contain a URL?	1
Query	Query-only	Contains 'news'	Does the query contain the term news?	1
7 streams	Frequency	TF_{stream}	Term frequency	14
7 streams	Frequency	DF_{stream}	Document frequency	14
7 streams	Frequency	$TF\text{-}IDF_{stream}$	Term frequency inverse document frequency	14
7 streams	Frequency	$DF\text{-}IDF_{stream}$	Document frequency inverse document frequency	14
7 streams	Retrieval	BM25	Sum of the scores for the top retrieved documents for the query using the BM25 document weighting model (Equation 2.4)	140
7 streams	Retrieval	DPH	Sum of the scores for the top retrieved documents for the query using the DPH document weighting model (Equation 2.9)	140
7 streams	Retrieval	LM_Dirichlet	Sum of the scores for the top retrieved documents for the query using the LM_Dirichlet document weighting model (Equation 2.13)	140
7 streams	Burstiness	# BurstyTerms	Number of bursty terms contained	224
7 streams	Burstiness	BurstMagnitude	Bursty term scores	224
7 streams	Burstiness	MultiResolution	Composite bursty term scores	28
Total				**961**

$NQC_{Apr2012}$				
Source(s)	Feature Set	Feature	Description	# Features
Query	Query-only	# Tokens	Query length (tokens)	1
Query	Query-only	# Terms	Query length (terms)	1
Query	Query-only	isMorning	6am to 1pm	1
Query	Query-only	isEvening	5pm to 11pm	1
Query	Query-only	isWeekend	Saturday or Sunday	1
Query	Query-only	isNightTime	11pm to 6am	1
Query	Query-only	Contains URL	Does the query contain a URL?	1
Query	Query-only	Contains 'news'	Does the query contain the term news?	1
Query	Query-only	Contains<entity>	Does the query contain a named entity. (AlchemyAPI)	173
7 Streams	Frequency	TF_{stream}	Stream Term frequency	14
7 Streams	Frequency	DF_{stream}	Stream Document frequency	14
7 Streams	Frequency	$TF\text{-}IDF_{stream}$	Stream Term frequency inverse document frequency	14
7 Streams	Frequency	$DF\text{-}IDF_{stream}$	Stream Document frequency inverse document frequency	14
8 streams	Retrieval	BM25	Sum of the scores for the top retrieved documents for the query using the BM25 document weighting model (Equation 2.4)	80
8 streams	Retrieval	DPH	Sum of the scores for the top retrieved documents for the query using the DPH document weighting model (Equation 2.9)	80
8 streams	Retrieval	LM_Dirichlet	Sum of the scores for the top retrieved documents for the query using the LM_Dirichlet document weighting model (Equation 2.13)	80
8 Streams	Importance	DPH+Votes	Number of recently retrieved documents for the query using the DPH document weighting model (Equation 2.9)	8
8 Streams	Importance	DPH+RWA	Sum of the scores for recently retrieved documents for the query using the DPH document weighting model (Equation 2.9)	136
8 Streams	Importance	DPH+RWA +GaussBoost	Sum of the scores for recently retrieved documents for the query normalised by their age	120
7 Streams	Burstiness	BurstMagnitude	Bursty term scores	14
Total				**755**

Table 7.2: All query-only and stream features extracted from the $NQC_{May2006}$ and $NQC_{Apr2012}$ datasets, excluding stream-specifc features.

$NQC_{May2006}$
No Stream-specific features

$NQC_{Apr2012}$				
Source(s)	Feature Set	Feature	Description	# Features
Digg	Retrieval	# Poster Profile Views	Sum of the digg view counts for diggs retrieved for the query	1
Digg	Retrieval	# Comments	Sum of the comment counts for diggs retrieved for the query	1
Digg	Retrieval	# Diggs	Sum of the digg counts for diggs retrieved for the query	1
Twitter	Retrieval	Retweet Count	Sum of the tweet retweet counts for tweets retrieved for the query	2
Twitter	Retrieval	User Statuses Count	Sum of the number of tweets made be authors of tweets retrieved for the query	1
Twitter	Retrieval	User Favourites Count	Sum of the number of tweets favourited be authors of tweets retrieved for the query	1
Twitter	Retrieval	User Friends Count	Sum of the number of friends that the authors of tweets retrieved for the query have	1
Twitter	Retrieval	User Followers Count	Sum of the number of followers that the authors of tweets retrieved for the query have	1
Twitter	Retrieval	User Listed Count	Sum of the number of user lists that the authors of tweets retrieved for the query appear in	1
Total				**10**

Table 7.3: All 10 corpus specific stream features extracted $NQC'_{2012}s$ Digg and Twitter streams.

report on whether using the features described here, when extracted from both news and user-generated content sources, leads to effective news query classification performance in comparison to a baseline classifier that leverages only features from newswire streams.

7.5 Research Questions

In the following two sections, we investigate whether user-generated content can aid the in the real-time classification of Web search queries as news-related or not. In particular, we aim to determine whether by adding features extracted user-generated content streams, we can more accurately classify news-related queries than baseline classification approaches. For this task, our baselines are a classifier that uses only newswire article streams to drive classification, and (for the $NQC_{May2006}$ dataset where query-logs are available) a classifier that uses recent search queries to drive classification, like that proposed by Diaz (2009).

As described in Section 7.3, we evaluate over two datasets, namely; $NQC_{May2006}$ and $NQC_{Apr2012}$. We evaluate each dataset in its own section. In Section 7.6, we evaluate using the $NQC_{May2006}$ dataset, while in Section 7.7 we use the $NQC_{Apr2012}$ dataset for evaluation. Within each of these two sections, we answer three research questions, namely:

1. Does the addition of features from user-generated content streams enable news-related queries to be more accurately classified than when using our baseline classifiers? (Sections 7.6.1 and 7.7.1)
2. Of the feature sets described in Section 7.4, which provide the most useful features and from which streams are do these come from? (Sections 7.6.2 and 7.7.2)

3. How is classification effectiveness under FANS impacted as stories mature over time, given the dynamic nature of the streams that drive it? (Sections 7.6.3 and 7.7.3)

In the context of the overall thesis, by evaluating whether the addition of evidence from user-generated content streams leads to increased classification accuracy, we can show that user-generated content can aid in satisfying news-related queries by more accurately identifying when news-related content should be integrated into the Web search results. Meanwhile, if the addition of user-generated content benefits the classification of breaking news queries, then this would also show that user-generated content can enable news-related queries to be satisfied in a more timely manner, i.e. by identifying new news-related queries faster than would be possible when relying on newswire reporting alone.

7.6 Evaluating News Query Classification on the May 2006 Dataset

In this section, we evaluate our FANS approach for news query classification on the $NQC_{May2006}$ dataset. As described in Section 5.2.2, this dataset contains three news and four user-generated content streams, in addition to queries manually labelled as news-related or not using crowdsourcing (see Section 5.5). Our evaluation methodology was described in Section 7.3. The aim is to determine whether under FANS, the addition of user-generated content can increase the accuracy of news query classification over our baseline classification approaches. In this case, our baselines are a classifier that uses only newswire streams to drive it and a classifier that uses past querying behaviour (Diaz, 2009).

Within each of the following three subsections, we address a single research question. In particular, in Section 7.6.1, we examine how accurately news and non-news queries can be classified in a real-time manner when using newswire content, and examine to what extent the addition of user-generated content improves classification performance. Section 7.6.2 provides a detailed analysis of each feature set on a per-stream basis, with a view to identifying the most informative feature set/stream combinations. Lastly, in Section 7.6.3, through a novel temporal evaluation, we investigate for what length of time user-generated content remains valuable for news query classification after a new news story is first broken.

Note that for display in the tables presented within this section, we shorten all stream names. For example, the $Blogs_{May2006}$ stream is simply referred to as Blogs.

7.6.1 Integrating User-generated Content

We begin by examining whether the addition of features extracted from user-generated content to a news query classifier can increase classification accuracy over the same classifier using only features

Streams(s)	Newswire	User-Generated	Precision	Recall	F_1
BBC	✔		0.586	0.584	0.580
Guardian	✔		0.519	0.507	0.471
Telegraph	✔		0.486	0.493	0.407
BBC+Guardian+Telegraph	✔		0.577	0.575	0.573
WikiNews		✔	0.493	0.493	0.490
Blogs		✔	0.631	0.615	0.603
WikiUpdates		✔	0.623	0.615	0.609
MSNLog		✔	0.648▲	0.637▲	0.631▲
WikiNews+Blogs+ WikiUpdates+MSNLog		✔	0.65▲	0.649▲	0.648▲
BBC+Guardian+Telegraph+WikiNews+ Blogs+WikiUpdates+MSNLog	✔	✔	**0.697▲**	**0.694▲**	**0.693▲**

Table 7.4: News query classification performance using multiple combinations of news and user-generated streams. Statistical significance (t-test $p < 0.05$) over the BBC+Guardian+Telegraph news baseline is denoted ▲.

from newswire streams. If classification performance is shown to be significantly increased either by the addition of features from a single user-generated content stream, or the combination of features from many such streams, then this would show that user-generated content can aid in satisfying news-related queries by increasing classification effectiveness. Table 7.4 reports classification accuracy in terms of precision, recall and combined F_1 metrics (see Section 2.4.1.1) when using features from each of the seven news and user-generated content streams both individually and in combination. Our baseline classifier representing classification using only newswire streams is BBC+Guardian+Telegraph (row 4). Statistical significance (t-test $p < 0.05$) over this baseline is denoted ▲.

From Table 7.4, we observe that classification using only features from the single Blogs, WikiUpdates or MSNLog user-generated content streams results in higher classification accuracy under all three measures than the baseline (comparing row 4 to row 5-8). This shows that single user-generated content sources can be more effective than newswire streams when classifying news-related queries. However, we also see that of these four sources, the MSNLog is the highest performing and was the only one to achieve statistical significance over the baseline. This indicates that of the sources tested, query logs are the most effective sources of evidence for news query classification. We also note that the MSNLog classifier uses (among others) features proposed by Diaz (2009) for the same task. Hence, we treat the classifier using features from the MSNLog as a second baseline.

Next, we see whether by combining features from multiple user-generated content streams we can enhance our MSNLog baseline. Comparing the MSNLog baseline (row 8) to the combination of features from all four user-generated sources (rows 9-10), we see that classification accuracy is further increased (0.631 F_1 to 0.648 F_1). This is a promising result, as it shows that the evidence from different user-

Query Features	Precision	Recall	F_1
Length metrics	0.489	0.49	0.485
Query composition	**0.714**	0.575	0.494
Named entities	0.600	**0.586**	**0.571**

Table 7.5: News query classification performance for each of the three query feature categories described in Appendix C.1.

generated content sources is additive, which in turn indicates that different streams cover different news events that users then search about. Hence, by combining features from different streams we are able to identify news-related queries that otherwise would have been missed.

Comparing the combination of user-generated content streams (rows 9-10) to the combination of all news and user-generated content streams (rows 11-12), we see that performance is further improved (0.648 F_1 to 0.693 F_1). This shows that the evidence provided by newswire streams is additive with the evidence from user-generated content streams.

Finally, if we compare the combination of newswire streams (row 4) to the combination of news and user-generated content streams (rows 11-12), we see that classification accuracy is increased by a large and statistically significant margin of 0.12 F_1. Hence, to answer our first research question, we conclude that the addition of features from user-generated content streams enables news-related queries to be more accurately classified than when using newswire classifiers alone. However, we also note that although the addition of user-generated content results in statistically significant increases over using newswire streams alone, the highest performance that we observe is 0.693 F_1. This indicates that, for the $NQC_{May2006}$ dataset, real-time news query classification is a hard task. This is intuitive, as we are trying to classify queries for which news content may not yet exist.

7.6.2 Feature Importance

Next, to answer our second research question, we investigate which of our four proposed feature sets are the most useful for news query classification. In particular, we compare the classification performance of each feature set on each stream individually, highlighting the trends observed. However, we first discuss the performance of the query feature set, as it is not derived to any stream, but rather generated from each query individually.

Table 7.5 reports query-only feature set performance for each of the three query-only feature categories (see Appendix C.1), i.e. length metrics, named entities and query composition, in terms of precision, recall and F_1 metrics. From Table 7.5, we observe two points of interest. First, we see that the query length and composition are not useful indicators of a query's news-relatedness, with classification accuracy of 0.485 F_1 and 0.494 F_1, respectively. Secondly, in contrast, we do observe a higher

	All Features			Frequency Features			Retrieval Features			Burstiness Features		
Stream(s)	Precision	Recall	F_1	Precision	Recall	F_1	Precision	Recall	F_1	Precision	Recall	F_1
BBC	0.586	0.584	0.580	0.586	0.584	0.580	0.489▼	0.49▼	0.485▼	0.631	0.532	0.421▼
Guardian	0.519	0.507	0.471	0.489	0.49	0.485	0.489	0.49	0.485	0.629	0.516	0.381▼
Telegraph	0.486	0.493	0.407	0.489	0.49	0.485	0.489	0.49	0.485	0.486	0.493	0.407
WikiNews	0.493	0.493	0.490	0.489	0.49	0.485	0.489	0.49	0.485	0.493	0.493	0.490
Blogs	0.631	0.615	0.603	0.678	0.646	0.629	0.548	0.547	0.545	0.474▼	0.487▼	0.406▼
MSNLog	0.648	0.637	0.631	0.66	0.581	0.522	0.559	0.555	0.548	0.476▼	0.479▼	0.461▼
WikiUpdates	0.623	0.615	0.609	0.633	0.612	0.597	0.626	0.626	0.626	0.487▼	0.487▼	0.487▼

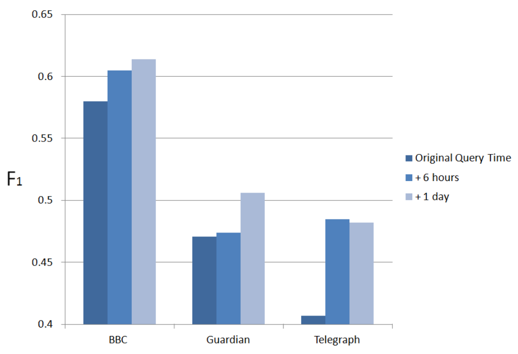
Table 7.6: News query classification performance using different feature set/streams combinations. Statistically significant (t-test $p < 0.05$) decreases in performance over using all features are denoted ▼.

classification accuracy when using only our named entity query-only features (0.571 F_1). This indicates that our named entity extraction features have some discriminative power when used in isolation.

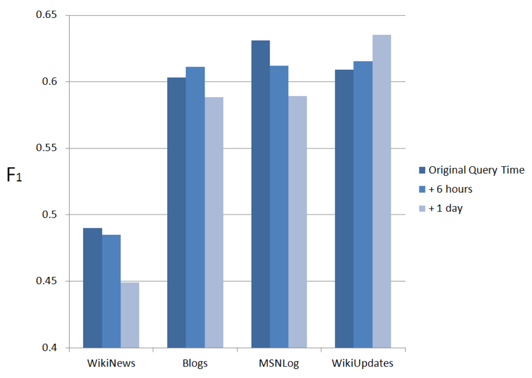
Table 7.6 reports the classification performance for each feature set/stream pair in terms of precision, recall and F_1. To provide a basis for comparison, Table 7.6 also reports classification performance when using all features. Statistically significant (t-test $p < 0.05$) decreases in performance against using all features from the same stream are denoted ▼. Feature groups that achieve significantly lower performance likely are not contributing to the combined classifier. In contrast, the feature sets that achieve the highest performance are likely to aid the combined classifier (we would not expect a single feature set to outperform the combination of all features however).

From Table 7.6, we observe the following. First, query frequency features appear to be some of the most effective, for instance frequency features on the BBC stream achieves 0.58 F_1, while the same features on the Blog stream achieves 0.629 F_1. Indeed, query frequency features on the Blogs stream, exhibiting the highest performance of any single feature set/stream combination. This indicates that frequency features are an important component of our news query classifier on the $NQC_{May2006}$ dataset.

Second, contrary to expectations, we observe that the retrieval features performed poorly on the news streams, achieving random a performance of 0.485 F_1. Indeed, for the BBC stream, the retrieval feature set was significantly worse than the combination of all features from that stream. This indicates that an insufficient proportion of the top documents returned for news queries were relevant. In hindsight, this is probably to be expected, as a news source may not publish more than one news article about each event over the period of a single day. If this is the case then retrieval features examining documents beyond rank 1 or 2 are unlikely to be useful. However, we also see that retrieval features were useful on the large Wikipedia updates stream, achieving 0.626 F_1. In this case, the ‘documents’ (short Wikipedia updates) being retrieved are extremely limited, i.e. only a few terms. The unexpectedly high performance is due to exact matching of the query to a related Wikipedia update.



(a) Newswire Streams



(b) User-Generated Content Streams

Figure 7.3: News query classification F_1 performance for three temporal settings using each news and user-generated content stream.

Notably, in line with results reported in Section 7.6.1, we observe that user-generated content provides higher overall performance than traditional news content. Specifically, the highest performing news stream is the BBC, with 0.580 F_1, while the highest performing user-generated content stream is Blogs with 0.629 F_1.

Finally, from Table 7.6, we observe that our burstiness features do not appear to aid classification on this dataset. Indeed, the use of burstiness features alone are often significantly worse than the combination of all features from the same stream. We believe that this can be attributed to the feature set's limited value to ongoing, rather than breaking queries. In their analysis of temporal aspects of querying in a Web search setting, Jones & Diaz (2007) indicated that news-related queries do under-go bursts. However, that burstiness can be short lived. Hence, burstiness features are only able to identify news-related queries that are currently experiencing a notable burst of activity in one or more streams, which may be gone mere minutes of hours later.

In answer to our second research question, we conclude that under this setting, query frequency and retrieval features are the most effective of those tested for to news query classification. In particular, query frequency features achieved similar levels of classification performance on both news and user-generated content streams to the use of all features. Meanwhile, retrieval features appear to be only useful on the large Wikipedia updates stream. On the other hand, the burstiness features tested were not effective on this dataset.

Time Point	All Streams		
	Precision	Recall	F_1
Original Query Time	0.697	0.694	0.693
+ 6 hours	0.661	0.660	0.660
+ 24 hours	0.595▼	0.595▼	0.595▼

Table 7.7: News query classification performance in terms of precision, recall and F_1 for three temporal settings using all streams. Statistically significant (t-test $p < 0.05$) decreases in performance over classification at the original query time are denoted ▼.

7.6.3 Timeliness

In line with our third research question, we examine how classification accuracy when using user-generated content changes as news stories mature. Over time, the classification of news queries may become easier as new content becomes available in each of the news and user-generated streams. In particular, we would expect classification performance to increase, as newswire streams in particular become more useful. To evaluate this, we measure the performance of our classifier at three points in time relative to the initial time each query was made, specifically, the original query time, six hours later and one day later. Figure 7.3 reports classification performance at these three relative time points in terms of F_1 for each of our seven news and user-generated content streams.

From Figure 7.3, we observe the following trends. First, from Figure 7.3 a), we see that in line with our expectations, newswire classification performance tends to increase over time. For instance, we see that for the BBC stream, classification accuracy is around 0.58 F_1 at the original query time. 6 hours later, this has increased to 0.61 F_1, while 24 hours later this has further increased to 0.63 F_1. Secondly, examining the user-generated content streams shown in Figure 7.3 b), we see a more mixed picture. For instance, classification accuracy for the WikiUpdates stream increases over time, but classification using the MSNLog stream decreases. This behaviour is to be expected, since querying activity about story will likely decrease over time as users lose interest, hence the value that the MSNLog brings will diminish. On the other hand, if Wikipedia lags news reporting in other sources, as was reported by Osborne *et al.* (2012), then the value of Wikipedia should increase as relevant updates are made to its pages.

Next, we examine how our news query classification accuracy when using features from all seven streams varies over time. Table 7.7 reports the classification performance in terms of precision, recall and F_1 for the three temporal settings when using all streams. From Table 7.7, we see that in contrast to our expectations, overall classification performance degrades over time. In particular, classification effectiveness using both news and user-generated content streams decreases by a statistically significant margin between the original query time and 24 hours later. Indeed, comparing Table 7.7 to Figure 7.3, we see that although classification accuracy using classifier combining all streams for the original query

Query Type	Original Query Time	+ 6 hours	+ 1 day
Generic	**70%**	65%	**70%**
Breaking	58%	**74%**	55%
Recent	57%	**61%**	40%
Long-Running	**75%**	66%	66%

Table 7.8: Percentage of the four query types that were classified correctly at the three points in time.

time exceeds that of any individual stream, at 24 hours later this is not the case. In particular, the combined classifier is outperformed (by a small margin) by classifiers driven by both the BBC and WikiUpdates streams. This indicates that the combined classifier is not making best use of the features available to it.

To examine this more closely, Table 7.8 reports the percentage of each query type that was classified correctly at the three points in time when using features from all seven streams. The aim is to see if particular types of news-related queries are disproportionally effected over time. From Table 7.8, we see the following. Firstly, that over the full 24 hour period, the proportion of Generic news queries classified correctly remains constant, while the remaining queries are all classified less accurately. However, while we observe that overall classification performance decreases, accuracy on the breaking news queries in particular markedly increases six hours after the original query was made (58% to 75%). This highlights how the influx of new news content can help classification of queries relating to breaking news.

To answer our third research question, when using query frequency, retrieval and burstiness features, as time progresses, news streams tend to provide better evidence for news query classification, as the delay in publication inherent within this type of news reporting is mitigated, whilst user-generated streams provide less evidence, as activity regarding current stories tails off. Moreover, we have shown that the number of breaking news queries classified markedly increases, although classification performance in general decreases.

7.7 Evaluating News Query Classification on the April 2012 Dataset

In the previous section, we evaluated our FANS approach for classifying user queries as news-related or not in real-time, using parallel news and user-generated corpora from the $NQC_{May2006}$ dataset. Our results showed that FANS could identify news-related queries and that the addition of user-generated content could increase classification accuracy over a news query classifier that used only news streams. However, the $NQC_{May2006}$ dataset used content streams from 2006. Hence, not only were more recent user-generated content sources not tested, but the value of user-generated sources may have grown with the rise in their popularity (Sirhan, 2011; Technorati, 2011).

In this section, we evaluate FANS when deployed on a modern collection of streams from the $NQC_{Apr2012}$ dataset that covers a period of 12 days in April 2012. In particular, this dataset contains newer Twitter and Digg streams that may provide timely evidence for news query classification. In each of the following subsections we investigate a specific research question regarding news query classification. We follow the same structure as our evaluation on the $NQC_{May2006}$ dataset (see Section 7.6). In particular, in Section 7.7.1, we compare the news query classification effectiveness of FANS when using news streams in contrast to user-generated streams. We ask whether features from user-generated content can classify end-user queries more effectively than features from newswire streams on the $NQC_{Apr2012}$ dataset. Section 7.7.2 examines the how effective each of the six feature sets described in Section 7.4 are on the $NQC_{Apr2012}$ dataset. We aim to determine which of the feature sets are the most useful for classification, both individually and when considered in conjunction with each of the eight streams within $NQC_{Apr2012}$. Finally, in Section 7.7.3, we investigate whether news query classification effectiveness improves over time as new evidence becomes available within each of the news and user-generated content sources tested.

Note that as before, we shorten stream names for easy display in our result tables. For example, the $NQC_{2012}^{BlekkoBlogSnippets}$ stream is simply referred to as BlekkoBlogSnippets.

7.7.1 Integrating User-generated Content

We begin by examining how effective news query classification is when using features extracted from each of the eight news and user-generated content streams from the $NQC_{Apr2012}$ dataset. In line with our thesis statement, by adding features extracted from timely streams of user-generated content, e.g. Twitter, we should be able to more effectively classify user queries as news-related or not than when using news streams alone. To this end, using FANS, we classify each of queries from the $NQC_{Apr2012}$ dataset as news-related or not and evaluate classification accuracy against their ground truth labels that state whether or not each is underpinned by a news-related information need. In particular, as our classification ground truth, we use a news query classifier that leverages only features extracted from newswire streams. We compare this to classifiers using features from user-generated content streams to see of classification accuracy increases or not.

Table 7.9 reports the news query classification performance of FANS when using all available features when extracted from different news and/or user-generated content streams. Query features, excepting the containsNews feature are omitted from this experiment. Recall that topics have been down-sampled to create a uniform class distribution, hence random effectiveness is 0.5 F_1. For comparison to our experiments on the $NQC_{May2006}$ dataset, the learner used to build the classification model was the

Logitboost classifier[1]. However, we examine other learners later in this section. Our baseline is the best performing classifier that leverages only newswire streams, i.e. BlekkoNewsSnippets (row 1). Statistical significance ($p < 0.05$) over the BlekkoNewsSnippets news baseline was tested, but no feature combination exceeded the 95% confidence threshold (the highest confidence observed was approximately 93% for the combination of all features under F_1).

From Table 7.9, we observe the following. Firstly, as with our previous experiments on $NQC_{May2006}$, FANS can effectively classify queries as news-related or not. Indeed, even when using only features from a single stream, e.g. blogs, a classification effectiveness of up to 0.7675 F_1 is observed. From this, we can conclude that FANS is an overall effective approach, as we have shown that it is able to distinguish news and non-news-related queries on two different datasets.

Secondly, if we compare classification performance when using features from either news or user-generated content streams, we see that FANS can be effective with either. For instance, using just our up-to-date aggregate news article stream (BlekkoNewsSnippets), we achieve classification performance of 0.7683 F_1, while using only features from the tweet stream, we achieve 0.7697 F_1. Furthermore, the most effective single stream under F_1 is DiggSnippets, a user-generated content stream, although our news article snippet stream provides similar performance. Indeed, it is interesting to note that Blekko's news article snippets, which are an aggregate of news articles from a large number of providers that Blekko follows, provide a more competitive performance to UGC streams than we observed when we compared news streams to UGC streams under our $NQC_{May2006}$ dataset. We believe that this due to a better coverage of news stories provided by Blekko in contrast to the three news providers available within the $NQC_{May2006}$ dataset. However, in line with our previous experiments on the $NQC_{May2006}$ dataset, we still see that UGC streams can outperform (albeit by a small margin) news streams, showing the value that user-generated content can bring to news query classification.

Next, recall that our news, blogs and digg streams have associated streams containing the documents that they link to. For example, for each news article snippet provided by Blekko, we have downloaded (where possible) the linked article for that snippet. We refer to the snippet stream as the source stream and the stream of linked documents as the referred stream. From Table 7.9, if we compare the news story ranking performance of each source stream to its associated referred stream, we see that in all three cases source stream features lead to more effective news query classification than features from the referred streams.

Next, we examine how news query classification effectiveness changes as we combine features from multiple streams together. In our previous experiments on the $NQC_{May2006}$ dataset we showed that

[1]We also tested other machine learning classifiers on this dataset and report their performance in Appendix D.

Feature Source(s)	Newswire	User-Generated	Precision	Recall	F_1
BlekkoNewsSnippets	✔		0.7831	0.7604	0.7683
BlekkoNewsArticles	✔		0.7155	0.6745	0.6898
BlekkoNewsSnippets+BlekkoNewsArticles	✔		0.7691	0.7468	0.7534
BlekkoBlogSnippets		✔	0.7742	0.7683	0.7675
BlekkoBlogsFull		✔	0.7101	0.7869	0.7424
DiggSnippets		✔	0.7624	0.7970	0.7756
DiggsFull		✔	0.7012	0.7563	0.7235
Tweets		✔	0.8094	0.7405	0.7697
WikiUpdates		✔	0.7938	0.6764	0.7248
BlekkoBlogSnippets+BlekkoBlogsFull+DiggSnippets+ DiggsFull+Tweets+WikiUpdates		✔	0.8284	0.7937	0.8072
BlekkoNewsSnippets+BlekkoNewsArticles+BlekkoBlogSnippets+ BlekkoBlogsFull+DiggSnippets+DiggsFull+Tweets+WikiUpdates	✔	✔	**0.8304**	**0.8032**	**0.8138**

Table 7.9: News query classification performance using multiple combinations of news and user-generated corpora.

classification accuracy increased markedly by combining features from different streams. Table 7.9 also reports classification effectiveness when combining features from news streams, UGC streams and all streams. From Table 7.9, we see that by combining our two news streams, i.e. features from BlekkoNewsSnippets and BlekkoNewsArticles together, classification performance decreases. This result is to be expected, as we showed above that linked article streams provide less evidence than snippet only streams. However, we see that both when combining features from many UGC streams, and when combining news and UGC streams together, classification effectiveness increases. In particular, our best UGC stream, i.e. DiggSnippets, provided 0.7756 F_1. By combining DiggSnippets with our other UGC streams, we observe an increase in classification effectiveness of 3.1% absolute under the combined F_1 measure. Moreover, by adding news streams in addition to UGC streams, effectiveness further increases by 1% absolute. This shows that by combining evidence from multiple streams (UGC or otherwise), we can better identify incoming news queries in real-time, also answering our first research question.

7.7.2 Feature Importance

Next, we examine the features that our FANS approach selects from each stream to aid news query classification. Notably, to enable a fine-grained analysis of individual features selected by our learner, we use an additional evaluation setting in this section. In particular, rather than performing a 10-fold cross validation, we instead split our queries into a 66% training set and a 33% testing set, while maintaining an even split of news and non-news queries for each set. By doing so, we can report the features that our learner selected on the 66% training set, and report performance on the 33% test set. Where this is used, we refer to it as *66/33 split*.

We begin by examining to what extent our query features (see Section C.1) – that are not linked to any stream – are also useful for classification. In particular, we consider three classification settings,

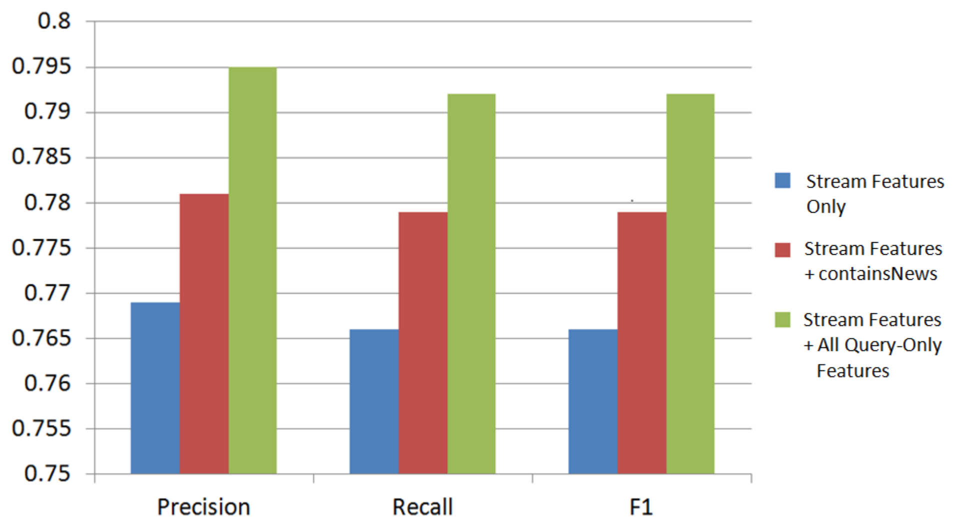


Figure 7.4: News query classification effectiveness when using different combinations of stream features query-only features (66/33 split).

namely: classification using only features from our eight news and user-generated corpora; these stream features plus the containsNews query feature; and the stream features plus all of the query-only features (including the containsNews feature). Figure 7.4 reports the news query classification performance of FANS for these three settings in terms of precision, recall and F_1 using the 66/33 split method. From Figure 7.4, we see that our query features are indeed contributing to the news query classification effectiveness. In particular, the containsNews feature alone increases classification accuracy from 0.766 F_1 to 0.779 F_1. By further adding our other query-only features that encapsulate information about the time each query was made and the entities that it contains, classification performance further increases to 0.792 F_1.

To examine this more closely, Table 7.10 reports the query features selected by the learner on the 66% training set under our third setting (Stream Features + All Query-only features). From Table 7.10, we make the following observations. First, as a sanity check, we see that the strongest positive feature, i.e. a feature that is indicative of news-relatedness, is the containsNews feature, as we would expect. Second, of the query time features selected, the learner selected isMorning as a negative feature, indicating that few news queries were made during 'morning' periods in our dataset (measured from UTC+0). Indeed, this translates into the early hours of the morning in the U.S., when new stories are less likely to be reported. Finally, we see that the learner identified a set of query-only entity features. These features act as positive or negative priors on the news-relatedness of that query. For instance, positive indicators

Positive/Negative	Query Feature	Weight
Positive	containsNews	2.6
	EntityProbability	1.35
	isOrganization	0.71
	isCelebrity	0.76
	isMusicalArtist	0.55
Negative	isProduct	-1.52
	isCompany	-1.48
	isCountry	-1.28
	isMorning	-1.26

Table 7.10: Positive and Negative query features selected on the 66% training set and their weights (66/33 split).

are celebrity or organisation names contained within the query, while countries and product names contained are negative indicators. We also see that just by having an entity (with high probability) in the query, increases the likelihood that the query is news-related (according to our learner).

Having examined the value of query features that do not leverage real-time information from news or user-generated content streams, we now evaluate the usefulness of our real-time stream features. In particular, we evaluate the classification performance of FANS when only subsets of our features are available. In this case, we break down our features into the feature sets described in Section 7.4 to see which of those add the most value. We test using these feature sets both when extracted from each stream individually, and when extracted over all eight streams in combination. For these experiments we return to a 10-fold cross validation evaluation strategy.

Table 7.11 reports the news query classification performance of FANS when using either frequency features; retrieval/stream-specific features; burstiness features; and importance features, from each stream. For easy display, we just refer to these as frequency, retrieval, burstiness and importance features respectively. Note that '—' indicate combinations which resulted in no features due to the large computational cost of calculating these features over high volume streams.

From Table 7.11 we make the following observations. Firstly, in terms of the four feature sets, we see that when using features from all streams, the retrieval and importance features are the most useful for news query classification, achieving 0.7684 F_1 and 0.7348 F_1 respectively. The query frequency features were less effective, with 0.7087 F_1, while the burstiness features are the least effective of the four feature sets, achieving 0.6090 F_1. These results support our earlier observations on the $NQC_{May2006}$ dataset, i.e that frequency and retrieval features are the useful for identifying news-related queries, while burstiness features are less so.

Examining the performance of our four feature sets on individual streams, we see that query frequency features were the most useful when extracted from the news snippet stream and the blog snippet stream. This indicates that the snippet streams provide useful term statistics when events break, i.e. unusual query term popularity may be easier to identify on the blog and news streams in comparison to more noisy streams. On the other hand, we note that the Digg snippet stream did not produce as effective frequency features as the blog and news streams. This in turn indicates that the Digg stream is noisier, possibly due to the ease that articles can be 'digged', making it more likely that important term frequency signals may be hidden in the background noise.

Our overall most effective feature set, i.e. retrieval features provided their highest performance when extracted from the Digg referred article stream (DiggFull). This contrasts with the general trend we see Table 7.11 of snippet streams outperforming their associated referred article streams. We can see that the reason for this is that the referred article streams achieved higher levels of recall than their snippet counterparts, e.g. 0.8150 recall on the Digg referred article stream.

Our additional feature set, importance features, that is based on the Votes and RWA models of (story) importance (see Chapter 6) that we repurpose for news query classification, is the second most effective individual feature set on $NQC_{Apr2012}$. We observe that these importance features have particularly low recall but equal or higher precision than the frequency and retrieval features. Indeed, on the WikiUpdates stream, importance features achieves a very high (0.8024) precision but a low (0.5551) recall. This is intuitive, as importance features are inherently 'fresh' in nature, i.e. they measure to what extent the topic of the query is important right now. As a result of this, they are useful to classify queries relating to breaking news stories, but not those relating to older stories. Indeed, we see the effect of this particularly for the news streams, where new documents about stories are not often posted after the initial 'wave' of reporting.

In summary, the overall most effective feature sets for news query classification were the retrieval and importance features, with frequency features adding value when extracted from the blog and news streams, answering our second research question. Moreover, we see that extracting the same features from many streams consistently increases classification accuracy over using features from a single stream. We have also shown that by combining timely features extracted from streams with query-only features, classification accuracy can be increased. Indeed, queries containing organisations and celebrities are apriori more likely to be news-related.

	Frequency Features			Retrieval Features			Burstiness Features			Importance Features		
Stream(s)	Precision	Recall	F_1	Precision	Recall	F_1	Precision	Recall	F_1	Precision	Recall	F_1
BlekkoNewsSnippets	**0.7020**	**0.6594**	**0.6757**	0.7024	0.7283	0.7113	0.6538	0.5812	**0.6114**	0.6822	0.5865	0.6265
BlekkoNewsArticles	0.6355	0.5360	0.5762	0.7103	0.6185	0.6564	0.6119	0.4591	0.5206	0.7276	0.5530	0.6234
BlekkoBlogSnippets	0.6759	0.6275	0.6468	0.6914	0.7570	0.7198	0.6455	0.5599	0.5951	0.7134	0.6338	0.6681
BlekkoBlogsFull	0.6900	0.5964	0.6355	0.6833	0.7850	0.7277	0.6270	0.5157	0.5614	0.6987	0.6638	0.6771
DiggSnippets	0.6594	0.6086	0.6284	0.6966	0.7645	0.7259	**0.6593**	0.4947	0.5608	0.6785	0.6421	0.6559
DiggFull	0.6398	0.5843	0.6062	0.6756	**0.8150**	**0.7363**	0.6285	**0.5982**	0.6077	0.6928	**0.7785**	**0.7303**
Tweets	0.5864	0.6023	0.5909	0.7308	0.6718	0.6950	—	—	—	0.7241	0.6777	0.6958
WikiUpdates	—	—	—	**0.7653**	0.6623	0.7042	—	—	—	**0.8024**	0.5551	0.6499
All	**0.7191**	**0.7055**	**0.7087**	**0.7953**	0.7489	**0.7684**	0.6469	0.5836	0.6090	0.7591	0.7191	**0.7348**

Table 7.11: News query classification performance using different feature set/streams combinations. The best performing feature combinations that use a single stream are highlighted in bold. Cases where combining streams results in increased performance over any single stream is also highlighted in bold.

7.7.3 Timeliness

Finally, we examine how classification accuracy of news queries changes over time. As time passes, classification performance may increase, as new evidence becomes available within each of the news and user-generated content streams considered. Recall that to evaluate this, we simulate querying for each topic within our dataset at later points in time, i.e. 6 hours after the query time and 1 day after the query time.

Table 7.12 reports the news query classification performance of FANS when classifying the queries in the $NQC_{Apr2012}$ dataset at the three points in time. Precision, recall and F_1 is reported as the average over 10-folds of cross validation. From Table 7.12, we observe the following. Firstly, if we compare news query classification performance using all features and streams (All), we see that classification accuracy increases by 5.14% absolute F_1 during the following 6 hours from the time of the original query. Hence, we can conclude that, indeed, the news query classification effectiveness of FANS increases over time. Moreover, as our query-dependant features do not change over time, this increase can be attributed to the real-time features extracted from the news and user-generated streams. However, we also see that overall, classification effectiveness does not continue to increase for the queries tested when we query 24 hours later. This either indicates that after approximately 6 hours have elapsed, little additional information becomes available in our news and user-generated streams, or that the value of some streams diminish over time, e.g. the DiggFull stream.

To examine this more closely, Figure 7.5 illustrates the change in F_1 for each of the streams at our three time points. From Figure 7.5, we see that each of our streams reacts differently over time. For instance, features from our news snippet stream are more useful 6 hours after the query time, while features from the referred article news stream become less effective over time. Indeed, we generally observe that the F_1 classification effectiveness of the referred article streams (BlekkoNewsArticles, BlekkoBlogsFull and DiggFull) diminishes over time. In contrast, user-generated tweet stream appears

	Original Query Time			+ 6 hours			+ 1 day		
Stream(s)	Precision	Recall	F_1	Precision	Recall	F_1	Precision	Recall	F_1
BlekkoNewsSnippets	0.7831	0.7604	0.7683	0.7990	0.7960	**0.7938**	0.8008	0.7844	0.7897
BlekkoNewsArticles	0.7101	0.7869	0.7424	0.7560	0.6998	0.7228	0.6873	0.6928	0.6857
BlekkoBlogSnippets	0.7742	0.7683	0.7675	0.7953	0.7943	0.7911	0.7724	0.7591	0.7609
BlekkoBlogsFull	0.7101	0.7869	0.7424	0.7179	**0.7994**	0.7528	0.7054	0.7412	0.7184
DiggSnippets	0.7624	**0.7970**	**0.7756**	0.7566	0.7685	0.7587	0.8054	**0.8317**	**0.8159**
DiggFull	0.7012	0.7563	0.7235	0.7433	0.7519	0.7444	0.6918	0.7032	0.6925
Tweets	**0.8094**	0.7405	0.7697	**0.8241**	0.7524	0.7822	**0.8257**	0.8007	0.8101
WikiUpdates	0.7938	0.6764	0.7248	0.8194	0.6525	0.7205	0.8024	0.6826	0.7332
All	**0.8284**	**0.7937**	**0.8072**	**0.8719▲**	**0.8505▲**	**0.8586▲**	**0.8595▲**	**0.8546▲**	**0.8542▲**

Table 7.12: News query classification performance in terms of precision, recall and F_1 for three temporal settings using each news and user-generated content stream. The best performing feature combinations that use a single stream are highlighted in bold. Cases where combining streams results in increased performance over any single stream is also highlighted in bold. Statistically significant increases (t-test $p<0.05$) over the same features extracted from the query time are denoted **▲**.

to become incrementally more useful as time progresses. This result is similar to that observed in the $NQC_{May2006}$ dataset, i.e. that different user-generated content streams behave differently over time. This highlights the differences between user-generated content sources, showing that some streams are useful early in a stories life-time, while others lag behind.

Overall, in answer to our third research question, our results indicate that the features extracted from certain news and user-generated content streams enable more effective news query classification as time progresses, e.g. Twitter tweets. However, not all streams do so, e.g. the blog snippet stream. On the other hand, combining features from multiple streams leads to consistently higher classification accuracy over time than any of our steams alone. This shows that combining different streams leads to a robust classifier that can effectively identify news-related queries for both breaking news stories and for more long running stories. Indeed, our classifier better maintained its effectiveness over time than when tested on the $NQC_{May2006}$ dataset.

7.8 Conclusions

In this chapter, we have examined the task of classifying incoming user queries as news related or not in real-time using user-generated content. In Section 7.2, proposed a novel machine learning approach, FANS, which considers different news and user-generated content sources to be document streams from which classification evidence can be drawn. The intuition is that by monitoring multiple streams for information about emerging events, and by cross-referencing each query against this information, we can better estimate when the information need underling a query is news-related. Moreover, by monitoring multiple streams and combining the evidence from each, we aim to increase classification accuracy, both

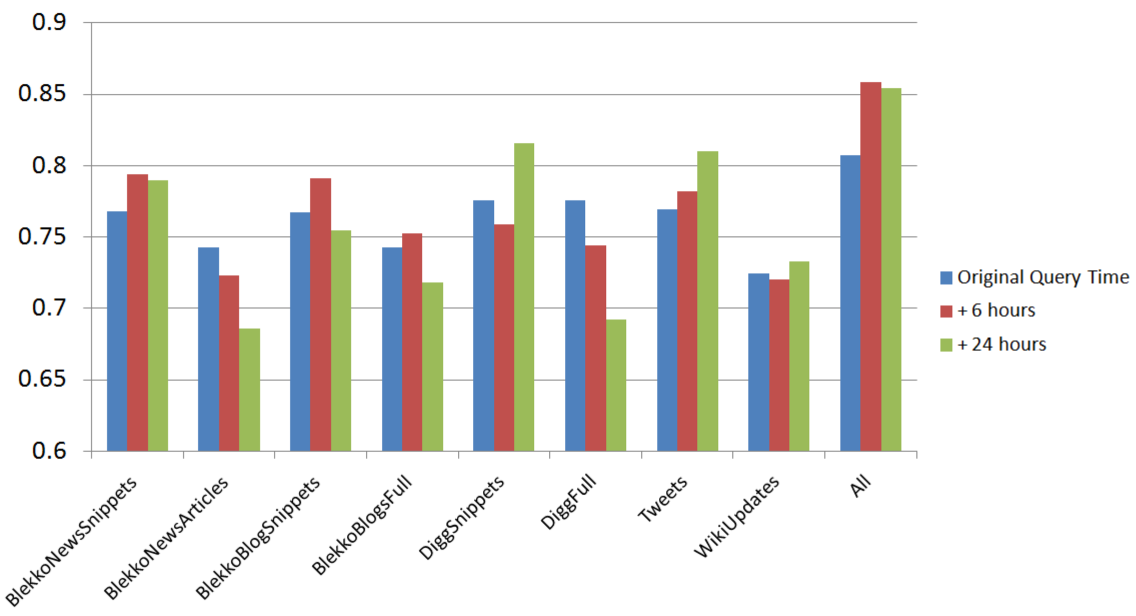


Figure 7.5: News query classification F_1 when using features derived from each stream over time.

by leveraging chorus effect from many different content sources and by enabling the faster identification of news-related queries relating to breaking news stories.

To achieve effective classification using FANS, in Section 7.4, we proposed a wide variety of features extracted from the different streams about each user query, each encapsulating a different type of evidence. For a query, these features are uniformly extracted from each the news and user-generated content streams, representing our current knowledge about both ongoing and emerging events that might be related to that query. FANS uses state-of-the-art machine learning techniques to discover those features that can distinguish between news and non-news queries.

Through extensive experimentation on two news query classification datasets ($NQC_{May2006}$ and $NQC_{Apr2012}$) in Sections 7.6 and 7.7, we investigated whether user-generated content can aid the classification of user queries by increasing classification accuracy. Our results showed that classifiers using only features extracted from user-generated content could outperform classifiers using only features from newswire streams on both datasets (see Table 7.4 and Table 7.9), showing that user-generated content streams can be used to more accurately classify news queries than newswire streams.

One motivation for using different types of news and user-generated content streams is that they might provide complementary evidence. To test this, we examined how news query classification accuracy was effected as we combine features from multiple streams. Our experimental results on both datasets were positive, with classification accuracy increasing over single stream classifiers in all cases (see Table 7.4 and Table 7.9).

We also analysed the types of features that were effective for news query classification. Over both datasets, query frequency and retrieval features that track mentions of the query terms over time were shown to be effective (see Table 7.6 and Table 7.11). Furthermore, mentions of a named entity an end-user query was shown to be a positive indicator that the query might be news-related. (see Table 7.5 and Table 7.10).

Some user-generated content streams, such as Twitter, are known for their real-time reporting and discussion of news-related topics. As a result of this, one advantage of leveraging user-generated content streams might be to identify news-related queries faster than using newswire streams. We analysed how classification accuracy using different news and user-generated content streams changed as stories matured. We showed that when the queries in our dataset were first issued, they could be more accurately classified using user-generated query logs than newswire streams. However, as time passed, classification accuracy using newswire increased, while classification accuracy using query logs decreased (see Figure 7.3). Indicating that the value of query logs comes early in a news story's lifetime. Furthermore, on the second dataset, we observed that our combined classifier was able to consistently outperform single stream classifiers both when the queries in that dataset were first issued and over time (see Figure 7.5). Indeed, classification accuracy overall increased by a statistically significant margin over the 24 hour period tested (see Table 7.12).

In our thesis statement (see Section 1.3), we hypothesised that user-generated content could increase the accuracy of a real-time news query classifier within a universal Web search engine. Based upon our experiments in this chapter, we conclude that user-generated content streams in combination can significantly improve a news query classifier that relies only upon newswire providers to identify news-related queries. This, in turn, enables a universal Web search engine that supports a news vertical to better satisfy news-related queries relating to both breaking and long-running events by classifying them more accurately.

In the proceeding chapters, we examine how to rank content for a news-related user query and then integrate that ranked content into the Web search results. In particular, in Chapter 8, we examine how to rank news-related content for a user query. Meanwhile, in Chapter 9 we perform a crowdsourced user study to investigate whether end-users find that the integration of user-generated content increases coverage for news-related user queries.

Chapter 8

Ranking News Content

8.1 Introduction

This chapter is concerned with the *Ranking News-Related Content* (RNRC) component of our news search framework (see Section 4.7). In particular, for queries that are classified as news-related, the RNRC component selects news-related content to be added to the Web search results for these queries. From Section 4.4, recall that this can be seen as a series of ranking tasks. Each task ranks documents from a single source of news-related content for the user query. The top ranked of this content from one or more sources may be integrated into the Web search ranking.

We formalised these ranking tasks in Equation 4.3 as a ranking function R. R takes as input a query Q, a point in time t and a content source S. R ranks the documents in S for the query Q that were published at or before time t. In general, the aim of this chapter is to develop ranking functions R for the content sources that we later integrate into the Web search ranking.

There are multiple content sources that the RNRC component might rank documents from. For instance, we illustrated in Section 1.2 how universal Web search engines integrate newswire articles into their Web search rankings for queries identified as news-related. Furthermore, we have motivated why user-generated content might be useful to return for some news-related queries (see Section 1.2). Hence, in this thesis, we examine the integration of both newswire and user-generated content into the Web search results. In particular, we use five different newswire and user-generated content sources namely: newswire articles; the blogosphere; Twitter; Wikipedia; and Digg. These sources were discussed earlier in Chapter 3.

However, we do not need to develop new ranking functions for all five of these sources, since document weighting models — such as those described in Section 2.3 — are already known to be effective for some sources. Indeed, in Section 4.7, we discussed why newswire articles, Wikipedia pages and Digg posts may be effectively ranked using document weighting models. Hence, the goal of this chapter is to develop and evaluate effective ranking functions for the remaining two of our sources, i.e. the blogosphere and Twitter. More specifically, we develop ranking functions that take into the account unique characteristics of these sources to improve retrieval effectiveness in comparison to traditional document weighting models.

This chapter plays two roles within the context of the overall thesis. First, our end-goal is to determine whether by adding user-generated content into the Web search ranking we can increase coverage for some news-related user queries. To do so, we need to have effective methods to rank documents from each of the five sources that we use, otherwise, we will have only irrelevant content to integrate into the Web search ranking. Second, as we discussed in Section 4.7, how to effectively rank user-generated content is not a solved problem. This chapter contributes novel learning to rank approaches for ranking blog posts and tweets and evaluates these approaches on standard TREC datasets. The remainder of this chapter is structured as follows:

- In Section 8.2, we propose learning to rank approaches to rank both blog posts and tweets in a news vertical search setting and describe the features that we use.
- Section 8.3 describes our experimental methodology, including the datasets that we use to evaluate the tweet and blog post rankings produced by our proposed learning to rank approaches.
- In Section 8.4, we list the research questions that we investigate in this chapter.
- Section 8.5 evaluates our proposed learning to rank approach for blog post ranking.
- In Section 8.6, we investigate the effectiveness of our learning to rank approach for tweet ranking.
- We summarise the findings of this chapter and provide conclusions in Section 8.7.

8.2 Learning to Rank User-Generated Content

In this section, we propose learning to rank approaches (see Section 2.5.2) to generate blog post and tweet rankings that we can subsequently integrate into the Web search ranking for display to the user. In particular, our learning to rank approach for blog post ranking is described in Section 8.2.1, while our tweet ranking variant is described in Section 8.2.2.

8.2.1 Learning to Rank Blog Posts

We propose a learned approach for ranking blog posts for a user query at a point in time. Formally, we aim to define a ranking function R that takes as input a query Q, a time t and ranks the blog posts contained within a source S^{Blogs} published before time t for the query:

$$R(Q, t, S^{Blogs}) = \{b \in S^{Blogs}_{t-w \to t} : score(Q, b)\} \tag{8.1}$$

where $S^{Blogs}_{t-w \to t}$ is the set of blog posts published within S^{Blogs} before time t, but after time $t - w$, b is a blog post in $S^{Blogs}_{t-w \to t}$ and $score(Q, b)$ calculates a score for blog post b given a query Q. w defines the window of time before t that we consider blog posts from. Note that blog post ranking is a real-time task, where we can only use evidence from before time t when ranking.

Under our learning to rank approach, we define a set of *blog post features* to describe each blog post. These features can be dependant upon the user query, e.g. the estimated relevance of the blog post to the user query as measured by a document weighting model (see Section 2.3). Features may also be query independent, e.g. the estimated opinionatedness of the blog post using a opinion detection tool. Our learning to rank approach builds a *blog post ranking model*, which combines the features of a blog post into a single score for that post. In effect, the model produced acts as the scoring function $score(Q, b)$ in Equation 8.1.

Our learning to rank approach can be considered to be comprised of four parts:

- **Blog post training topics**: A set of queries with blog post relevance assessments.
- **Blog post sampling technique**: A methodology for producing the initial ranking of blog posts for the user query that will subsequently be re-ranked.
- **Blog post features**: A vector of numbers, each derived from the blog post (and possibly the query).
- **Learning to rank algorithm**: A machine learning algorithm that learns the blog post ranking model from the training data and features (see Section 2.5.2).

These four components are used to build a blog post ranking model as follows. First, for each query in our blog post training topics, we use a blog post sampling technique to generate the *blog post sample*. The blog post sampling technique that we use here is to take the top ranked 1000 blog posts produced by the DPH document weighting model (see Equation 2.9). It is important to note that our learning to rank approach re-ranks this sample, rather than ranking all of the blog posts that match one or more of the query terms (which may be many times larger). For each blog post in each of the samples, that post has its *features* extracted, forming a feature vector describing it. The blog post ranking model is built by the *learning to rank algorithm*, using the feature vectors for each sampled blog post and those post's relevance assessments (where available). The exact means for producing the model is dependant upon the learning to rank algorithm employed. We use the Automatic Feature Selection (AFS) (Metzler, 2007) list-wise learning to rank approach. This learner produces the ranking model $score(Q, b)$ by finding the weighted linear combination of the blog post features that results in the best ranking performance over all of the training queries. Hence, we can express our model $score(Q, b)$ as follows:

$$score(Q, b) = \sum_{0<i<|F^b|} weight(i) \cdot F_i^b \tag{8.2}$$

where Q is the query, b is the blog post to be scored, F^b is the set of features extracted from b, $|F^b|$ is the number of features, F_i^b is the ith feature and $weight(i)$ is the learned weight for feature F_i^b. The overall ranking model R is therefore defined as:

$$R(Q, t, S^{Blogs}) = \left\{ b \in DPH(S^{Blogs}_{t-w\rightarrow t}, Q) : \sum_{0<i<|F^b|} weight(i) \cdot F_i^b \right\} \tag{8.3}$$

where $DPH(S^{Blogs}_{t-w\rightarrow t}, Q)$ returns the blog post sample using the DPH weighting model.

We use 78 blog post features (F^b) in our later experiments. We divide our features into five different feature sets, namely: Relevance, Dictionary Aspects, Part of Speech, Structure and Related Article. Table 8.1 lists each of our feature sets and how many features are contained within each. We summarise each feature set below:

Relevance: Our first feature set describes the relatedness between a blog post and the user query. In particular, this feature set uses the DPH document weighting model (see Equation 2.9) to score each blog post for the user query. In a real-time setting, the document weighting model uses the background statistics of the source within a time window, $S^{Blogs}_{t-w\rightarrow t}$. The parameter w defines how much of the stream history to use during this calculation, i.e. the window of time considered. We use two settings for w, 1 day and 7 days, forming two features.

In Section 2.3.5, we noted that query expansion techniques can be used to improve retrieval performance in a variety of settings. For instance, Arguello *et al.* (2008) reported that query expansion using an external Wikipedia corpus aided in a blog recommendation setting. Query expansion when using an external corpus, rather than the corpus that is being searched, is referred to as collection enrichment (see Section 2.3.5). We use collection enrichment to expand the user query using multiple external corpora. Each expanded query will result in a different blog post score. We use each of these scores as features.

Feature Set	Feature	Description	#
Relevance	DPH$_{t-1day}$	The relevance of the post when considering only the last day of posts for the purposes of the collection statistics	1
	DPH$_{t-7days}$	The relevance of the post when considering the last 7 days of posts for the purposes of the collection statistics	1
	DPH$_{t-1day}$+CE	Expanded query representations using collection enrichment with news articles and blog posts published from before the time of the query	3
Dictionary Aspects	Opinionated	Number of opinionated terms the post contains	4
	Factual	Number of factual content indicator terms the post contains	4
	Indepth	Number of terms indicating an indepth post	4
	Official	Number of terms indicating the post is from an official source	4
	Personal	Number of terms indicating the blog is not run by an organisation	4
	Shallow	Number of terms indicating that the discussion in the post is shallow	4
Part of Speech	Term Type Counts	Number of part of speech terms contained	30
Related Article	DPH$_{t-1day}$+Article	The relevance of the post to the related newswire article when considering only the last day of posts for the purposes of the collection statistics	1
	DPH$_{t-1day}$+QE (Article)	Expanded query representation using the related news article	1
Structure	Links	The number of in-links and out-links contained	4
	Ads	Does the post contain advertising	2
	Length	The length of the post under three metrics	3
	Content	Does the post contain images, video, comments, different entity types or offensive words?	8
Total			78

Table 8.1: Blog post ranking features.

The corpora that we use for collection enrichment are the *NYT08* and *TRC2* newswire corpora, in addition to the *Blogs08* blog post corpus. Statistics of these corpora can be found in Table 5.1. The aim is to expand the query with synonyms for popular terms that appear both within the query and these corpora. Notably, only documents published before the time of the first topic are used for collection enrichment to maintain a real-time setting. The DPH scores for the three expanded queries form three features in this feature set. The feature set in total contains 5 features.

Dictionary Aspects: Our second blog post feature set describes the characteristics of each blog post. In particular, we use a blog post taxonomy from the literature (Macdonald, Soboroff & Ounis, 2009) that defines six blog post aspects, namely: Opinionated; Factual; Indepth; Shallow; Official; and Personal. For each of these aspects, we use a dictionary of terms relating to that aspect, e.g. a dictionary of terms that are indicative of opinionated blog posts (McCreadie, Macdonald, Ounis, Peng & Santos, 2009). We compare a blog post to each dictionary to form features. In particular, inspired by the prior work of He, Macdonald, He & Ounis (2008), we use each dictionary as a query, scoring the terms in each blog post using the Bo1 or KL term weighting models (Equations 2.14 and 2.17). Here, the usage of each term in the blog post is compared against the background distribution over all blog posts within the time window. The score produced for a blog post acts as a feature describing that post. In this way, we describe how similar the blog post is to each dictionary and hence, the aspects that those dictionaries represent. Furthermore, we use two different versions of each dictionary. First, we use all of the dictionary terms as the query. Second, we prune the dictionary, removing very common and very rare terms according to the blog post source $S^{Blogs}_{t-w\rightarrow t}$. Hence, for each of the six aspects, we generate four features, each using either Bo1 or KL, and either the pruned or unpruned dictionary. The full feature set contains 24 features.

Part of Speech: Our third set of blog post features describe the part of speech (POS) terms contained within each blog post. We use the LingPipe[1] part of speech tagger to identify 30 POS classes within each blog post. We use the number of terms belonging to each POS class as features, one feature per class. Hence the feature set contains 30 features.

Related Article: Our fourth feature-set compares each blog post to a related newswire article for the query, in a similar manner to our Relevance feature set. Within a news vertical setting, such an article can be obtained by retrieving the top news article for the query from one or more news providers. In our experiments, these articles are pre-provided by the dataset used (see Section 8.3.1). We use the DPH

[1]`http://alias-i.com/lingpipe`

document weighting model to score the blog post using the newswire article content as the query. We also perform query expansion using the most informative terms from the newswire article as identified by the Bo1 term weighting model. Hence, this feature set contains two features.

Structure: The final feature set that we use describes the structure of each blog post. In particular, we sub-divide this feature set into four categories, namely: Links, Ads, Length and Content. The links category contains features that describe the number of in-links and out-links that the blog post has. We have two methods for counting both in-links and out-links. The Ads category contains features that estimate whether a blog post contains any advertising. The Length category provides three features that describe the length of the blog post, namely: the number of unique terms contained; the number of tokens total; and the number of sentences contained. Finally, the Content category describes other types of content present within the blog post. We identify images, video, comments, entities and offensive words. The total number of features within the Structure feature set is 17 features.

In total, we use 78 blog post features to train our blog post ranking model. We describe how this model is trained in Section 8.3 and evaluate its effectiveness in Section 8.5.

8.2.2 Learning to Rank Tweets

In this section, we propose a learning to rank approach for ranking Twitter tweets for a user query at a point in time. As with blog post ranking, we aim to define a ranking function R, which takes as input a query Q, a time t and ranks documents within a source S. However, this time, the documents to be ranked are tweets. We denote the tweet source as S^{Tweets}. R is calculated as follows:

$$R(Q, t, S^{Tweets}) = \{d \in S^{Tweets}_{t-w\rightarrow t} : score(Q, d)\} \tag{8.4}$$

where $S^{Tweets}_{t-w\rightarrow t}$ is the set of tweets posted within S^{Tweets} before time t, d is a tweet in $S^{Tweets}_{t-w\rightarrow t}$ and $score(Q, d)$ calculates a score for tweet d for query Q. As before, w defines the window of time before t that we consider tweets from.

Recall that in Section 3.4.1, we discussed how tweets have a variety of characteristics, e.g. a use of hashtags or mentions. Meanwhile, in Section 4.7, we discussed how these characteristics may impact their relevance and quality to a user query and how they are not accounted for by traditional document weighting models that only consider the tweet as a bag of words. Inspired by prior works in tweet ranking (see Section 3.4.3), we propose to use learning to rank to generate a tweet ranking model that incorporates these characteristics with the aim of increasing retrieval effectiveness. Here, the idea is to represent each tweet as a vector of features. In this way, the unique characteristics of the tweet can be encoded in a form that is suitable for use when ranking. For example, whether the tweet contains a URL can be seen as a feature.

As before, our learning to rank approach can be considered to be comprised of four parts, namely: tweet training topics; a tweet sampling technique; tweet features; and a learning to rank algorithm. In particular, we first, define a set of *tweet training topics*, comprised of queries and tweets ranked for those queries with associated relevance assessments. Next, a *tweet sampling technique* is used to create an initial ranking of tweets for each query in the tweet training topics, referred to as the sample. In this case, we use the DFReeKLIM (Equation 2.10) document weighting model to produce each sample, denoted $DFReeKLIM(S^{Blogs}_{t-w\rightarrow t}, Q)$. For a tweet d in the sample for a query Q, *tweet features* F^d are extracted about that tweet. A *learning to rank algorithm* takes as input each tweet's features and its relevance assessment (if available). The output of the learning to rank algorithm is a tweet ranking model, i.e. a feature combination that (based upon the training data) can rank tweets effectively. This model can then be applied to re-rank samples created for unseen queries.

However, while the basics of our learning to rank approach are the same as for blog post ranking, Twitter raises some new challenges that we do not encounter in the blog setting. First, consider the generation of the tweet sample that we rank. The aim of sample generation is to reduce the number of tweets considered, while maintaining high levels of recall (see Section 2.5.2), i.e. most of the relevant tweets should be contained within the sample. The assumption we make when using a document weighting model to produce the sample is that relevant tweets will predominantly receive higher scores than non-relevant ones. Hence, most of the relevant tweets for a query will appear somewhere in the top ranks, e.g. the top 1000. However, in a Twitter setting, this assumption may not hold for two reasons. First, for some queries, very large numbers (tens of thousands) of tweets will contain one or more of the query terms. Meanwhile, unlike in a blog setting, relevant tweets may only contain single query term due to their short length. This may mean that relevant tweets occur deeper into the ranking than when ranking blog posts. Secondly, the short length and unique characteristics of tweet texts means that relevant tweets may not contain any of the query terms because the poster uses a synonym or compression, e.g. 'Cesar Milan' might be written as '@cesarmillan'. Furthermore, one of the unique aspects of tweets is the prevalence of URLs within them. Indeed, a tweet with little actual content may be highly relevant if it links to a valuable article. In both of these cases, such relevant tweets will not reach the document sample, either because a synonym for a query term is used or because the relevancy of the tweet is derived from a link it provides rather than from its text.

Since relevant tweets may occur at lower ranks and based on recommendations in prior work (Macdonald *et al.*, 2012), we increase the depth of the tweet sample to increase recall. Under AFS, the tweet ranker R can be formalised as follows:

$$R(Q,t,S^{Tweets}) = \left\{ d \in DFReeKLIM(S^{Tweets}_{t-w\rightarrow t}, Q) : \sum_{0<i<|F^d|} weight(i) \cdot F^d_i \right\} \tag{8.5}$$

where Q is the query, d is a tweet to be scored, F^d is the set of features extracted from d, $|F^d|$ is the number of features, F^d_i is the ith feature and $weight(i)$ is the learned weight for feature F^d_i.

Next, since some tweets may not have any of the query terms due to vocabulary mismatches, e.g. the use of synonyms, we propose to use query expansion (see Section 2.3.5) to add related terms to the user query. The aim is to increase the recall of the tweet sample by including tweets that contain terms related to the query and to better estimate each tweet's relevance to that query. In particular, we use a form of real-time pseudo-relevance feedback, whereby recently posted tweets that are scored highly for the original query using a document weighting model are first selected. Frequently appearing and informative terms from the selected tweets are chosen to add to the original query. More precisely, we rank our tweet sample using the DFReeKLIM (Equation 2.10) document weighting model for the query and then select the highest scoring terms within these tweets using the KL term weighting model (Equation 2.16).

As some tweets may be relevant due to a document that they link to, we propose a novel sample expansion technique designed to add tweets to the sample that do not contain any query terms but link to related documents. In particular, we assume that for the set of tweets posted within our time window $S^{Tweets}_{t-w\rightarrow t}$, we have a related corpus of all documents linked to from that set, denoted S'. We first rank all of the linked documents in S' for the user query Q and time t. Any documents containing one or more of the query terms are selected and the tweets that linked to them are added to the tweet sample. In this way, our tweet sample contains any tweets that match one or more of the query terms or link to a document containing one or more of the query terms.

Finally, the features representing the tweets are also a critical factor. We implement a range of features from the literature. We break down our features into three different feature sets, each encoding a different type of evidence, namely: Relevance, Quality and Tweet Language. Table 8.2 lists each of our feature sets and how many features are contained within them. We summarise each feature set below:

Relevance: The first feature set that we consider describes how relevant the tweet is to the user query. In particular, we use the relevance scores assigned by the DFReeKLIM document weighting model as a

Feature Set	Feature	Description	#
Relevance	DFReeKLIM$_{t-14days}$+QE	The relevance of the tweet when considering all tweets made during the prior 2 weeks for the purposes of the collection statistics	1
	DPH-Linked$_{t-14days}$	The relevance of the document linked to from the tweet for the user query	1
Quality	Non-Alphabet	The number of non a $\rightarrow$ Z or 0 $\rightarrow$ 9 characters contained	1
	DF-Avg	The average of the document frequency scores for each of the tweet's terms	1
	DF-Sum	The summation of the document frequency scores for each of the tweet's terms	1
Tweet Content	containsURL	Does the tweet contain a URL?	1
	# Hashtags	How many hashtags does the tweet contain?	1
	Length	The total length of the tweet in characters	1
Total			8

Table 8.2: Tweet learning to rank features.

feature. Note that this document weighting model uses the background statistics of the source $S^{Tweets}_{t-w\rightarrow t}$. The parameter w defines how much of the stream history to use for the background statistics. In this case we use up to the 2 weeks of tweets from before the query time t. The second feature that we use is the relevance score for any document linked to from the tweet to the query. In particular, for each tweet posted before the query time, we score any document that it links to with the DPH document weighting model (Equation 2.9), using that score as a feature. If a tweet contains no linked document then it receives a score of 0.

Quality: The Quality feature set contains indicators of the quality of the tweet. In particular, it is comprised of three features, namely: Non-Alphabet, DF-Avg and DF-Sum. The Non-Alphabet feature describes the number of non a $\rightarrow$ Z or 0 $\rightarrow$ 9 characters that the tweet contains. This feature aids in identifying tweets in other languages that primarily use UTF-8 characters. DF-Avg and DF-Sum measure the relative informativeness of the terms contained within the tweet by measuring the number of other tweets that they appear in (Duan *et al.*, 2010).

Tweet Content: This feature set uses heuristics to encode information about the content of the tweet. In particular, it describes whether the tweet contains a URL, the number of hashtags that the tweet contains, and the length of the tweet in characters (Duan *et al.*, 2010).

In total, we use 8 tweet features to train our ranking model. We describe how this model is trained in Section 8.3 and evaluate its effectiveness in Section 8.6.

8.3 Experimental Methodology

We use TREC datasets to evaluate the effectiveness of the blog and tweet ranking approaches that we proposed in the previous section. In particular, we use the blog post ranking dataset designed for the TREC 2010 Blog track top news stories identification task to evaluate our blog post ranking approaches. For tweet ranking, we use the Twitter dataset created for the TREC 2011 Microblog track. These datasets were summarised in Section 5.2.3. We describe how we use each in the following two subsections respectively.

8.3.1 Blog Post Ranking Methodology

The aim of our blog post ranking evaluation is to determine if our proposed learning to rank approach is more effective than using existing document weighting models and state-of-the-art ranking approaches. To perform such an evaluation, we require a dataset that contains a corpus of blog posts, news-related user queries and blog post relevance assessments for those user queries. The TREC 2010 Blog track top news stories identification task examined a similar problem, i.e. the ranking of blog posts for a news story (Ounis *et al.*, 2010). Here, a news story is represented as a newswire article headline. The aim of the blog post ranking task at TREC can be summarised as "find me current and diverse blog posts relating to headline X". Although this does not exactly match our target evaluation setting (the query text is a headline rather than an actual news-related query), there are advantages to using this dataset. First, by using a TREC dataset, we are able to compare against other systems that participated in TREC for that task, providing state-of-the-art baselines. Second, using a standard evaluation methodology and open dataset enables our results to be reproduced at a later date. Finally, it is important that we are able to rank blog posts effectively both when an event first breaks and later in that event's lifetime, since (news-)related queries may occur throughout that lifetime. The TREC 2010 task enables us to evaluate this by providing blog post ranking assessments for three different points in time for the same newswire article headlines (the original article publication time, one day later and 7 days later). As a result of these advantages, we use the TREC 2010 Blog track top news stories identification task dataset to evaluate our blog post ranking approaches.

The TREC 2010 Blog track top news stories identification task uses the *Blogs08* blog post corpus, comprised of 28.5 million blog posts from the period of the 14 of January 2008 to the 10th of February 2009 (see Table 5.1). We indexed this corpus using the Terrier IR Platform (Ounis, Amati, Plachouras, He, Macdonald & Lioma, 2006). Terrier's stopword removal was applied and terms were stemmed using Porter's stemmer. Further details regarding the *Blogs08* corpus can be found in Section 5.2.1.

As noted above, news article headlines were used rather than real user queries. These news article headlines were provided by the Reuters news agency in the form of the TRC2 news article corpus (Leidner, 2010). *TRC2* contains approximately 1.8 million news articles from the period of the 1st of January 2008 to the 28th of February 2009 (see Table 5.1). From these 1.8 million news articles, 68 of them were chosen as *blog post ranking topics*. The headlines for these 68 newswire articles act as the 'queries' in our evaluation. To avoid confusion, we refer to these as blog post ranking topics in the remainder of this chapter. For all 68 topics, our learning to rank approach produces three rankings of blog posts, one containing posts published before each topic, one containing blog posts from the following day and before and one containing posts from a week following and before. This represents a system ranking for a news story at different times, i.e. as each news story matures.

For evaluation, blog post ranking assessments for 7975 blog posts were assessed across the 68 topics. Each of these posts were assessed as either Not Relevant, Relevant or Highly Relevant. Furthermore, for the 68 topics, each blog post in the pool was assessed in terms of nine perspectives: Factual Account, Opinionated Positive, Opinionated Negative, Opinionated Mixed, Short summary/Quick bites, Live Blog, In-depth analysis, Aftermath and Predictions. Notably, these assessments were crowdsourced by ourselves. Details regarding how these assessments were produced can be found in Section 5.7.

We evaluate the blog post rankings produced by our learning to rank approach based upon the number, rank and diversity of relevant blog posts contained within the top ranks. To do so, we use the official combined α-nDCG@10 measure (Clarke *et al.*, 2008). α-nDCG accounts for both the three relevance grades and the number of aspects contained within the results to a specified ranking depth, in this case the top ten results. Note that our approach focuses on blog post relevance and does not attempt aspect diversification. We compare our proposed blog post ranking approaches with traditional document weighting models (see Section 2.3) and the most effective blog post ranking approaches submitted to TREC 2010.

Importantly, the top news stories identification task was run both during 2009 and 2010. However, the experimental setting for the 2009 task makes it less suitable to evaluate our blog post ranking approaches. In particular, the 2009 setting involved the ranking of newswire article headlines by their importance (see Section 6) first, and then the ranking blog posts for the top newswire article headlines selected. In contrast, the 2010 setting separated the two tasks. Although we can rank blog posts for the newswire article headlines selected by the participants for the 2009 task, we cannot compare to them, since their performance is also dependant upon how well they could rank the newswire article headlines in the first place. Furthermore, the 2009 task did not consider the diversity of blog post aspects. For these reasons, we do not report blog post ranking performance on the 2009 topics. Instead, we use the

2009 topics to train our learning to rank approach. In particular, for the 2009 task, 22 newswire article headlines were selected. Over all 22 headlines, 7,828 blog posts were assessed as Relevant or Not Relevant. We use these newswire article headlines and relevance assessments as our blog post ranking training topics. The objective function we use to train our learning to rank model is α-nDCG, while the learning to rank algorithm used was Automatic Feature Selection (AFS) (Metzler, 2007).

8.3.2 Tweet Ranking Methodology

We evaluate our learning to rank approach for tweet ranking within the context of the TREC 2011 Microblog track (see Section 3.4.3). In particular, for a fixed set of queries, we produce reverse-chronologically ordered rankings of tweets from the public Tweets2011 TREC Twitter corpus[1]. The statistics of Tweets2011 were reported earlier in Section 5.2. We index the Tweets2011 corpus using the Terrier IR Platform (Ounis, Amati, Plachouras, He, Macdonald & Lioma, 2006), using stopword removal and Porter stemming.

Tweet ranking queries are comprised of both a textual query and a point in time to the nearest second. The aim is to produce a ranking of tweets from before the specified time that maximises relevance while maintaining recency[2]. We use the official 50 queries used during TREC 2011. This query set was judged on a three point graded scale, Highly relevant, Relevant and Not Relevant, by human assessors. Tweets were assessed based upon both their text and any documents that they link to.

We evaluate our time-ordered tweet rankings using the official precision at rank 30 (P@30) and mean average precision (MAP) measures (see Section 2.4). For evaluation, we compare to the highest performing of the TREC 2011 Microblog track participating systems as baselines, in addition to the tweet sample that our learning to rank approach re-ranks.

To train the weights for each of the features described earlier in Section 8.2.2, we train and test on the same query set using a cross-fold validation comprised of 5 training and test folds. We use MAP as the objective function and the Automatic Feature Selection (AFS) learning to rank technique for training (see Section 2.5.2).

8.4 Research Questions

Recall that within the context of this thesis, the aim of this chapter is to develop and test effective approaches for blog post and tweet ranking that we can later use to select content to integrate into the Web search ranking for news related queries. In Sections 8.2, we proposed learning to rank approaches

[1]`http://trec.nist.gov/data/tweets/`
[2]`https://sites.google.com/site/microblogtrack/2011-guidelines`

for blog post and tweet ranking. We hypothesise that these approaches will be more effective than traditional document weighting models because they incorporate evidence specific to each source in the form of features. We define three research questions that we investigate through experimentation in the following two sections (Sections 8.5 and 8.6). These three research questions are:

- Are our proposed learning to rank approaches for blog post and tweet ranking more effective than the document ranking sample that they re-rank and other state-of-the-art approaches? (Section 8.5.1 and Section 8.6.1)
- How the effectiveness of our learning to rank approach for blog post ranking change as time passes and additional documents about each topic are published? (Section 8.5.2)
- Which blog post and tweet ranking features are influential when discriminating between relevant and irrelevant posts? (Section 8.5.3 and Section 8.6.2)

8.5 Evaluation: Blog Post Ranking

In this section, we evaluate the performance of our proposed learning to rank approach to blog post ranking. In particular, we examine how effective this approach is in comparison to state-of-the-art approaches to blog post ranking and the sample that it re-ranks in Section 8.5.1. Section 8.5.2 investigates how blog post ranking performance of our approach varies over time for the same topics. In Section 8.5.3, we examine which blog features were selected by our approach and hence, are influential for blog post ranking. We provide a short summary of the outcomes of this section in Section 8.5.4.

8.5.1 Learning to Rank Performance

Recall that in Section 8.2.1, we proposed a learning to rank approach to rank blog posts using 76 features of those posts. In this sub-section, we evaluate how effectively we can rank blog posts for a news article headline (acting as a surrogate for the user query). We begin by evaluating whether our learning to rank approach is effective in comparison to our baseline approaches. In particular, we use two baselines; the best systems submitted to the TREC 2010 blog track top news stories identification blog post ranking task and the blog post sample that our approach re-ranks. If our learning to rank approach outperforms these baselines, then we can conclude that it is effective and therefore suitable to use to rank blog posts for display to the user in the next chapter.

Table 8.3 reports the news story ranking performance of our learning to rank approach (DPH Sample+LTR) in comparison to the best performing blog post ranking systems submitted to the TREC 2010

Approach	Mean	α-nDCG@10 by Category		
	α-nDCG@10	t	$t + 1day$	$t + 7days$
2^{nd} Ranked TREC System	0.4651	0.4665	0.4626	0.4663
3^{rd} Ranked TREC System	0.4266	0.4255	0.4175	0.4368
DPH Sample	0.4236	0.4092	0.4255	0.4361
DPH Sample+LTR	**0.4771▲**	**0.4688 ▲**	**0.4671 ▲**	**0.4953 ▲**

Table 8.3: Blog post ranking performance of the best TREC runs in comparison to our proposed learning to rank approach. Note that $LTR^{C8.2.1}_{BlogPosts}$ was the best TREC system, hence we do not list it twice. Cases where our learning to rank approach achieves statistically significant increases (t-test p<0.05) over the DPH sample are denoted ▲.

top news stories identification task (2^{nd} Ranked TREC System and 3^{nd} Ranked TREC System) and the blog post sample (DPH Sample). The highest performing approach is highlighted. Cases where our learning to rank approach achieves statistically significant increases (t-test p<0.05) over the DPH sample are denoted ▲.

From Table 8.3, we observe the following. First, comparing the blog post sample (row 3) to the TREC best systems (rows 1-2), we see that in terms of mean α-nDCG@10 our sample provides comparable effectiveness to the 3^{rd} Ranked TREC System but is outperformed by the 2^{nd} Ranked TREC System by a margin of 4% absolute (0.4236 to 0.4651 α-nDCG@10). This shows that the DPH ranking model offers competitive performance when ranking blog posts for these topics. Second, comparing our proposed learning to rank approach (row 4) to the sample that it re-ranks (row 3), we see that our learning to rank approach outperforms the sample by a statistically significant margin of 7.2% absolute (0.4953 to 0.4236 α-nDCG@10). This shows that our learning to rank approach is not only effective but is able to leverage the blog post features that we described in Section 8.2.1. We examine these features more closely later in Section 8.5.3. Finally, comparing our learning to rank approach (row 4) to the best TREC systems (rows 1-2), we see that our approach outperforms these systems. Indeed, our learning to rank approach was the top performing system submitted to TREC 2010 for this task. Hence, answering our first research question when ranking blog posts, based on its high performance in comparison to our baselines, we conclude that our learning to rank approach is effective and therefore suitable to use to generate rankings of blog posts that can be integrated into the Web search ranking for news-related queries.

8.5.2 Ranking Effectiveness Over Time

To answer our second research question, we next examine how our approach performs over time. Indeed, news-related queries will be submitted throughout the lifetime of an event, hence it is important to be able to effectively rank documents for a news-related query relating to an event throughout that event's

Positive/Negative	Feature Set	Feature	Weight
Positive	Relevance	DPH$_{t-1day}$	0.8890
	Relevance	DPH$_{t-1day}$+CE (TRC2)	0.4689
	Related Article	DPH$_{t-1day}$+Article	0.0717
	Structure	Ads	0.0693
	Dictionary Aspects	Official	0.0219
Negative	Structure	Links (Out-links)	-0.3853
	Dictionary Aspects	Opinionated	-0.0940
	Dictionary Aspects	Indepth	-0.0259
	Dictionary Aspects	Personal	-0.0228
	Structure	Content (Entities contained)	-0.0141

Table 8.4: The most influential features for blog post ranking for time t

lifetime. To evaluate this, we examine how our proposed approach and the baselines that we compare to varies between three points in time; the time when our blog post ranking topics were first published, one day after that time, and seven days after that time. Ideally, our approach should maintain or increase its effectiveness over time, as more relevant blog posts are published.

Table 8.3 also reports the performance of our learning to rank approach and baselines for the three time points. The highest performing approach for each time point is highlighted and time points where our approach achieves statistically significant increases (t-test p<0.05) over the DPH sample are denoted ▲. From Table 8.3, we observe two points of interest. First, comparing the performance of our approach across the three time points (row 4), we see that its performance remains almost unchanged between the original ranking time and one day later. However, when ranking a week later, ranking effectiveness is further improved by a margin of 2.8% (0.4671 to 0.4953 α-nDCG@10). This shows that our approach does indeed maintain its effectiveness over the short term and increases its effectiveness over the longer term, answering our second research question. Secondly, we observe when comparing to the next best TREC system (2^{nd} Ranked TREC System), our approach experiences greater gains as time passes between the one day later and one week later time points. This is encouraging, in that it indicates that our learning to rank approach is better able to find additional relevant documents as time passes than other state-of-the-art approaches. In turn, this indicates that the performance gain by using blog post features may increase over time.

8.5.3 Blog Post Ranking Features

Next, to answer our third research question, we investigate the features that our learning to rank approach selected. Recall that for our learning to rank approach we used the AFS learning to rank technique to combine our features (see Section 8.2.1). AFS is a linear learner, i.e. the final score for a document is the weighted sum of the individual features (Equation 8.2). The weights assigned are indicative of the influence that each feature has on the final blog post rankings produced. The features used were normalised, meaning that the resultant weights learned for each feature are directly comparable. We first report the most influential features for blog post ranking for the time point t.

Table 8.4 reports the 5 highest weighted positive and negative features selected by our approach when ranking for the initial time point t. From Table 8.4, we observe the following. First, examining the positive features selected, we see that the Relevance feature set receives the highest total weight. In particular, the two features with the highest positive impact come from the Relevance feature set. Of these two features, the highest impact feature is the DPH score for the topic, while the second is the DPH score for the headline expanded using collection enrichment on the TRC2 corpus. The weights for these two features combined comprises 89% of the total weight for the top 5 positive features. This shows the overall impact of the document weighting model scores on the overall ranking. Secondly, the third most influential positive feature is also related to the relevancy of the blog post, specifically it is the DPH score when using the article body associated to the headline as the query. However, the much lower weight assigned in comparison to the other relevancy-based features indicate that it is less useful. Third, we see that the remaining two positive features are query independent. In particular, the fourth positive feature selected was whether each blog post contained adverts. This indicates that for the topics tested here, blog posts that contain adverts are more likely to be relevant. This in turn indicates that some relevant blog posts come from dedicated news-related blogs that are funded by advertising. For example, the top result returned by our learning to rank approach for the topic ‘Tele Atlas in 5-yr deal to provide maps to Google’ was a page from the TechCrunch blog[1]. The final positive feature selected was the number of terms from our ‘official’ dictionary that the blog post contained. This similarly indicates that official blogs are more likely to be relevant than personal ones.

Next, we examine the negative features, i.e. those that are indicative of irrelevant blog posts. The lower half of Table 8.4 reports the 5 highest weighted (most influential) negative features. From Table 8.4, we observe the following. Firstly, the largest negative indicator of relevance as selected by our learner is the number of out-links that a blog post has, with a weight of -0.3853. This indicates that blog posts containing large numbers of links are often irrelevant. Indeed, we have anecdotal evidence that such pages belong to *link-farms* (Wu & Davison, 2005). We also see that the opinionatedness feature received a negative weight. This indicates that highly opinionated blogs are likely to be irrelevant. This in turn may mean that end-users do not want to see opinionated blog posts as much as authoritative

[1]`http://www.techcrunch.com/2008/06/30/in-new-deal-with-tele-atlas-google-maps-sends-data-back/`

ones. We also observe that negative weights were assigned to the Indepth and Personal dictionary aspect features, although the limited weight assigned to these features implies that they are weak negative indicators of relevance at best.

In Section 8.5.2 we reported that our learning to rank approach was more effective when ranking one week later for the same topics. This result might indicate that the performance gain from using blog post features increases over time. To investigate this, we examine whether the features selected to rank blog posts change when ranking at this later time point. In particular, we compare the features selected when ranking at time t to those selected at time $t + 7days$. Table 8.5 reports the top 5 positive and negative features in terms of their assigned weights when ranking for time $t + 7days$. Comparing the features from Table 8.4 to those from Table 8.5, we observe the following. First, as we would expect, the Relevancy feature set contributes the most influential positive features. However, we also observe that the second most influential feature is no longer a Relevancy feature, but rather indicates that relevant blogs often contain copyright statements. A reason that this feature might become so discriminative is that as time passes, blog posts are likely to be posted from blog commentators employed by news-related companies, e.g. TechCrunch[1]. These posts are likely to both be relevant and contain copyright statements from their employer.

In terms of negative features, we see that the opinionatedness feature is still selected, indicating that highly opinionated blogs are seen as less relevant. However, in contrast to ranking at time t, we also see that some part of speech features are selected. In particular, the presence of foreign words were selected as a negative feature. This is likely an artefact of the blog post assessment by workers, who were instructed to treat non-English blogs as irrelevant. The presence of superlative adverbs, e.g. ‘most carefully’ also appear to be negative indicators of relevance.

8.5.4 Conclusions

In this section, we evaluated our learning to rank approach for blog post ranking within the context of the TREC 2010 Blog track top news stories identification blog post ranking task. Our experimental results attest to the effectiveness of the approach in comparison to the best TREC 2010 systems for the same task and against the blog post sample that it re-ranks. Furthermore, we have seen that the ranking performance of our proposed approach maintains its effectiveness over time, indeed increasing in effectiveness when ranking for the same topics one week later. Through an examination of the features selected by our learning to rank approach, we have shown that the relatedness of the blog post to the topic using the DPH document weighting model is the largest indicator of relevance (as expected).

[1]`http://techcrunch.com/`

Positive/Negative	Feature Set	Feature	Weight
Positive	Relevance	DPH$_{t-1day}$	0.5505
	Structure	Content (Copyrighted Blogs)	0.4868
	Relevance	DPH$_{t-1day}$+CE (Blogs06)	0.2488
	Related Article	DPH$_{t-1day}$+QE (Article)	0.0189
	Relevance	DPH$_{t-1day}$+Article	0.0182
Negative	Part of Speech	Term Type Counts (Foreign word)	-0.1135
	Dictionary Aspects	Opinionated	-0.1082
	Part of Speech	Term Type Counts (Adverb, superlative)	-0.0259
	Dictionary Aspects	Shallow	-0.0146
	Part of Speech	Term Type Counts (Existential there)	-0.0094

Table 8.5: The most influential features for blog post ranking for time $t + 7days$

However, we have also seen that official blog posts are more likely to be relevant and that our learner can encapsulate this evidence using features such as the presence of advertising or copyright statements.

From these experiments, we conclude that our learning to rank approach is state-of-the-art in blog post ranking effectiveness. We use this approach to rank blog posts that are subsequently integrated into the Web search ranking in Chapter 9. In the next section, we experiment to determine whether our variant of this approach tweet ranking is also effective.

8.6 Evaluation: Tweet Ranking

In this section, we evaluate the effectiveness of our learning to rank approach for real-time tweet ranking. In particular, we examine how effective this approach is in comparison to state-of-the-art approaches to real-time tweet ranking submitted to the TREC 2011 Microblog track and the tweet sample that it re-ranks in Section 8.6.1. In Section 8.6.2, we investigate which tweet features were selected by our approach and hence, are influential on the resultant tweet ranking. We provide a short summary of the outcomes of these experiments in Section 8.6.3.

8.6.1 Learning to Rank Performance

We begin by investigating how whether our learning to rank approach is more effective than traditional document weighting models for tweet ranking. If our learning to rank approach is shown to be effective, then we will use it in the next chapter to rank tweets for news-related user queries and subsequently integrate those tweets into the Web search ranking. We compare our approach against a series of tweet ranking baselines. First, we compare against two document weighting models that only consider only the tweet text when ranking, namely BM25 and DFReeKLIM. Second, we compare against three document weighting models with query expansion, namely BM25, DirichletLM and DFReeKLIM, of which

Approach	Sample	QE	MAP	P@30
TREC Median	N/A	N/A	N/A	0.259
1st Ranked TREC 2011 System	Unknown	Unknown	0.2078	0.4082
2nd Ranked TREC 2011 System	N/A	Unknown	0.2049	0.3986
Document Ranking Model	BM25	✖	0.2443	0.3204
Document Ranking Model	DFReeKLIM	✖	0.3103	0.3769
Document Ranking Model	BM25	✔	0.2599	0.3381
Document Ranking Model	DirichletLM	✔	0.2981	0.3857
Document Ranking Model (*Sample*)	DFReeKLIM	✔	0.3464	0.4252
LTR	DFReeKLIM	✔	**0.3814▲**	**0.4548**

Table 8.6: Tweet ranking effectiveness of LTR using a variety of learning algorithms. ▲ denotes a statistically significant (t-test p<0.05) increase over tweet ranking sample (*Sample*).

DFReeKLIM is the sample that our approach re-ranks. More information on these document weighting models and query expansion can be found in Section 2.3. We also report the effectiveness of the TREC 2011 best systems, representing state-of-the-art approaches to real-time tweet ranking, in addition to the median of all systems that participated in the TREC 2011 Microblog track. However, since our learning to rank approach employs 5 folds of cross validation for training (due to a lack of a second training dataset), it is unfair to make a direct comparison to these systems (who could not employ a similar training approach).

Table 8.6 reports the tweet ranking performance of our learning to rank approach (LTR) as an average over 5 folds of cross-validation on the 2011 Microblog topic set in comparison to our baselines for the official MAP and P@30 measures. The best performing approach under each measure is highlighted and statistically significant (t-test p<0.05) increases over the tweet ranking sample (*Sample*) is denoted ▲. From Table 8.6 we observe the following. First, as a sanity check, we compare our baseline document weighting models without query expansion (rows 4-5) to the median of the TREC participating systems (row 1). We observe that these basic approaches outperform the median of the TREC 2011 systems by 6.1% and 11.8% P@30 for BM25 and DFReeKLIM, respectively. This shows that our baselines are able to rank tweets with some degree of accuracy. Second, comparing these document weighting models without query expansion (rows 4-5) to the best TREC systems (rows 2-3), we see that they outperform the TREC best systems under MAP but underperform in terms of P@30. From this result, we conclude that these basic approaches are strong tweet ranking baselines and that the TREC best systems appear to have been optimised for the P@30 measure. Third, comparing the document weighting models (rows 4-5) to those same models with query expansion applied (rows 6-8), we see that the addition of query expansion increases tweet ranking performance markedly. For instance, adding query expansion to the DFReeKLIM document weighting model increases MAP performance from 0.3103 to 0.3464 and P@30 performance from 0.3769 to 0.4252. Indeed, we see that DFReeKLIM with query expansion (row 8) outperforms the TREC best systems (rows 2-3) under both measures. This shows that query expansion is effective in a tweet ranking scenario and that DFReeKLIM with query expansion is a strong baseline. This approach also generates the tweet sample that our learning to rank approach re-ranks. Finally, comparing our learning to rank approach (row 9) to the sample that it re-ranks (row 8), we see that it significantly outperforms the sample in terms of MAP (0.3464 to 0.3814) and markedly outperforms the sample in terms of P@30 (0.4252 to 0.4548). From this result, we conclude that our learning to rank approach is effective when ranking tweets, indeed outperforming all of the other approaches tested, answering our first research question.

8.6.2 Tweet Ranking Features

To answer our second research question, we next examine the features that our learning to rank model selected. In our previous experiments, the learning to rank performance was reported over 5 folds of cross validation. Rather than report the features selected by each fold individually, we train a model using all of the available training data (the combination of all five folds) and report the features selected. As before, the feature scores were normalised such that the weights output by AFS are comparable. The higher the weight assigned to each feature, the more influential it is on the final tweet rankings produced.

Table 8.7 reports positive and negative features selected by our approach when using all 50 available training topics and their weights. From Table 8.7, we observe the following three points of interest. First, we see that the most influential tweet ranking feature is the retrieval score of the tweet using the DFReeKLIM document weighting model with query expansion (the tweet sample). This is expected given the effective performance of that approach, as shown in the previous section. Second, the next most influential positive feature is the score of the document linked to from the tweet. This is an important result, since it shows that our learning to rank approach is effectively using the documents linked to from the tweets to better estimate whether those tweets are relevant. The third positive feature that was selected is whether the tweet contained a URL or not. This indicates that in general, tweets that contain URLs are more likely to be relevant than those that do not.

We also examine the features that indicate irrelevant tweets, i.e. those features that receive negative weights. The bottom half of Table 8.7 reports the most influential negative features selected by our approach. From Table 8.7 we observe the following. The two most influential negative features are the sum and average of the document frequencies of the terms contained within it. This indicates that tweets containing highly popular terms are unlikely to be relevant. In effect, this feature causes tweets containing common topics like 'Justin Beiber' to be ranked lower. The third negative feature selected

Positive/Negative	Feature Set	Feature	Weight
Positive	Relevance	DFReeKLIM$_{t-14days}$+QE	0.4440
	Relevance	DPH-Linked$_{t-14days}$	0.0687
	Tweet Language	containsURL	0.0256
Negative	Quality	DF-Avg	-0.2299
	Quality	DF-Sum	-0.1934
	Tweet Content	Length	-0.0235
	Tweet Content	# Hashtags	-0.0143
	Tweet Content	Non-Alphabet	-0.0004

Table 8.7: The most influential features for tweet ranking.

is the length of the tweet (in characters). This in turn indicates that long tweets are less likely to be relevant than shorter ones. However, the low nature of the weight for this feature also indicates that the correlation between the number of characters a tweet contains and the relevance of that tweet is weak. The fourth negative feature selected is the number of hashtags that the tweet contains, indicating that tweets containing multiple hashtags are less likely to be relevant. The final feature is the number of non a $\rightarrow$ Z and 0 $\rightarrow$ 9 characters that the tweet contains. The very low weight assigned indicates that this feature is not sufficiently discriminative or is redundant. In general, we conclude that our approach combines both document weighting model scores and tweet-specific features to better estimate when a tweet is relevant for the query.

8.6.3 Conclusions

In this section, we investigated how effective our learning to rank approach is for tweet ranking. We evaluated this approach using the TREC 2011 Microblog track dataset. Our results show that our learning to rank approach is effective, as it significantly outperforms the tweet sample that it re-ranks, while that sample in turn outperforms the best TREC 2011 approaches for the task. We then analysed the features that our learning to rank approach selected. We observed that the most influential positive feature is the score for each tweet from the sample, followed by the DPH score for any document linked to from each tweet. This shows how our approach combines evidence from both the tweet text and linked documents to produce a more effective tweet ranking. Tweets containing URLs were also shown to be more likely to be relevant for each query. Meanwhile, the presence of terms with high DF scores were shown to be influential negative features.

From these experiments, we conclude that our learning to rank approach provides state-of-the-art tweets ranking performance and hence we use this approach to rank tweets that are subsequently integrated into the Web search ranking in Chapter 9. In the next section, we conclude on the overall findings of this chapter.

8.7 Conclusions

In this chapter, we investigated the Ranking News-Related Content (RNRC) component of our proposed news search framework. The aim of the RNRC component is to rank content for a news-related user query from each of the news-related content sources available at that time, such that the ranked content can be integrated into the Web search ranking for display top the user for news-related queries. In particular, we focused on two user-generated content sources, namely blog posts and tweets. For each of these two sources, we proposed learning to rank approaches to tackle them and evaluated their effectiveness against more traditional document weighting models.

In particular, in Section 8.2 we proposed learning to rank approaches for ranking blog posts and tweets in a real-time manner. These approaches leverage source specific information to more effectively identify relevant blog posts and tweets for a user query. For blog posts, this takes the form of identified aspects of each post such as opinionatedness. Meanwhile, for tweets, we consider the characteristics of each tweet such as hashtag usage and the relevance of any documents linked to from each tweet.

As described in Section 8.3, we evaluate these learning to rank approaches using standard TREC datasets. In particular, for blog post ranking, we use the TREC 2010 Blog track top news stories identification task blog post ranking dataset. Under this dataset, each query is a newswire article headline. We rank blog posts from the $Blogs08$ blog post corpus for each headline for the day that the headline was published. For tweet ranking, we use the dataset designed for the TREC 2011 Microblog track. Here we rank tweets from the Tweets2011 corpus for a series of tweet search queries and points in time. We listed the research questions that we addressed in this chapter in Section 8.4.

Through experimentation in Section 8.5, we showed that our proposed learning to rank approach for blog post ranking on the $BlogTrack_{2010}^{TopNews-Phase2}$ dataset outperformed the best TREC 2010 systems for the same task in addition to the blog post sample that it re-ranks (Table 8.3). This attests to the effectiveness of the approach. Furthermore, we showed that the ranking performance of our proposed approach maintains or increases its effectiveness when ranking for the same topics over week long period (Table 8.3). Through an examination of the features selected by our approach (Table 8.4), we have showed that the relatedness of the blog post to the topic using the DPH document weighting model is the largest indicator of relevance (as expected) and that presence of advertising or copyright statements are also positive indicators of relevance in a blog post ranking setting.

We then examined how our learning to rank approach faired when ranking tweets in a real-time setting on the $MicroblogTrack_{2011}$ dataset in Section 8.6. Our results similarly attest to the effectiveness of the proposed approach when ranking tweets. In particular, we showed that our approach significantly outperforms the tweet sample that it re-ranks, while that sample in turn outperforms the best TREC 2011 approaches for the task (Table 8.6). We then analysed the features that our learning to rank approach selected. We observed that the most influential positive feature is the score for each tweet in the sample, followed by the DPH score for any document linked to from each tweet (Table 8.7). This illustrates how our approach is able combine evidence from both the tweet text and linked documents to produce a more effective tweet ranking. Furthermore, an analysis of the negative features selected (Table 8.7) indicated that the presence of terms with high DF scores were shown to be influential, i.e. tweets using overly common terms are less likely to be relevant.

In our thesis statement (see Section 1.3), we postulated that the appropriate integration of user-generated content into the Web search ranking would increase the coverage of events that users are looking to know more about. To achieve this, effective approaches to find and rank documents are needed, such that irrelevant documents are not integrated. From our experiments in this chapter, we conclude that the proposed learning to rank approaches are effective for ranking user-generated content from the blogosphere and Twitter. These approaches enable blogs and tweets to be ranked effectively within a universal Web search engine for news-related user queries, which can subsequently be integrated into the Web search ranking for display to the user.

In the last of our experimental chapters (Chapter 9), we examine how news and user-generated content can be integrated into the Web search results. In particular, we perform a crowdsourced user study to investigate whether end-users find that the integration of user-generated content increases coverage for news-related user queries, supported by the effective ranking approaches proposed and tested in this chapter.

Chapter 9

News-Related Content Integration

9.1 Introduction

In the previous three chapters, we have shown how user-generated content can be leveraged in three components of our news search framework that describes a universal Web search engine with support for a news vertical. These three components were: the Top Events Identification (TEI) component, which automatically identifies important news stories in real-time (Chapter 6); the News Query Classification (NQC) component that classifies user-queries as news-related or not in real-time (Chapter 7); and the Ranking News-Related Content (RNRC) component, which ranks user-generated content from different sources for the user query (Chapter 8). In this chapter, we investigate the fourth and final component of our news search framework, i.e. the News-Related Content Integration (NRCI) component.

Recall that the purpose of the NRCI component is to integrate the news-related content ranked by the RNRC component and the event rankings produced by the TEI component into the Web search ranking. The aim is to increase either user satisfaction or coverage for the user query by providing additional news-related documents. From Equation 4.4, the functionality of this component can be expressed as a function M, which takes a series or news and user-generated content rankings (produced by the RNRC component), a news article ranking (produced by the TEI component), the Web search ranking, the user query and the time that query was issued, producing a final ranking that combines the different rankings together for that query and time.

In this chapter, we define an approach M based upon prior work in resources selection (see Section 2.8). We use the defined function M to produce rankings enhanced with news and user-generated content for a series of news-related queries. We then develop a ranking evaluation framework that enables users to compare rankings and state their preference for one ranking or another. Using this framework in conjunction with the crowdsourcing marketplace Amazon's Mechanical Turk[1], we evaluate whether end-users prefer rankings that integrate user-generated content and study the types of queries that benefit from this content.

Within the context of the overall thesis, this chapter has three aims. First, we aim to validate our thesis statement and news search framework from a user perspective, by evaluating whether the integration of user-generated content increases user satisfaction and provides better coverage for some news-related queries. Secondly, we examine whether the addition of user-generated content to the Web search ranking can increase coverage for those user queries where newswire articles are shown not to help. Thirdly, we investigate whether the addition of user-generated content enables news-related queries to be satisfied in a more timely manner.

We structure the remainder of this chapter as follows:

- In Section 9.2, we describe our approach for integrating news and user-generated content into the Web search ranking.
- Section 9.3 details our evaluation framework for performing comparative evaluations between document rankings.
- In Section 9.4, we define the research questions that we investigate in this chapter.
- Section 9.5 describes our experimental methodology, including how we generated the document rankings and the specific setup of our crowdsourced evaluation.
- In Section 9.6, we experiment to determine whether the integration of user-generated content better satisfies users than unaltered Web search rankings for news-related queries.
- We summarise the findings of this chapter and provide conclusions in Section 9.7.

9.2 Resource Selection for News-Related Content Integration

As we described previously in Section 4.4, to combine multiple rankings of news and user-generated content into the Web search ranking, we require a ranking function M (see Equation 4.4). Importantly, M combines three different types of rankings together, namely: the Web search ranking R_W for the query Q and time t; the news and user-generated content rankings $R_{1..n}$ for the query Q and time t; and the query independent news article ranking I, representing the top events occurring at time t. For this reason, we consider the ranking function M to be comprised of two sub-functions, one that merges the

[1]http://mturk.com

query dependant rankings (R_W and $R_{1..n}$) together, and one that optionally integrates the event ranking I based upon the query Q, as shown below:

$$M(R_{1..n}, R_W, I, Q, t) = \begin{cases} Integration_{News}(R_W, I) & \text{if } Generic(Q) \\ Integration_{RE}(Q, R_{1..n}, R_W, S) & otherwise. \end{cases} \tag{9.1}$$

where $Generic(Q)$ estimates whether the query Q is a generic news query (see our news query taxonomy in Section 4.3). $Integration_{News}(R_W, I)$ integrates the ranking of top news events I into the Web search ranking R_W by selecting up three top events to add to the top of the search ranking. This simulates top events integration by major Web search engines such as Bing (see Figure 4.4 for an example). $Integration_{RE}(Q, R_{1..n}, R_W, S)$ integrates the rankings $R_{1..n}$ and R_W by normalising the scores for each document in those rankings to make those scores comparable across the sources S that they came from. $Integration_{RE}(Q, R_{1..n}, R_W, S)$ is calculated as follows:

$$Integration_{RE}(Q, R_{1..n}, R_W, S) = \{d \in R_{i \in 1..n \cup W} : score(Q, d, R_i, S_i, S)\} \tag{9.2}$$

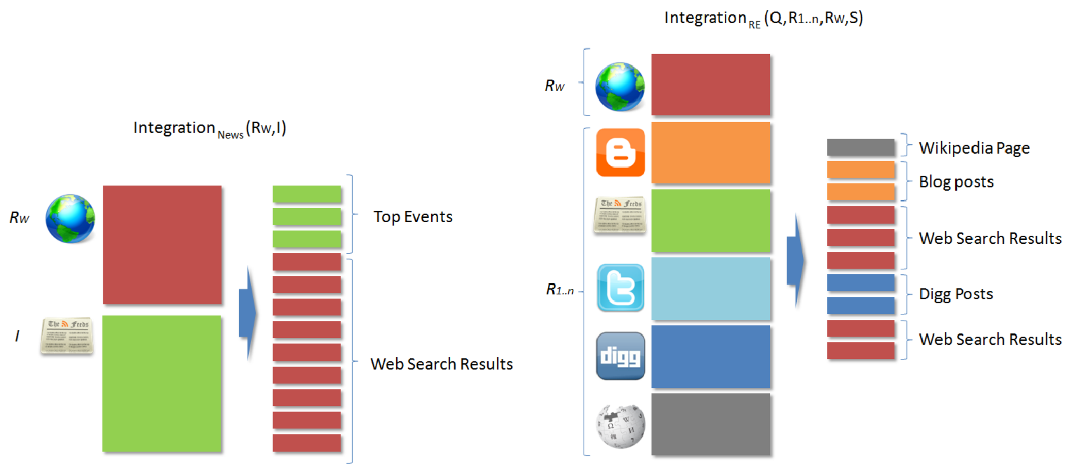
where d is a document in either $R_{1..n}$ or R_W (denoted $R_{i \in 1..n \cup W}$), i is the index of the ranking and source that d comes from. $score(Q, d, R_i, S_i, S)$ re-scores d given the query Q, the ranking that d came from R_i, the source that d came from S_i and the set of all sources being combined S.

To calculate $score(Q, d, R_i, S_i, S)$, we propose a simple baseline function. Note that we choose a simple function here to avoid introducing a confounding variable to our later experiments. In particular, we adapt the scoring function used by the INQUERY resource selection system (Equation 2.27). However, since the collection statistics of our sources markedly differ, we use the CRCS (Equation 2.26) resource selection approach to score each of our sources rather than the CORI algorithm (Equation 2.24) that was used in the original paper. Our final scoring function is shown below:

$$score(Q, d, R_i, S_i, S) = \frac{score(d, R_i) - d_{min_{Ri}}}{d_{max_{Ri}} - d_{min_{Ri}}} + 0.4 \cdot \frac{score(d, R_i) - d_{min_{Ri}}}{d_{max_{Ri}} - d_{min_{Ri}}} \cdot \frac{score_{CRCS}(Q, S_i, S)}{\max_{s' \in S} score_{CRCS}(Q, s', S)} \tag{9.3}$$

where $score(d, R_i)$ is the score for document d from the ranking R_i, $d_{min_{Ri}}$ is the minimum score of any document received for the query Q, while $d_{max_{Ri}}$ is the maximum score that any document received. $score_{CRCS}(Q, S_i, S)$ is the score for the source $S_i \in S$ for the query Q, using the CRCS resource selection approach (Equation 2.26) described in Section 2.8.1, while $\max_{s' \in S} score_{CRCS}(Q, s', S)$ is the maximum CRCS score that was assigned to any source $s' \in S$ for Q.

Figure 9.1 a) and b) illustrate the inputs and outputs of the ranking function M for both sub-functions ($Integration_{News}(R_W, I)$ and $Integration_{RE}(Q, R_{1..n}, R_W, S)$). We observe that for top events



(a) Top events integration.

(b) News and user-generated content integration.

Figure 9.1: Illustration of the top events and news and user-generated content integration into the Web search ranking ($Integration_{News}(R_W, I)$ and $Integration_{RE}(Q, R_{1..n}, R_W, S)$).

integration (Figure 9.1 a), the Web search ranking R_W and top events ranking I are taken as input. The output is a ranking of the top three events, followed by the Web search ranking. For the news and user-generated content integration sub-function (Figure 9.1 b), the rankings of news articles, blogs, tweets, Digg posts and Wikipedia pages are taken as input along with the Web search ranking. The documents within each of these sources are re-scored and then ranked. In this example, one Wikipedia page, two blogs, five Web search results and two Digg posts made the top ten results. In the next section, we describe how the rankings produced by M are presented to the user for evaluation.

9.3 Comparative Ranking Evaluation and Presentation

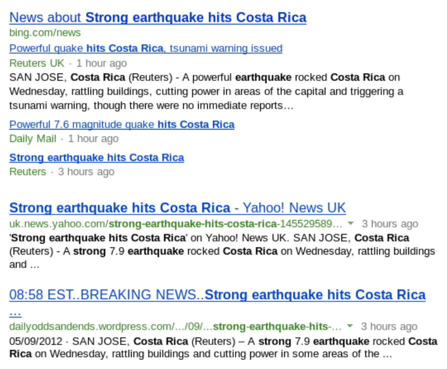
Having developed an approach to generate rankings enhanced with news and user-generated content for news-related queries, the next stage is to develop a framework to both display these rankings and to facilitate the comparison of the rankings by users. Indeed, the overall goal of this chapter is to determine whether the enhanced rankings produced are superior to Web search rankings for news-related queries through an evaluation with end-users.

In this chapter, inspired by work by Benjamin (2009), we perform a comparative evaluation of different document rankings, where the top ten documents from each ranking are considered as a single evaluation unit (rather than evaluating the individual documents). For this type of evaluation, multiple different rankings are displayed to the user, who then selects the one that he/she prefers. In our scenario, the user is shown both the unmodified Web search ranking and a ranking enhanced with user-generated content for a query. Comparative ranking evaluation is more suitable for our scenario than a typical document relevance style evaluation, since relevance alone does not encapsulate the overall value of the ranking to the user in a news vertical setting. In particular, a document may seem relevant, but is in-fact out-of-date, provides only redundant information or does not cover the aspect of the query that the user is interested in. In contrast, by evaluating the combined ranking as a whole, we aim to better identify when the integration of user-generated content results in better user satisfaction and coverage for news-related queries.

Importantly, using assessments from only one or two different users is potentially problematic, since individual preferences may bias the outcome. For this reason, we use crowdsourcing for assessment, leveraging the easy access to a large pool of workers that the medium provides. We use these workers as a surrogate for end-users. In particular, we have 10 different workers compare each Web search ranking to its corresponding enhanced ranking for a set of news-related queries. This enables us to aggregate assessor preferences to find the majority preference, i.e. to determine whether the majority of users find that the integration of additional news-related content improves coverage or not.

However, no tools exist to facilitate ranking comparisons by end-users (workers). Instead, we develop our own framework to achieve this. In particular, our framework takes as input a series of document ranking pairs for a query, each pair is comprised of a Web search ranking and a ranking enhanced with news and/or user-generated content. The framework automatically handles the presentation of these ranking pairs, facilitates user-interaction for stating a preference for one of the rankings and is responsible for recording that preference. Note that we have workers assess multiple ranking pairs at one time to speed up the assessment process.

For presentation of a single ranking, our framework supports both individual documents and document groups. This is an important distinction to make when presenting vertical results to the user. For example, Figure 9.2 (a) shows the top three search results for the query ‘Strong earthquake hits Costa Rica’ by the Bing search engine on the 5th of September 2012. From Figure 9.2 (a), we observe that there are two distinct types of result, a news vertical box containing three top news results for the query, and two Web page results. Our framework automatically presents groups of documents in a styled box, in a similar manner to the Bing news results in the prior example. Single results are also presented in a similar manner to Bing. For instance, Figure 9.2 (b) illustrates how our framework displays three blog posts and two Web search results for the query ‘north korea’. To avoid drowning the Web search ranking with large volumes of news and user-generated content, our framework performs automatic grouping

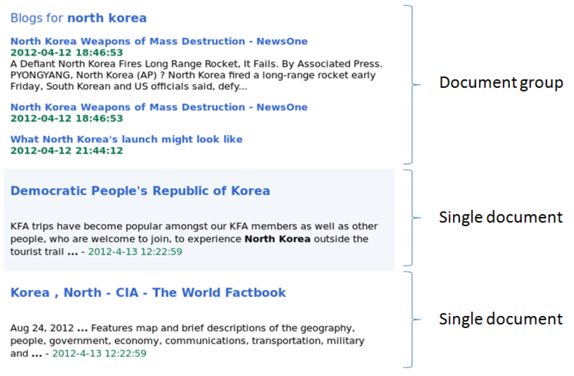


News about **Strong earthquake hits Costa Rica**
bing.com/news
Powerful quake **hits Costa Rica**, tsunami warning issued
Reuters UK · 1 hour ago
SAN JOSE, **Costa Rica** (Reuters) - A powerful **earthquake** rocked **Costa Rica** on Wednesday, rattling buildings, cutting power in areas of the capital and triggering a tsunami warning, though there were no immediate reports…
Powerful 7.6 magnitude quake **hits Costa Rica**
Daily Mail · 1 hour ago
Strong earthquake hits Costa Rica
Reuters · 3 hours ago

Strong earthquake hits Costa Rica - Yahoo! News UK
uk.news.yahoo.com/**strong-earthquake-hits-costa-rica**-145529589… 3 hours ago
'**Strong earthquake hits Costa Rica**' on Yahoo! News UK. SAN JOSE, **Costa Rica** (Reuters) - A **strong** 7.9 **earthquake** rocked **Costa Rica** on Wednesday, rattling buildings and ...

08:58 EST..BREAKING NEWS..**Strong earthquake hits Costa Rica** ...
dailyoddsandends.wordpress.com/…/09/…**strong-earthquake-hits-**… 3 hours ago
05/09/2012 · SAN JOSE, **Costa Rica** (Reuters) – A **strong** 7.9 **earthquake** rocked **Costa Rica** on Wednesday, rattling buildings and cutting power in some areas of the ...

(a) Top three Bing Web search results for the query 'Strong earthquake hits Costa Rica' on 5/9/12.



Blogs for **north korea**
North Korea Weapons of Mass Destruction - NewsOne
2012-04-12 18:46:53
A Defiant North Korea Fires Long Range Rocket, It Fails. By Associated Press. PYONGYANG, North Korea (AP) ? North Korea fired a long-range rocket early Friday, South Korean and US officials said, defy...
North Korea Weapons of Mass Destruction - NewsOne
2012-04-12 18:46:53
What North Korea's launch might look like
2012-04-12 21:44:12

Democratic People's Republic of Korea
KFA trips have become popular amongst our KFA members as well as other people, who are welcome to join, to experience **North Korea** outside the tourist trail **...** - 2012-4-13 12:22:59

Korea , North - CIA - The World Factbook
Aug 24, 2012 **...** Features map and brief descriptions of the geography, people, government, economy, communications, transportation, military and **...** - 2012-4-13 12:22:59

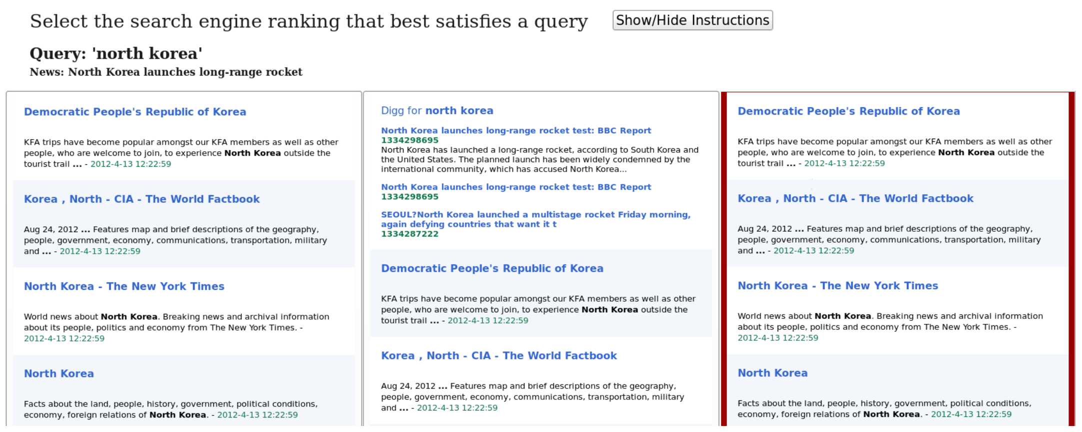
(b) Top three Web search results produced by our framework for the query 'north korea' on 13/4/12 enhanced using blog posts.

Figure 9.2: Comparison of our framework presentation of search results to the Bing Web search engine.

of content by their type using three presentation heuristics. These heuristics are designed to simulate document presentation by major Web search engines. The three heuristics are:

- If a group of integrated documents exist, i.e. two or more documents are appear sequentially in the enhanced ranking, then these are rendered as a document group. See Figure 9.2 for an example of document group presentation in comparison to a similar ranking produced by the Bing search engine.
- No news or user-generated content source may have more than three documents or one document group integrated per ranking. This heuristic aims to avoid flooding the Web search ranking with integrated content.
- Wikipedia pages are always rendered singly. Web search engines typically treat Wikipedia pages as normal Web search results, this heuristic simulates that behaviour.

Our framework combines a set of worker instructions with the different document rankings to be compared in a single interface that is displayed to the user. Figure 9.3 provides an illustration of the assessment interface produced by our framework for the query 'north korea' made on the 13th of April 2011. The enhanced ranking in this case integrates Digg posts. At the top of the interface, the title of the task is displayed along with a button that reveals the instructions for the assessment task (hidden in this example). The instructions are divided into two separate components, the main instruction block that describes what the worker is being asked to do, and the guidelines block that provides additional clarifications about the task. The instructions provided to each worker for the experiments detailed in



Select the search engine ranking that best satisfies a query

Query: 'north korea'

News: North Korea launches long-range rocket

Democratic People's Republic of Korea

KFA trips have become popular amongst our KFA members as well as other people, who are welcome to join, to experience **North Korea** outside the tourist trail **...** - 2012-4-13 12:22:59

Korea , North - CIA - The World Factbook

Aug 24, 2012 **...** Features map and brief descriptions of the geography, people, government, economy, communications, transportation, military and **...** - 2012-4-13 12:22:59

North Korea - The New York Times

World news about **North Korea**. Breaking news and archival information about its people, politics and economy from The New York Times. - 2012-4-13 12:22:59

North Korea

Facts about the land, people, history, government, political conditions, economy, foreign relations of **North Korea**. - 2012-4-13 12:22:59

Digg for **north korea**

North Korea launches long-range rocket test: BBC Report
1334298695
North Korea has launched a long-range rocket, according to South Korea and the United States. The planned launch has been widely condemned by the international community, which has accused North Korea...

North Korea launches long-range rocket test: BBC Report
1334298695

SEOUL?North Korea launched a multistage rocket Friday morning, again defying countries that want it t
1334287222

Democratic People's Republic of Korea

KFA trips have become popular amongst our KFA members as well as other people, who are welcome to join, to experience **North Korea** outside the tourist trail **...** - 2012-4-13 12:22:59

Korea , North - CIA - The World Factbook

Aug 24, 2012 **...** Features map and brief descriptions of the geography, people, government, economy, communications, transportation, military and **...** - 2012-4-13 12:22:59

Democratic People's Republic of Korea

KFA trips have become popular amongst our KFA members as well as other people, who are welcome to join, to experience **North Korea** outside the tourist trail **...** - 2012-4-13 12:22:59

Korea , North - CIA - The World Factbook

Aug 24, 2012 **...** Features map and brief descriptions of the geography, people, government, economy, communications, transportation, military and **...** - 2012-4-13 12:22:59

North Korea - The New York Times

World news about **North Korea**. Breaking news and archival information about its people, politics and economy from The New York Times. - 2012-4-13 12:22:59

North Korea

Facts about the land, people, history, government, political conditions, economy, foreign relations of **North Korea**. - 2012-4-13 12:22:59

Figure 9.3: Illustration of the ranking interface produced for the query ‘north korea’ on 13/4/12 with our framework, where the enhanced ranking integrates Digg posts. The first two rankings are the ranking pair to be assessed. The third (red) bordered ranking a validation ranking designed to catch workers that are randomly clicking.

this chapter are listed later in Section 9.5.4. When an assessor first views the task (before accepting it) the instructions are displayed to the user. When actively attempting the task, the instructions are by default closed to save screen real-estate.

Below the instructions, the user query is displayed along with a short description of the news event that the user is searching for. For the example in Figure 9.3, the user was searching for information about the failed long range rocket test by North Korea that had occurred that day. The query descriptions were manually generated by the author with reference to the main news stories that were in press at the time that each query was made. Below the query and description, the document rankings are rendered. To enable assessors to state their preference for one of the displayed rankings, our framework makes each ranking clickable. Clicking a ranking records the user’s preference for that ranking and loads the next ranking pair to assess. Notably, our evaluation framework skips cases where a source contains no relevant documents for a query, or the documents returned for that source are scored such that they would not make the top ten results, i.e. when the enhanced and Web rankings would be identical.

As described in Chapter 5, one of the criticisms of crowdsourcing is susceptibility to spam and poor quality work. To counter-act this, the assessments produced need to be validated. Our framework interface also integrates one method for validating user work. In particular, the framework renders the Web result ranking twice, creating three rankings. One of these rankings has a red border. The assessors are instructed never to select the red bordered ranking. We use this ranking to identify bots or malicious assessors that are randomly selecting rankings. Any set of assessments where a red-bordered ranking has been clicked is automatically rejected without payment. In the next section, we list the research questions that we use our evaluation framework to address.

9.4 Research Questions

In this chapter, we investigate four research questions through a crowdsourced user study using the ranking approach described in Section 9.2 and our framework for comparative ranking evaluation described in Section 9.3. These four research questions are as follows:

- Does the integration of *top event rankings* (I) into the Web search results better satisfy end-users for generic news-related queries? (Section 9.6.1)
- Do end-users find that the addition of content ranked from any of our *news or user-generated content sources* ($R_{1..n}$) increase user-satisfaction and coverage for news-related user queries? (Section 9.6.2)
- Can the integration of user-generated content better satisfy news-related queries than the integration of newswire articles? (Section 9.6.3)
- Does the addition of user-generated content enable news-related queries to be satisfied in a more timely manner? (Section 9.6.4)

9.5 Experimental Methodology

In this section, we describe our methodology for evaluating the rankings produced by our framework for a series of news-related user queries. In particular, in Section 9.5.1, we summarise the content sources and queries that we use during our evaluation. Section 9.5.2 details how each of the document rankings that we use are generated. In Section 9.5.3, we describe how our evaluation is structured into two experiments. In Section 9.5.4, we list the instructions that were provided to the workers for each of the two experiments. Finally, Section 9.5.5 details how our crowdsourcing tasks are configured.

9.5.1 News-Related Queries and Content Sources

For the experiments in this chapter, we use a set of 219 news-related queries from the period of April 11th to April 23rd 2012 (see Section 5.2). These news-related queries can be considered to be either event related or event unrelated. For example, an event related news query might be the query ‘north korea’ on the same day that the country is in the news. On the other hand, an event unrelated news query

Query Classification	Event-Related	# Queries	Proportion
Generic	✖	20	9.1%
Breaking	✔	74	33.8%
Recent	✔	38	17.4%
Long-Running	✔	29	13.2%
Other	✔	58	26.5%

Table 9.1: Statistics of the 219 news-related queries used during our evaluation.

might be the query 'CNN news'. From this 219 query set, 20 queries are event unrelated while the other 199 are related to an identifiable event. We use the 20 event unrelated queries to evaluate our integration of top event rankings by their importance. We use the other 199 event-related queries to evaluate the integration of news and user-generated content rankings for each of those queries.

To facilitate further analysis, the author manually classified each of the 219 queries into our news query taxonomy (see Section 4.3), i.e. into the four classes: generic; breaking; recent; and long-running. We also define a fifth class – denoted 'Other' – for queries that are event related, but where we cannot determine when the event driving the query originally occurred. The statistics of these queries are listed in Table 9.1.

For our evaluation experiments, we use a number of news and user-generated sources from which to rank documents to subsequently integrate into the Web search ranking. We use five different sources of content from the period of April 11th to April 23rd 2012. In particular, we use newswire articles from a wide variety of news providers tracked by the Blekko Web search engine, including CNN, Fox News, the New York Times and the Washington Post. For our second source, we use blog posts tracked by the Blekko Web search engine. Our third source is the Digg social news aggregator. For our fourth source, we use a sample of tweets provided by the Twitter streaming API. Our fifth and final source is the public encyclopedia Wikipedia. Further details regarding these sources can be found in Section 5.2.4.

9.5.2 Document Rankings

To generate the basic Web page ranking (R_W) into which we subsequently integrate news and user-generated content, we crawl the Google search engine[1] for the time of each query using its search API[2]. Notably, Google automatically returns Wikipedia pages for some queries. Since we consider Wikipedia to be a separate source of user-generated content, we filter out any Wikipedia pages that Google returns.

The Top Events Identification component within our news search framework (see Chapter 4) is responsible for producing rankings of events (I) represented by news articles for a point in time by their

[1] `http://google.com`
[2] `https://developers.google.com/custom-search/v1/overview`

current importance/newsworthiness. In Chapter 6, we proposed and investigated a series of approaches for ranking news articles their current importance and evaluated them on blog post and Twitter datasets. For the experiments within this chapter, we choose to use the real-time Twitter stream rather than blog posts to avoid any reporting lag in the blogosphere. We use the most effective of our unlearned news article ranking approaches for tweets (see Section 6.7), namely RWA. The newswire articles that we rank by importance come from our news article source. In particular, we use only newswire articles from the CNN news provider[1] that were published during a 24 hour period before the each query time. We use CNN since it is a major news provider that gives good coverage of events worldwide.

Within our news search framework described in Chapter 4, the generation of the news and user-generated content rankings ($R_{1..n}$) is the responsibility of the Ranking News-Related Content (RNRC) component. For these experiments, we use a series of document weighting models, in addition to the learning to rank approaches proposed in Chapter 8. For all rankings, only documents published before the query time are ranked. In general, we use the DPH document weighting model to rank documents, as it is a parameter free model that has been shown to be effective in a Web search setting (Santos, McCreadie, Macdonald & Ounis, 2010). For blogs and tweets, we employ the learning to rank approaches that we developed and showed were effective for those sources in Chapter 8. We describe how documents from each of our sources are ranked below:

- **News Articles**: The DPH document weighting model (Equation 2.9) is used to rank news articles for the user query.

- **Blog Posts**: The learning to rank approach described in Section 8.2.1 is used to rank blog posts, as this was shown to be effective in Section 8.5. The DPH document weighting model (Equation 2.9) is used as the blog post ranking sample, as described in Section 8.2.1.

- **Digg Posts**: The DPH document weighting model (Equation 2.9) is used to rank Digg posts for the user query. Duplicate posts are removed, since the same article can be posted multiple times to Digg.

- **Twitter Tweets**: The learning to rank approach described in Section 8.2.2 is used to rank tweets, since our experiments in Section 8.6 showed it to be effective. As described in Section 8.2.2, the DFReeKLIM document weighting model (Equation 2.10) with query expansion using the Kullback-Leibler (KL) term weighting model (Equation 2.16) is used as the tweet ranking sample.

[1]`http://cnn.com`

- **Wikipedia Pages**: The DPH document weighting model (Equation 2.9) is used to rank Wikipedia pages for the user query. The Wikipedia source also includes 'user talk' pages that discuss changes to individual articles. We filter these out of the ranking since they are not public-facing pages.

However, these ranking approaches do not guarantee that relevant documents will be returned. In particular, no relevant documents may exist, or the ranking approach may fail to return them. This might impact our evaluation, since users will quite naturally prefer rankings without irrelevant documents. Since our aim in this chapter is to evaluate whether the integration of our ranked content is useful, rather than to evaluate how effectively we are able to rank documents from each individual source, we take steps to minimise the risk that irrelevant documents might be integrated. In particular, we filter out non-relevant documents from each of the news and user-generated content rankings produced. This filtering was one the four tasks that we achieved using crowdsourcing and was described in detail earlier in Section 5.7.

9.5.3 Experiment structure

To structure the evaluation of ranking pairs produced by our framework, we break down our evaluation into two experiments. The first experiment examines whether end-users find that the integration of top event rankings for generic news queries leads to increased user satisfaction. In this case, we have crowdsourced workers compare rankings produced by the Google search engine for the 20 event unrelated/generic queries, to the same rankings with three events estimated as important for the time of each query added to the top of it (see Section 9.2).

Our second experiment evaluates whether users prefer rankings enhanced with content from news or user-generated sources to the unaltered Web search ranking. Notably, rather than integrating documents from all five sources of news and user-generated content available (news articles, the blogosphere, Twitter, Digg and Wikipedia) into a single ranking, we instead integrate and evaluate rankings from each source individually. The reason for this is that integrating all sources at once makes it difficult to determine what sources are actually increasing coverage for the query. Instead, for each query, we produce one enhanced ranking for each of the five news and user-generated sources and have users compare it to the unaltered Web search ranking.

Table 9.2 summarises each of our two evaluation experiments. From Table 9.2, note that our first experiment uses the 20 event unrelated/generic news-related queries, while our second experiment uses 199 event related news queries. Since our second experiment considers each source individually, the total number of ranking pairs is five times the number of queries.

Experiment	Ranking Sub-Function	Enhanced Ranking	# of Queries	# of Ranking Pairs
1	$Integration_{News}(R_W, I)$	Web + Top News Events	20	20
2	$Integration_{RE}(Q, R_{1..n}, R_W, S)$	Web + Single Source	199	995

Table 9.2: News-related content integration evaluation experiment statistics.

9.5.4 Worker Instructions

For each of the two experiments, we provide instructions to our workers on how to complete the assessment task. Recall from Section 9.3 that the instructions provided to the workers come in two parts, namely the instruction block and the guidelines block. The instruction block that we provide for both experiments is shown below:

```
"Below (in bold text) is a user search query, a short description of what
the user is searching for and three potential search engine rankings returned
for that query by a Web search engine. Each ranking is a button. You need to
select the ranking that best satisfies the query by clicking it. Of the three
rankings, one has a red border. Do NOT click the red bordered ranking, we use
this ranking to detect bots, any HIT where a red bordered ranking has been
selected will be automatically rejected."
```

The guidelines block varies depending upon the experiment. The guidelines for the first experiment are:

```
> Most of the queries are related to an event or news story that had just
  happened at the time the query was made.
> We are primarily interested in whether the additional results added to one
  of the rankings make it better or worse in terms of coverage for the query.
> We are not trying to assess the styling or look and feel of the ranking.
> You may see the same query/description pair multiple times. This is expected
  and the rankings may be different. These cases are where the same query was
  made on different days.
```

while the guidelines for the second experiment are:

```
> Most of the queries are related to current news in some manner, but it is
  often not clear if the user was interested in a particular story.
> We are primarily interested in whether the additional results added to one
  of the rankings make it better or worse in terms of coverage for the query.
> We are not trying to assess the styling or look and feel of the ranking.
> If both of the rankings are identical (no results are added to either ranking,
  click on either to continue (do not click the red bordered ranking though!).
```

9.5.5 Crowdsourcing Configuration and Statistics

We use the crowdsourcing marketplace Amazon's Mechanical Turk (MTurk) to recruit workers for our evaluation. We have ten individual workers assess each document ranking pair. This is larger than the three workers used previously for relevance assessment tasks (see Sections 5.4 to 5.7), since we wish to aggregate the opinions of a wider user-base. Furthermore, prior work in the field has indicated that collecting ten redundant assessments and taking the majority vote is sufficient to overcome noise introduced by low quality work, even should no other worker validation be performed (Ipeirotis, 2011). We paid workers based on the estimated amount of time it would take to judge each query at a rate of US $2 per hour. In particular, we paid US $0.01 per document pair, since assessment requires that the user browses the two rankings and makes a single click. Each evaluation task (MTurk HIT) involved assessing five different ranking pairs. Hence, each HIT was costed at US $0.05. We use the in-built worker validation described in Section 9.3 to detect bots and spammers. Any HIT where a red-bordered ranking was selected was rejected without payment and re-submitted for another user to complete.

In terms of the two experiments, 23 individual workers assessed the first experiment, while 57 workers assessed the second experiment. Both experiments were completed individually in under 36 hours. Automatic validation (based on clicking on red-bordered rankings) resulted in the rejection of 33.9% and 9.5% of assignments for each of the two experiments respectively. Inter-worker agreement was 0.34 Cohen κ for the first experiment and 0.33 Cohen κ for the second experiment. Note a lower agreement is expected in this case due to both the larger number of workers attempting each task and greater scope for disagreement between workers due to individual preferences.

9.6 Evaluation: Integrating News and User-generated Content

In this section, we evaluate the News-Related Content Integration (NRCI) component of our news search framework. In particular, Section 9.6.1 investigates whether the integration of top event rankings into the Web search results for news-related queries leads to increased user satisfaction. In Section 9.6.2, we examine whether end-users find that the addition of content ranked from any of our news or user-generated content sources increases coverage for news-related user queries. Section 9.6.3 investigates whether end-users prefer rankings integrated with user-generated content to rankings integrating news articles for some news-related user queries. For example, this might occur when no relevant newswire articles have yet been published. Finally, in Section 9.6.4, we examine whether the addition of user-generated content can enable news-related queries to be satisfied in a more timely manner.

9.6.1 Integrating Top Events

We begin our evaluation of the NRCI component of our news search framework by examining whether end-users prefer rankings with integrated top events in comparison to unaltered Web search rankings for generic news queries. This type of query is clearly news-related, but is not linked to any identifiable news event, e.g. the query ‘CNN news’. One way that a news vertical might be able to better satisfy

Aggregation	# Prefer Web	# Prefer Web + Events
None (Raw Preferences)	149	51
Majority Vote	20	0

Table 9.3: Number of end users that prefer the Web search ranking with the integrated top events ranking in comparison to the Web search ranking unaltered.

these queries is by integrating a ranking of top news articles into the Web search results. Indeed, in Section 4.5, we showed that Web search engines such as Microsoft Bing[1] already do so. In this section, we use the content integration approach described in Section 9.2 and our framework for comparing rankings that we described in Section 9.3, to evaluate whether end-users prefer rankings that contain a listing of current events for generic news queries. This section corresponds to the first of our two experiments, as described in Section 9.5.3.

Table 9.3 reports the number of users that prefer the Web search ranking with the integrated top events in comparison to the Web search ranking unaltered for the 20 generic news queries. From Table 9.3, we see the following two points of interest. First, examining the raw preferences of users, we see that users preferred the ranking with added top news events 25.5% of the time. This indicates that some users do indeed find that the addition of top news events helpful, however the majority do not. Secondly, if we take the majority vote across the ten users that assessed each document ranking pair, we see that the majority always favours the Web search ranking unaltered. This result indicates that most end-users find that the inclusion of a top events ranking adds little to the Web search ranking. This is a surprising result, since we have observed major Web search engines integrating top events into their rankings in a similar manner (see Section 4.5). One possible explanation for this result is that the rankings of top events used were of insufficient quality to be useful. However, in Section 6.7, we showed that news story ranking using tweets appeared to be effective. Furthermore, we subsequently analysed the top event rankings produced for the 20 generic news queries, and the top events ranked were important at the time. Hence, we rule out event ranking quality as the cause. Instead, it may be that for the queries tested, top event rankings were not needed, or that the manner in which they were presented could have been improved. Indeed, we identify a detailed examination of how to integrate top events into the Web search ranking as future work. In answer to our first research question, we conclude that for the queries and presentation method tested here, the integration of a top event ranking into the Web search results for the generic news-related queries did not increase user satisfaction.

[1]`http://bing.com`

Source	# Queries	# of times that users prefer the Web search ranking	# of times that users prefer the Enhanced ranking	Total Assessments
News Articles	145 (72.8%)	561 (38.7%)	**889 (61.3%)**	1,450
Blogosphere	183 (92.0%)	828 (45.2%)	**1,002 (54.8%)**	1,830
Digg	19 (9.5%)	84 (44.2%)	**106 (55.8%)**	190
Twitter	191 (96.0%)	789 (41.3%)	**1,121 (58.7%)**	1,910
Wikipedia	155 (77.9%)	748 (48.3%)	**802 (51.7%)**	1,550
All		3,010 (43.4%)	**3,920 (56.6%)**	6,930

Table 9.4: Number of preferences for both the Web and Enhanced rankings.

9.6.2 Integrating News and User-Generated Content vs the Web Search Ranking

To answer our second research question, we investigate whether the majority of users in our crowd-sourced study prefer the rankings enhanced with news and user-generated content over the Web search ranking. In particular, for our second experiment, we have our evaluation framework generate rankings enhanced with content from one of our five sources for the set of 199 queries related to newsworthy events. For each query, we then have 10 different users compare the enhanced ranking to the Web search ranking for that query. We asked each user to select the ranking that they preferred for the query. In the guidelines provided to each user, we clarify that they should focus on whether the additional results added to one of the rankings make it better or worse in terms of coverage for the query and that they should not be overly influenced by the styling or look and feel of either ranking.

Table 9.4 reports the number of queries for which enhanced rankings were produced and the number of users that preferred those enhanced rankings in comparison to the Web search rankings. Sources for which more users preferred the enhanced ranking than the Web search ranking are highlighted (bold font). Recall from Section 9.3 that our evaluation framework skips cases where no documents are integrated (either because no relevant documents were returned for the query or because any retrieved documents would not make the top ten results). For this reason, the number of queries that are evaluated for each source is less than 199. From Table 9.4, we observe the following. First, in terms of the number of queries for which an enhanced ranking could be generated, we see that our news, blog post, Twitter and Wikipedia sources were able to enhance between 72.8% and 96.0% of all (199) news-related queries tested. The exception was the Digg source that only added content to the Web search ranking for 9.5% of the 199 queries. From further analysis, the Digg corpus did in fact retrieve documents for many queries, but the scores assigned to those documents were not sufficiently high for them to make the top results. These low scores resulted from a low source score being assigned to Digg by the CRCS (Equation 2.26) resource selection algorithm, rather than being caused by the documents retrieved themselves. Future

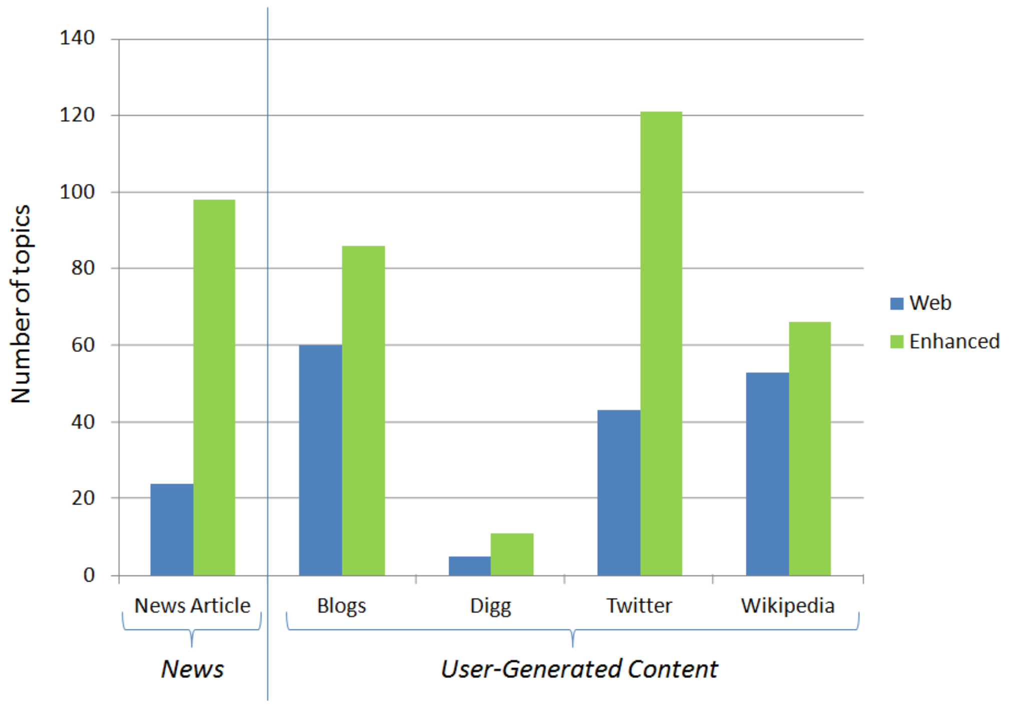


Figure 9.4: The number of topics where the majority of users preferred the Web search ranking or the enhanced ranking that integrates each source.

work might examine more effective integration functions than the simple baseline approach described in Section 9.2.

Secondly, examining the number of times that users prefer each ranking, we observe that in general, users appear to prefer the enhanced rankings over the Web search rankings. Indeed, for all five sources, more enhanced rankings were clicked than Web search rankings. For instance, of the 1,910 document ranking pairs displayed to users for the Twitter source, 58.7% of users clicked on the enhanced ranking, while 41.3% clicked on the Web search ranking. Within the context of this thesis, the result is promising, since it indicates that end-users do indeed find that the addition of user-generated content increases coverage for news-related queries and therefore can aid when satisfying those queries in real-time.

However, measuring only the number of clicks does not account for disagreement between users for each query. Indeed, recall that each query is assessed by ten different users. To determine whether the addition of user-generated content increases user satisfaction overall, we calculate the majority vote for each of the query topics. For example, if three users select the Web search ranking and seven select the enhanced ranking, then we consider the enhanced ranking preferred over the Web search ranking.

Figure 9.4 shows the number of query topics for each source for which the majority of users preferred either the Web search ranking or the enhanced ranking. For each source, the first column indicates the number of topics where users prefer the Web search ranking, while the second column indicates the number of topics where the enhanced ranking was preferred. From Figure 9.4, we see that, again, for all sources, the majority of users prefer the enhanced rankings to the Web search rankings. However, we also see that the differences between each source are more pronounced. Integrating news articles into the Web search results improved the ranking for the majority of users for almost 100 of news-related queries tested (approximately 50%). Moreover, out of those queries for which news article content was integrated (145 out of the 199), over 80% were improved. This shows that when newswire articles are returned for a news-related query, they are most often relevant and good documents to integrate. Furthermore, we observe that integrating tweets improved an even larger proportion of the overall query set than integrating newswire articles (121 query topics as opposed to 99), although the proportion of queries where tweets were added and users opted for the unaltered ranking was also higher (26.2%). This indicates that Twitter has better coverage of the events referred to in our query topics than the newswire article stream, but that a larger proportion of the content returned from Twitter is irrelevant. Next, we see that the integration of blog posts and Wikipedia pages resulted in a similar numbers of enhanced and unenhanced rankings being preferred. In particular, integrating blog posts aided for 43% all queries, but also harmed 30% of the remaining queries. Similarly, integrating Wikipedia pages, improved 33% of all queries and harmed 26.6%. This shows that blog posts and Wikipedia pages are still able to better satisfy end-users for some news-related queries, but they have less coverage of the events described by our topics than either the newswire article or Twitter streams. Of the queries for which additional content from Digg was integrated, 69% were improved, however this comprises only 5.5% of the total query set.

In general, we can conclude that as expected, integrating news articles is effective at increasing coverage for news-related queries, with rankings for 49.2% of news-related queries being improved. We have also shown that for many news-related queries, user-generated content is also useful. Indeed, rankings integrating Twitter tweets were preferred more often than those integrating newswire articles (60.1%). Hence, in answer to our second research question, end-users do indeed find that the integration of news and user-generated content can improve coverage for some news-related user queries.

9.6.3 Integrating User-Generated Content vs News Content

Next, we examine where integrating user-generated content can be more beneficial than integrating news articles. Indeed, if at the time each query was made relevant newswire articles are available, then

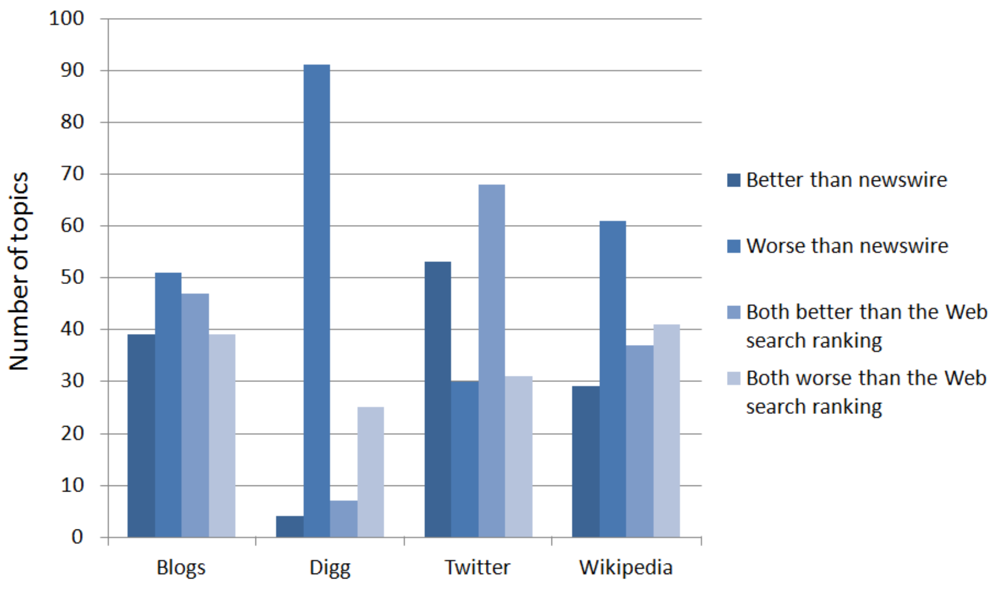


Figure 9.5: Topic comparison between rankings enhanced with news articles and those enhanced with other user-generated content sources.

user-generated content may not be required to satisfy the query. To examine this, we contrast the query topics that benefited from the integration of news articles to those that are benefited by the integration of user-generated content. For a query, if relevant user-generated content was found, integrated and the ranking was preferred by the majority of users, then user-generated content can be said to better satisfy that query than when integrating nothing. However, that same query may be equally or better satisfied by the integration of newswire articles. On the other hand, if there are news-related queries that user-generated content can satisfy, but newswire articles cannot (at the time those queries were made), then we can show that user-generated content is necessary to satisfy those queries.

Figure 9.5 shows the number of query topics where integrating content from each user-generated source resulted in a better or worse ranking than the ranking enhanced with news articles for those topics. For each of the four user-generated content sources, the first column indicates the number of query topics for which user-generated content aided and newswire did not, while the second column indicates the reverse case. The latter two columns denote the number of query topics where either both user-generated content aided or neither did. From Figure 9.5, we observe the following. First, all of the four user-generated content sources improve some query topics. For instance, the integration of blog posts improved the Web search ranking for 39 topics, where the integration of newswire articles did not. Similarly, integrating Twitter tweets aided for 53 topics, while Wikipedia pages aided 29 topics. This indicates that a large proportion of our news-related queries were both not well served by the news articles at the time but could be satisfied by user-generated content. For instance, for the query 'lionel richie' issued on the 13th of April 2012, the Twitter stream returned tweets reporting that music star Lionel Richie was facing a large tax bill over allegations he owed more than \$1 million to the U.S. government. On the other hand, our newswire stream return no relevant articles for that query and time.

However, from Figure 9.5, we also see that there are news-related queries well served by newswire articles but not by user-generated content. For instance, 51 topics were aided by the integration of newswire articles, but were not aided by the integration of blog posts (second column of Blogs in Figure 9.5). To illustrate, on the 16th of April 2012 the query 'irs' was popular. This query refers to U.S. internal revenue service tax deadline, which was extended that day. In this case, our newswire stream returned articles about the deadline extension and implications, but the blog post stream only returned unrelated content[1]. Finally, we observe that the integration of tweets in particular benefited many topics in comparison to newswire articles. This may be an indication that the queries for which newswire article integration did not help, are those where news has broken first in Twitter and related newswire articles had not yet been published. We examine this further in the next sub-section. In answer to our third research question, our results suggest that there are a large number of news-related queries that can be better satisfied by user-generated content as opposed to newswire articles and vice versa.

9.6.4 Integrating User-Generated Content for Breaking and Recent Events

Finally, we examine whether user-generated content enables us to satisfy some news-related queries in a more timely manner than if we just used newswire articles. To this end, we examine the query topics for which the integration of user-generated content aided when the integration of newswire articles did not, in terms of our news-query taxonomy (see Section 4.3). The aim is to determine whether integrating user-generated content, particularly Twitter tweets, aids more for breaking news queries than for other types of news-related query. Table 9.5 reports the proportion of query topics where integrating user-generated content was more effective than integrating newswire articles for the breaking news queries, recent news queries and queries relating to longer running events, in our 199 query set. For example, a value of 64% for Twitter on Recent queries means that for the set of queries judged to be about recent events (news events that occurred between 12 and 48 hours previously), 64% where improved by the integration of tweets but not by the integration of newswire articles. Note that we omit results from Digg

[1]A post on the 'ten hottest IRS Commissioners'

User-Generated Content Source	Breaking	Recent	Long-Running
Blogs	58%	58%	73%
Twitter	89%	64%	57%
Wikipedia	71%	36%	44%

Table 9.5: Proportion of query topics where integrating user-generated content was more effective than integrating newswire articles.

in this table due to the very small number of queries where Digg improved over integrating newswire articles (see Section 9.6.2).

From Table 9.5, we see that when integrating blog posts, 58% of queries relating to breaking or recent events received better coverage. This increased to 73% for long running events, indicating that blog posts are more valuable in cases where an event has been running for some time. On the other hand, we see that integrating tweets into the Web search ranking is more effective the more recent the event is. In particular, 89% of breaking news queries were improved, while a smaller proportion of queries relating to only recent or long running news events were improved (64% and 57% respectively). For example, a reoccurring query in our dataset was the query 'levon helm', referring to American rock multi-instrumentalist and actor who was fighting cancer at the time. When this query was issued on the 19th of April 2012, Twitter returned tweets reporting his death, while our newswire stream only returned articles reporting that he was in the final stages of cancer. This illustrates the value that Twitter can bring from a content integration perspective for breaking news queries where no newswire articles have yet been published. Finally, when integrating Wikipedia pages, we observe that a higher proportion of breaking news queries than those relating to recent or long running events were improved. Indeed, the rankings for 71% of breaking news queries were improved, in contrast to only 36% and 44% for queries relating to longer running events. This indicates that users find the background information provided by Wikipedia pages are more useful for breaking news events than for longer running events that they may already know about. In answer to our fourth research question, we have shown that the addition of user-generated content does enable some news-related queries to be satisfied in a more timely manner.

9.7 Conclusions

In this chapter, we investigated the final component of our news search framework, namely News-Related Content Integration. We first defined a function to integrate news-related content into the Web search ranking. This function either integrates rankings of top events by their importance or rankings of news and user-generated content for the user query. We then developed a framework for comparative evaluation to evaluate rankings with additional content integrated. Using crowdsourced workers as a surrogate for end-users, we evaluated the integration of news-related content increased user satisfaction and coverage for some news-related queries.

In particular, in Section 9.2, we proposed a simple baseline function M for integrating news and user-generated content into the Web search ranking. This function enables documents ranked from different sources to be integrated, even though the background statistics of each source differ. Meanwhile, in Section 9.3, we described an evaluation framework to enable end-users to compare different rankings for the same query. This framework is responsible for displaying each query and a description of the information need underlying that query to the user, in addition to rendering the different rankings to be assessed.

In Section 9.4, we defined the research questions that we investigated in this chapter, while our evaluation methodology that uses the proposed integration function M and evaluation framework was detailed in Section 9.5. In particular, we divided our evaluation into different two experiments. The first experiment examined the integration of top events into the Web search ranking for generic news queries, while the second experiment investigated the integration of news and user-generated content ranked for queries related to newsworthy events. For both experiments, we used crowdsourced workers to compare the Web search ranking for news-related queries to rankings enhanced with additional content for those same queries.

Through experimentation detailed in Section 9.6, we showed that the majority of workers were not better satisfied by the integration of top event rankings for generic news-related user queries e.g. ‘CNN news’, with users preferring the Web search ranking unaltered (see Table 9.3). On the other hand, for event-related queries, more workers preferred rankings enhanced with user-generated content than the Web search ranking (see Table 9.4). Furthermore, when considering the majority vote between all workers for each query, our results showed that almost half of all the queries tested were better covered by the addition of newswire articles, while 61% were similarly better covered by the inclusion of tweets (see Figure 9.4). Through an analysis of the queries for which either news and user-generated content streams aided, we showed that there were many queries that could be better satisfied through the integration of user-generated content, but not newswire articles (see Figure 9.5). Indeed, over a quarter of the queries tested could be better satisfied by integrating Twitter tweets, while for those same queries, no relevant newswire articles were available. Finally, we examined the types of news-related queries that were aided by the integration of user-generated content but not newswire content. Our results indicated that different user-generated content sources are better at satisfying different types of query (see Section 9.6.4). Indeed, Twitter was shown to favour breaking news queries, while blog posts aided for a larger proportion of queries relating to longer running events.

In our thesis statement (see Section 1.3), we postulated that the appropriate integration of user-generated content into the Web search ranking would increase the coverage of events that users are looking to know more about. Our experiments in this chapter confirm that end-users prefer rankings integrating blog posts, tweets and Wikipedia pages to unaltered Web search rankings for news-related queries. Hence, we conclude that user-generated content can indeed aid in satisfying news-related queries by providing relevant content to display, especially when newswire content is not yet available or is insufficient. In the next chapter, we close this thesis by summarising the conclusions and outcomes from each of the individual chapters, in addition to providing possible new research directions uncovered by this work.

Chapter 10

Conclusions and Future Work

10.1 Contributions and Conclusions

This thesis proposed the use of user-generated content to aid in satisfying news-related queries within a universal Web search engine. During the course of this thesis, we have proposed a novel news search framework that defines four components, which are needed to deploy real-time news vertical search. We have drawn insights from a broad range of experiments examining each framework component and concluded upon the potential and scope for enhancements that can be brought about by the leverage of user-generated content for real-time news search. The remainder of this section discusses in more detail the contributions and conclusions of this thesis.

10.1.1 Contributions

The main contributions of this thesis are as follows:

- In Chapter 4, we proposed a novel news search framework that describes the search process of a universal Web search engine and supports a news vertical. This framework defines four components that are needed to deploy real-time news vertical search. In particular, the Top Events Identification component (see Section 4.5) produces rankings of events by their current importance, which can be used to aid when classifying news-related queries and/or be integrated into the Web search results to better satisfy the user. The News Query Classification component (see Section 4.6) identifies news-related queries in real-time as they are submitted to the search engine. The Ranking News-Related Content component (see Section 4.7) produces rankings of documents from different news and user-generated content sources for queries identified as news-related. Finally, the News-Related Content Integration component (see Section 4.8) integrates news and user-generated content into the Web search ranking for display to the user. The challenges that arise when tackling each component in a real-time news search setting were discussed and the use of user-generated content to solve these challenges was motivated (see Sections 4.5 to 4.8). This novel news search framework is a key contribution, in that it identifies the areas of the search process that might benefit from user-generated content, divides our evaluation into individual components that can be assessed in isolation or as a whole, and provides a structure for the thesis in general. Indeed, to the best of our knowledge, no prior work has examined the news search process in this detailed manner.

- In Chapter 5, we presented the diverse datasets that were used to validate our thesis statement. In particular, we used four TREC derived datasets (see Section 2.4.3), in addition to developing six new datasets to facilitate our evaluation of each framework component (see Table 5.2). Each of these datasets contains one or more aligned corpora of user-generated content that we used in the subsequent experimental chapters, either as a source of evidence to enhance the search process, or as a source of documents to display to the user. Moreover, to create these datasets, we extensively leveraged the emerging medium of Crowdsourcing to produce fast and cheap assessments (see Table 5.7). Indeed, Chapter 5 contributed four novel crowdsourcing studies using Amazon's Mechanical Turk, totalling over 60,000 individual assessments and a total cost of $1,442.93 (US dollars). In each study, we described in detail how we developed effective new interfaces for crowdsourcing assessments and then validated the resulting assessments in real-time, enabling the rejection of poor quality work (see Sections 5.4 to 5.7). Furthermore, we closed each of these four crowdsourcing studies by evaluating the quality of the final assessments produced, showing through high levels of inter-worker agreement or accuracy against a ground truth, that the resultant assessments were of sufficiently good quality (see Sections 5.4.5, 5.5.5, 5.6.5 and 5.7.5).

- The first component of our news search framework, namely Top Events Identification, was investigated in detail in Chapter 6. A novel automatic real-time approach for the identification of important current news-related events based upon a voting paradigm was proposed (see Section 6.2.1). This approach leverages the volume and relatedness of discussions within different user-generated content streams to rank events. We also proposed a new learning to rank framework for the real-time identification of important events (see Section 6.3). This new framework allows existing event ranking approaches to be expressed in terms of four elements, namely: the story representation; the document ranking approach; the document sub-feature; and the aggregation model. Combinations of these elements form features for real-time story ranking. Multiple effective story ranking approaches are expressed in this way and then combined to provide better estimations of the importance of each event. Using real-time discussions from both tweet and blog post streams over five datasets (see Table 5.3), these approaches were thoroughly evaluated to determine how effectively they can rank events represented by newswire articles in comparison to rankings produced by newspaper editors (see Sections 6.6 and 6.7). Whether these approaches tend to promote recent or more long-running events when deployed on tweet streams was also examined (see Section 6.7.6).

- In Chapter 7, we investigated the News Query Classification component of our news search framework, with the aim of leveraging user-generated content streams to more accurately classify news-related queries in real-time. We proposed a new real-time classification approach that leverages recent discussions from multiple aligned news and user-generated content streams (see Section 7.2). When a query is issued, this approach extracts novel real-time signals regarding related and recent discussions within each stream – including importance estimations derived from the new approaches we proposed in Chapter 6 – combining them to estimate whether that query is indeed news-related. By combining signals from multiple streams, each with their own characteristics and audience, this approach aims to provide a better overall accuracy and more timely classification of queries relating to breaking events. Two datasets containing parallel newswire, blog post, tweet, Wikipedia edit, query log and Digg post streams (see Table 5.4) were used to evaluate whether each of these streams enable faster and more accurate classification of news-related queries (see Sections 7.6 and 7.7). We contributed an in-depth analysis of the streams and features that are the most effective for real-time news query classification (see Sections 7.6.2 and 7.7.2), in addition to how classification accuracy and the most effective streams and features change over time as events mature (see Sections 7.6.3 and 7.7.3).

- To facilitate the integration and display of up-to-date and relevant news-related content to the user when they submit a news-related query, content needs first to be ranked for that query. The Ranking News-Related Content component of our news search framework is responsible for ranking documents from different news and user-generated content sources. Chapter 8 examined how to effectively rank two types of user-generated content, namely blog posts and tweets. Learning to rank approaches that leverage the unique characteristics of these sources were proposed (see Section 8.2). These approaches aim to better capture the aspects of blog posts and tweets that may make them relevant, by representing them as sets of novel real-time stream features. Two standard

TREC datasets (see Table 5.5) were used to evaluate whether the proposed approaches were better able to effectively retrieve tweets or blog posts for (news-related) user queries than traditional document weighting models and other recent approaches from the literature (see Sections 8.5.1 and 8.6.1). The features that were the most effective when ranking blog posts and tweets using our learning to rank approaches were also identified (see Sections 8.5.3 and 8.6.2).

- A novel large-scale user-study examining the last News-Related Content Integration component of our news search framework was presented in Chapter 9. This large user-study evaluated whether the integration of user-generated content into the Web search ranking could better satisfy end-users than returning unaltered Web search rankings. In particular, building upon our earlier crowd-sourced studies (see Chapter 5), we used Amazon's Mechanical Turk workers to compare Web search rankings to alternate rankings integrating news and user-generated content for news-related queries. An adaptation of resource selection techniques was proposed to perform the integration of content into Web search ranking (see Section 9.2). We developed a novel framework for comparative ranking evaluation, that simulates the ranking presentation by major Web search engines and facilitates preference assessment across rankings by workers (see Section 9.3). Through a comparative evaluation involving over 6,900 preference judgements, we examined whether end-users find that the integration of top events, newswire articles, blog posts, Digg posts, Twitter tweets and Wikipedia pages, better satisfy end-users than unaltered Web search rankings for news-related queries (see Section 9.6). The types of queries that particularly benefit from the integration of user-generated content were also examined (see Section 9.6.4).

10.1.2 Conclusions

This section discusses the achievements and conclusions of this work.

Effectiveness of Top Events Identification From the experiments detailed in Chapter 6, we conclude the following. When using blog posts to estimate event importance, the voting-based approaches for top events identification proposed in Section 6.2 were shown to be effective in comparison to the best systems submitted to the TREC 2009 and 2010 Blog track top news stories identification task (see Table 6.4) over two datasets ($BlogTrack_{2009}^{TopNews}$ and $BlogTrack_{2010}^{TopNews-Phase1}$). In particular, relevance weighted aggregation (RWA) and its $GaussBoost$ enhancement achieved the highest news article ranking performance observed on blogs (see Sections 6.6.1 and 6.6.2). Next, when using tweet streams to estimate the importance of each story, our RWA or RWAN approaches (see Section 6.2) that consider the relevance score of each tweet were shown to outperform our voting-based baseline by a statistically significant margin over all three datasets tested ($Twitter^{TopNews-NYT}_{Dec2011}$, $Twitter^{TopNews-NYT}_{Jan2012}$ and $Twitter^{TopNews-Reuters}_{Jan2012}$) (see Section 6.7.1). Moreover, the $MaxBurst$ enhancement to these approaches – that promotes stories associated with large bursts of discussion over time – further increased performance by a statistically significant margin (see Section 6.7.3). These results show that modelling event importance from user-generated content as a voting process is effective and that both blogs and tweets are good sources of evidence to use for real-time event ranking.

Chapter 6 also proposed a learning to rank stories (LTRS) framework that facilitates the combination of multiple event ranking approaches, with the aim of better estimating the importance of events. Experimental results from Chapter 6 showed that when using the two blog stream datasets, LTRS was significantly more effective than the best event ranking approaches that it uses, which include our voting-based approaches (see Section 6.6.3). However, on the three datasets that use tweet streams, LTRS effectiveness varied depending upon the dataset used for training (see Section 6.7.4). Through analysis of the features selected by LTRS when using both blog post and tweet streams, the ranking approaches that consider only the last day's blog posts and tweets were shown to produce the most effective event ranking features (see Sections 6.6.4 and 6.7.5). From these results, we conclude that LTRS is overall more effective than our voting-based approaches when estimating the importance of events in real-time using the blogosphere.

This thesis postulated that the most important events at a point in time could be identified using user-generated content. From the experiments in Chapter 6, we have shown that this is indeed the case when using blogs and tweets.

Accuracy of News Query Classification Chapter 7 examined the effectiveness of real-time news query classification using features extracted from parallel news and user-generated content streams over the $NQC_{May2006}$ and $NQC_{Apr2012}$ datasets. When comparing to a baseline news query classifier that uses only newswire articles to identify emerging news queries, our results showed that classifiers which leverage user-generated content are often more effective (see Sections 7.6.1 and 7.7.1). Furthermore, classifiers that combine multiple streams of news and user-generated content were shown to outperform news stream-only classifiers by a large and statistically significant margin (see Tables 7.4 and 7.9). Through analysis of the features from each stream selected by our classifiers, we have shown that tracking the frequency of query terms and the number of related pages to each query over time, resulted in the most effective features for both datasets (see Sections 7.6.2 and 7.7.2).

From the analysis of classification accuracy over time as events mature, we have shown that when the queries were first issued, they could be more accurately classified using user-generated query logs than newswire streams (see Section 7.6.3). However, as time passed, the classification accuracy using newswire increased, while the classification accuracy using query logs decreased (see Section 7.6.3). We also observed that classifiers using both news and user-generated content streams were able to consistently outperform single stream classifiers, both when the queries in that dataset were first issued and over time (see Section 7.7.3).

In our thesis statement (see Section 1.3), we hypothesised that user-generated content could increase the accuracy of a real-time news query classifier within a universal Web search engine. Based upon our experiments in Chapter 7, we conclude that user-generated content streams in combination can significantly improve a news query classifier that relies only upon newswire providers to identify news-related queries. This, in turn, enables a universal Web search engine that supports a news vertical to better satisfy news-related queries relating to both breaking and long-running events by classifying them more accurately.

Effectiveness when Ranking News-Related Content Once a news-related query has been identified, relevant documents from different news and user-generated content sources are ranked, such that they can be subsequently integrated and displayed to the user. Chapter 8 investigated how to effectively rank two types of user-generated content for news-related queries, namely; blog posts and tweets. Learning to rank approaches for ranking these two types of content in a real-time manner were proposed (see Section 8.2). Through experimentation in Section 8.5 on the $BlogTrack_{2010}^{TopNews-Phase2}$ dataset (see Table 5.5), we have shown that the proposed learning to rank approach for blog post ranking outperformed both effective documents weighting models and the best TREC 2010 systems for the same task (see Section 8.5.1). Furthermore, it was shown that this approach maintains or increases its effectiveness as new blog posts are published over time (see Section 8.5.2). Through an examination of the features selected by the proposed learning to rank approach for blog post ranking, the relatedness of each blog post to the query topic using the DPH document weighting model (see Section 2.3.3) was the most effective indicator of relevance, followed by the presence of advertising or copyright statements (see Section 8.5.3).

Later in Section 8.6, the proposed learning to rank approach for tweet ranking was evaluated. Experimental results when testing on the $MicroblogTrack_{2011}$ dataset (see Table 5.5) showed that the approach significantly outperformed effective document weighting models enhanced with real-time query expansion techniques (see Section 2.3), which in turn outperformed the best TREC 2011 approaches for the task (see Section 8.6.1). Subsequent analysis of the features selected by the approach indicated that the most influential were the retrieval score for the tweet, followed by the retrieval score for any document linked to from that tweet (see Section 8.6.2). This illustrates how the approach is able to combine evidence from both the tweet text and linked documents to produce a more effective tweet ranking.

In our thesis statement (see Section 1.3), we postulated that the appropriate integration of user-generated content into the Web search ranking would increase the coverage of events that users are looking to know more about. To achieve this, effective approaches to find and rank documents are needed, such that irrelevant documents are not integrated. From our experiments in Chapter 8, we conclude that the proposed learning to rank approaches are effective for ranking user-generated content from the blogosphere and Twitter. These approaches enable blogs and tweets to be ranked effectively within a universal Web search engine for news-related user queries, which can subsequently be integrated into the Web search ranking for display to the user.

Better Satisfying News Queries by Integrating User-generated Content In Chapter 9, we examined whether the integration of news and user-generated content could better satisfy end-users for news-related queries. Through a large crowdsourced user study, we showed that end-users found that integrating rankings of important events (represented by newswire articles) did not aid for generic news-related queries e.g. ‘CNN news’, with users preferring the Web search ranking unaltered (see Section 9.6.1). In contrast, users prefered rankings integrating news articles for half of the event-related news queries tested (see Section 9.6.2). We also showed that for many news-related queries, users preferred rankings that integrate blog posts, tweets and Wikipedia pages to the unaltered Web search ranking (see Table 9.4). Most notably, rankings integrating Twitter tweets were preferred more often than those integrating newswire articles (see Figure 9.4). Furthermore, we showed that there were many queries where users found that integrating user-generated content was more useful than integrating newswire articles (see Section 9.6.3). Indeed, over a quarter of the queries tested could be better satisfied by integrating Twitter tweets, while for those same queries, no relevant newswire articles were available (see Figure 9.5). Finally, we examined the types of news-related queries that were aided by the integration of user-generated content but not by the integration of newswire articles. Our results indicated that different user-generated content sources are better at satisfying different types of query (see Section 9.6.4). For instance, integrating tweets into the Web search ranking was more effective the more recent the event the user was searching about was. In particular, 89% of breaking news queries were shown to be improved, while a smaller proportion of queries relating to only recent (64%) or long running (57%) news events were shown to be improved (see Table 9.5). Meanwhile, when integrating blog posts, 58%

of queries relating to breaking or recent events were improved (see Table 9.5). This was shown to increase to 73% for long running events (see Table 9.5), indicating that blog posts are more valuable in cases where an event has been running for some time.

In our thesis statement (see Section 1.3), we postulated that the appropriate integration of user-generated content into the Web search ranking would increase the coverage of events that users are looking to know more about. Our experiments in Chapter 9 confirm that, in general, end-users prefer rankings integrating blog posts, tweets and Wikipedia pages to unaltered Web search rankings for news-related queries. Hence, we conclude that user-generated content can indeed aid in satisfying news-related queries by providing relevant content to display, especially when newswire content is not yet available or is insufficient.

10.1.3 Thesis Conclusions

This thesis stated that user-generated content could aid in satisfying news-related queries submitted to universal Web search engines in real-time. We proposed a novel news search framework comprised of four components and argued that user-generated content could improve the performance of each of these components. In Chapters 6-9, we empirically investigated each of these components in turn and concluded the following:

- Top Events Identification can be facilitated by using user-generated blog post and Twitter streams to produce event importance estimations (see Table 6.4 in Section 6.6 and Table 6.8 in Section 6.7).
- News Query Classification can be made more timely and accurate by leveraging multiple news and user-generated content streams for classification evidence (see Table 7.4 in Section 7.6 and Table 7.9 in Section 7.7).
- Ranking News-Related Content from the blogosphere and Twitter can be made more effective by incorporating the unique aspects and characteristics of these types of user-generated documents as features (see Table 8.3 in Section 8.5 and Table 8.6 in Section 8.6).
- News-Related Content Integration can better satisfy end-users by integrating user-generated blog posts, tweets and Wikipedia pages into the Web search ranking (see Table 9.4 and Figure 9.4 in Section 9.6).

Hence, we conclude that user-generated content can indeed aid when satisfying news-related queries submitted to universal Web search engines in real-time. In particular, user-generated content enables news queries relating to breaking and long-running events to be more accurately classified and the resultant rankings to be enhanced with additional content that end-users find useful.

10.2 Directions for Future Work

This section discusses several directions for future work related to the real-time satisfaction of news-related queries in a universal Web search engine.

Event Categorisation When we developed the top event identification assessments for our experiments using blog posts (see Section 5.4), workers assessed newswire articles as important or not for specific news categories. Indeed, we deemed it easier for workers to assess importance with respect to a specific news category than importance in general. However, the approaches that we proposed to identify top events in Chapter 6 do not model these different news categories. On the other hand, estimating event importance on a per-category basis has the potential to produce more useful overall estimations, since the scale of importance may differ between categories (Ounis *et al.*, 2010). For instance, financial news stories may be under-reported in social media, while being the subject of many queries to Web search engines. Similarly, the Celebrity news may be over-reported in social media, while comparatively few queries might be made about such stories. Hence, one direction for future work would be to model the category of each news story when estimating its importance, with the aim of producing more accurate estimations.

Linking Related News Queries During the development of the news query classification datasets that we use (see Section 5.7), we noted that different news-related queries can refer to the same event. For instance, on the 21st of April 2012, two popular news-related queries were ‘perfect game’ and ‘philip humber’. These two queries share no terms, but refer to the same event, i.e. that the White Sox player Philip Humber tossed a perfect game. Each of these queries alone may be difficult to classify as news-related or not. For example, the terms ‘perfect’ and ‘game’ are not highly discriminative terms and are used in many contexts, hence these terms may not result in useful classification features. However, linking such queries together may lead to more effective news query classification, since these queries are more informative when combined. Therefore, a promising direction for future work would be to develop approaches to link related news queries together such that they can be classified more accurately. Query recommendation approaches may be effective in this setting. For instance, Beeferman & Berger (2000) and Baeza-Yates *et al.* (2005) proposed a query log clustering approaches to find related queries to an initial one.

Predictable Queries When producing relevance assessments for our news-related queries (see Section 5.7), we noted that a small proportion of the queries were related to reoccurring predictable events. For example, a popular query on the 17th of April 2012, was the query 'nfl schedule 2012', relating to the U.S. National Football League match schedule that is published yearly. A promising direction of future work would be to identify reoccurring queries relating to predictable events known to be news-related a-priori. Indeed, related work by Sanderson & Dumais (2007) examined hourly and weekly periodicities in querying behaviour, while Shokouhi (2011) examined how seasonal queries can be identified by Web search engines. By building upon these works, queries may be classified automatically on appropriate days based on past periodicities in usage, thereby potentially increasing classification accuracy.

Blog Post Diversification One of the interesting aspects of the blogosphere in a news search setting is that blog posts may bring more than simply relevant content for news-related queries. In particular, blogs represent a personal log containing the opinions of the author(s). Different bloggers can have greatly diverging opinions about the same story. For example, for a U.S. political news story, there may be blog posts from both republican and democrat viewpoints. Hence, in a universal Web search engine context, it might be advantageous to return blogs or blog posts covering different aspects or viewpoints to maximise the chance that a post which will satisfy the user will be returned in the top ranks (Santos *et al.*, 2012). When ranking blog posts, this could be operationalised by applying diversification approaches, with the aim of maximising the number of event aspects and perspectives covered by the top blog posts retrieved. Future work in this area might involve examining existing diversification approaches, e.g. the Explicit Query Aspect Diversification (xQuAD) framework (Santos, Macdonald & Ounis, 2010*a*), in a news setting and/or the development of new approaches tailored for specific categories of news-related queries. For instance, Yano *et al.* (2009) investigated how to identify political blogs, which we could subsequently diversify.

10.3 Closing Remarks

In this thesis, we argued that user-generated content could aid in satisfying news-related queries submitted to universal Web search engines in real-time. Through a thorough investigation of the search process of a universal Web search engine that supports a news vertical, we showed that user-generated content enables news queries relating to breaking and long-running events to be more accurately classified and the resultant rankings to be enhanced with additional content that end-users find useful. The future of real-time news search appears to be promising. As ever more and richer user-generated content is made available Web search engines can gain more insights into the real-world and how it changes over time. Meanwhile, as wireless internet coverage and the prevalence of internet-enabled devices increases, news will be able to be reported via social media by anyone, anytime, anywhere, enabling search engines to respond faster to emerging events. Furthermore, as search using portable devices that support geographical positioning such as smart phones become ubiquitous, Web search engines will be able to increasingly tailor news results to both the user's current location and context, thereby more effectively serving users with news relevant to them.

Appendix A

Parameter Analysis for Top Events Identification

In Chapter 6 of this thesis, we proposed a variety of unlearned ranking models for identifying top news events, represented by newswire articles. These approaches contain a series of parameters that may effect news story ranking effectiveness, namely a the news story representation, the document ranking depth, size of the recent and background time windows r and w. This appendix details our analysis of these parameters. In particular, Section A.1 details our analysis of the news story representation and document ranking depth on the $BlogTrack_{2009}^{TopNews}$ and $BlogTrack_{2010}^{TopNews-Phase1}$ datasets. Meanwhile, Section A.2 describes our analysis of the document ranking depth, size of the recent and background time windows r and w on the $Twitter_{Dec2011}^{TopNews-NYT}$, $Twitter_{Jan2012}^{TopNews-NYT}$ and $Twitter_{Jan2012}^{TopNews-Reuters}$ datasets.

A.1 Parameter Analysis on Blogs

A.1.1 News Story Representations

Both Votes and RWA use a textual representation of a news story a to retrieve recent and related blog posts or blog feeds for a news story. This story representation is derived from a related news article, e.g. that article's headline. In effect, the story representation acts as the bridge between the story and the blog posts relating to it. However, there are many different possible representations. For example, from a news article, either that article's headline or content could be used. Furthermore, from these base representations, other techniques can then be applied, e.g. query expansion on the headline. In this sub-section, we investigate how using eight different news story representations (see Section 6.4.1) effects the story ranking effectiveness of Votes and RWA.

Approach	Representation	Query Expansion	$BlogTrack_{2009}^{TopNews}$ MAP	$BlogTrack_{2010}^{TopNews-Phase1}$ statMAP
Votes	Headline	None	0.1525	**0.1407**
Votes	Headline	Blogs06	**0.1537**	0.1250
Votes	Headline	NYT06	**0.1537**	0.1146
Votes	Headline	TRC2	0.1501	0.1221
Votes	Content	None	0.0756	0.1068
Votes	Content Named Entities	None	0.0756	0.0939
Votes	Content Noun Phrases	None	0.0756	**0.1175**
Votes	Content Summary	None	**0.1028**	0.0449
RWA	Headline	None	**0.1836**	**0.1846**
RWA	Headline	Blogs06	0.1798	0.1603
RWA	Headline	NYT06	0.1720	0.1482
RWA	Headline	TRC2	0.1665	0.1669
RWA	Content	None	0.0919	**0.1689**
RWA	Content Named Entities	None	0.0919	0.1023
RWA	Content Noun Phrases	None	0.0918	0.1333
RWA	Content Summary	None	**0.1057**	0.0497

Table A.1: News story ranking performance of Votes and RWA when using different story representations a.

Table A.1 reports the story ranking effectiveness of Votes and RWA over the $BlogTrack_{2009}^{TopNews}$ and $BlogTrack_{2010}^{TopNews-Phase1}$ datasets when using eight different story representations, four derived from the news article headline and four from the content of the news article. The best performing Headline/Content-based representation for each story ranking approach and dataset is highlighted. From Table A.1, by comparing the performance difference between story representations, we observe the following. Firstly, the headline-based representations always lead to more effective story rankings than the content-based representations. This is likely due to noise being added by the textually large article content representations. Indeed, long queries are known to be difficult to satisfy due to the existence of extraneous terms (Kumaran & Carvalho, 2009). Secondly, except in the case of Votes on $BlogTrack_{2009}^{TopNews}$, headline enrichment either provides no benefit to story ranking effectiveness or results in a reduction in effectiveness. This result contrasts with that an observation that we made in a previous work (McCreadie *et al.*, 2010*c*), where collection enrichment was shown to lead to small increases in story ranking effectiveness. However, we note that not only is our baseline story ranking approach is stronger than the one used previously (due to the inclusion of heuristic story pruning), but we are working in a real-time rather than a retrospective setting. Hence, either collection enrichment and heuristic story pruning perform the same task, or collection enrichment is less useful in a real-time scenario. Overall, we conclude that for our real-time blog stream setting, using story representations other than the headline do not lead to increased news story ranking effectiveness.

A.1.2 Ranking Depth

In the previous sub-section, we investigated the effect that the news story representation used as the query to retrieve either blog posts or blog feeds has on story ranking effectiveness. However, the number of those blog posts or blog feeds that are retrieved (ranking depth or $|R()|$) can also impact the news story ranking. In particular, each blog post or feed that is returned for a story acts as a vote for that story at (or more precisely around) the time that post/feed was published. However, if any retrieved blog posts/feeds are not in fact relevant to the story, then noise will be added and hence story ranking accuracy will worsen. The more blog posts/feeds that are retrieved, the higher the likelihood that irrelevant posts will be retrieved, but retrieving too few posts/feeds would mean that useful voting evidence will be missed. In the prior experiments, we used the default ranking depth of 1000, as suggested in (Macdonald, 2009). In this section, we investigate how retrieving different numbers of blog posts for each news story effects ranking performance.

Table A.2 reports the ranking effectiveness of Votes and RWA over the $BlogTrack_{2009}^{TopNews}$ and $BlogTrack_{2010}^{TopNews-Phase1}$ datasets when retrieving between 1000 and 20 blog posts for each news story. The results from Table A.2 show the following. Firstly, we see that story ranking effectiveness is improved in the majority of cases when 100 or fewer blog posts are selected, i.e. fewer than the default value of 1000. Indeed, on the $BlogTrack_{2009}^{TopNews}$ dataset, stories were most effectively ranked when only 50 posts were returned, while on $BlogTrack_{2010}^{TopNews-Phase1}$ the best setting tested was 100 blog posts. Secondly, we see that for the content story representation more than for the headline representation, few posts returned resulted in more effective rankings. This indicates that fewer relevant blog posts are retrieved when using content representations in the top ranks, hence more noise is added by increasing $|R()|$. Thirdly, we see that unexpectedly, the most effective story ranking run on $BlogTrack_{2010}^{TopNews-Phase1}$ uses a content representation, rather than a headline representation. This indicates that while fewer relevant results are returned, the ones that are indeed ranked highly, do provide useful evidence. We conclude that retrieving less than the 1000 documents suggested in (Macdonald, 2009) is effective in this setting.

A.2 Parameter Analysis on Twitter

In this section, we examine the effect that the parameters of Votes, RWA and RWAN have on news story ranking effectiveness. In particlar, we evaluate the effect that the background window size w has on story ranking effectiveness in Section A.2.1, while Section A.2.2 similarly examines the effect of the

Approach	Representation	Ranking Depth	$BlogTrack_{2009}^{TopNews}$ MAP	$BlogTrack_{2010}^{TopNews-Phase1}$ statMAP
TREC Best System	Headline	N/A	0.1862	0.1898
Votes	Headline	1000	0.1525	0.1407
Votes	Headline	500	0.1656	**0.1480**
Votes	Headline	300	0.1705	0.1453
Votes	Headline	200	0.1693	0.1458
Votes	Headline	100	0.1721	0.1475
Votes	Headline	50	**0.1812**	0.1465
Votes	Headline	20	0.1661	0.1452
Votes	Content	1000	0.0756	0.1068
Votes	Content	500	0.0789	0.1196
Votes	Content	300	0.0822	0.1246
Votes	Content	200	0.0833	0.1285
Votes	Content	100	0.0857	0.1398
Votes	Content	50	**0.0899**	0.1404
Votes	Content	20	0.0829	**0.1494**
RWA	Headline	1000	0.1836	0.1846
RWA	Headline	500	0.2049	0.1859
RWA	Headline	300	0.2049	**0.1908**
RWA	Headline	200	0.2072	0.1905
RWA	Headline	100	0.2100	0.1866
RWA	Headline	50	**0.2156**	0.1785
RWA	Headline	20	0.2023	0.1728
RWA	Content	1000	0.0919	0.1689
RWA	Content	500	0.0990	0.1796
RWA	Content	300	0.1000	0.1913
RWA	Content	200	0.1061	0.1956
RWA	Content	100	0.1095	**0.2071**
RWA	Content	50	**0.1124**	0.2053
RWA	Content	20	0.1056	0.2021

Table A.2: News story ranking performance of Votes and RWA when retrieval depth is varied.

size of the recent window r. Then, in Sections A.2.3 and A.2.4 we examine the effect that changing the story representation a and the ranking depth $|R()|$ have on story ranking effectiveness, respectively.

A.2.1 Background Window

Our news story ranking approaches are dependant on an underlying ranking of tweets for a news story $R(a, S_{t-w \to t})$. In a real-time stream setting, only a fixed background window of tweets can ever be available due to finite compute resources. The parameter w in $R(a, S_{t-w \to t})$ defines the size of this background window, for instance w could be of size 'one day'. For our initial evaluation of approaches described above, we used a default background window size of 10 days, i.e. $R(a, S_{t-w \to t})$ retrieved related tweets to the news story a for the period of 10 days preceding the ranking time t. Naturally, to deploy such an approach, a computer (or cluster of computers) with enough resources to store and efficiently access 10 days worth of tweets is required. For reference, we estimate that 10 days worth of GZIP[1] compressed JSON format tweets from the full fire-hose stream (not available to researchers) would require approximately 1 terabyte of disk storage space. Hence, we consider 10 days to be a reasonable upper limit for the background window size w. Furthermore, there is a disadvantage to increasing the background window size further, in that the more background tweets that we make available for retrieval, the higher the likelihood that we will start to retrieve tweets for similar but different news stories to the one that we are actually ranking for a.

Table A.3 reports the news story ranking effectiveness of Votes, RWA and RWAN over the three news story ranking datasets for the background window size values of [10,8,6,4,2] days. As an aside, it is notable that we omit the background window size of 1 day from these experiments. This is because the recent window size r is also set to the default of 1 day. As a result, the number of tweets retrieved and the number that receive votes will always be the same (unless fewer than 1000 tweets are retrieved), making Votes meaningless for this one setting. We examine the effect that varying r has in the next sub-section. From Table A.3, we observe the following. Firstly, for Votes, the background window size w used does not have a statistically significant impact on story ranking effectiveness until four or less days worth of evidence is used, and only on the New York Times datasets. Secondly, for RWA and RWAN, the picture is less clear. For $Tweets_{Dec2011}^{TopNews-NYT}$, we see similar significant decreases in story ranking effectiveness under NDCG for window sizes less than 6. However, in contrast, for a the window size of 2 days, we see small but statistically significant increases in effectiveness on the $Tweets_{Dec2012}$ datasets. If we examine effectiveness across approaches, we see that in two out of the three cases, the most effective news story ranking setting used the smallest window size, while in the

[1]www.gzip.org/

Approach	Background Time Window Size w	Effectiveness					
		$Tweets_{Dec2011}^{TopNews-NYT}$		$Tweets_{Jan2012}^{TopNews-NYT}$		$Tweets_{Jan2012}^{TopNews-Reuters}$	
		NDCG	Success@1	NDCG	Success@1	NDCG	Success@1
Votes	10	0.7443	0.0666	0.7313	0.0178	0.7650	**0.2539**
Votes	8	**0.7509**	0.1111	0.7315	0.0178	0.7648	0.2380
Votes	6	0.7394	0.1555	0.7324	0.0178	0.7648	0.2380
Votes	4	0.7052▼ ▼	0.0666	0.7246▼	0.0714	0.7612	0.2063
Votes	2	0.7050▼ ▼	**0.2000**	**0.7408**	**0.1428**	**0.7678**	0.2222
RWA	10	0.7665	**0.1333**	0.7821	0.2678	**0.7324**	0.1587
RWA	8	0.7674	0.0444	0.7818	0.2678	0.7322	**0.1746**
RWA	6	**0.7712**	0.0888	0.7814	0.2678	0.7318	0.1428
RWA	4	0.7480▼ ▼	0.0666	0.7811	0.2678	0.7297	0.1111
RWA	2	0.7648	0.1111	**0.7867▲**	**0.2857**	0.7280	**0.1746**
RWAN	10	0.7022	**0.0444**	0.7328	**0.0357**	0.7765	0.1587
RWAN	8	**0.7035**	**0.0444**	0.7326	**0.0357**	0.7763	**0.1746**
RWAN	6	0.6867▼ ▼	**0.0444**	0.7346	**0.0357**	0.7753	**0.1746**
RWAN	4	0.6735▼ ▼	**0.0444**	0.7368	**0.0357**	0.7821▲	0.1587
RWAN	2	0.6650▼ ▼	0.0000	**0.7407**	**0.0357**	**0.7853▲ ▲**	0.1428
Topic Count		45	45	71	56	63	63

Table A.3: News story ranking performance of Votes when the background window size of available tweets is varied. The most effective NDCG and Success@1 performance observed under each approach and dataset is highlighted. ▲ and ▼ indicate statistically significant increases/decreases over the same approach with a default window size of 10 days under *t-test* $p < 0.05$. ▲▲ and ▼▼ indicate statistically significant increases/decreases under *t-test* $p < 0.01$.

other case ($Tweets_{Dec2011}^{TopNews-NYT}$) a window size of 6 days lead to the best ranking. From this we can conclude that our approaches do not need a full 10 days of background tweets to be effective. However, the small nature of the variance in news story ranking effectiveness observed here lead us to conclude that in general the volume of background tweets used has little overall effect on news story ranking effectiveness for the settings tested. However, if this generalises to other settings, then it is a valuable property for our approaches to have, as it means that less background tweets need be stored, in turn reducing the compute requirements deploy such an approach.

A.2.2 Recent Window

Each of our news story ranking approaches are based on the idea that if a news story is important, then there will be comparatively more discussion about that story in user-generated content around the time of ranking t than normal. This is operationalised by measuring the number of related posts retrieved for the story from a background window size w that are also present in a smaller recent window of size r and positioned just before t. In the previous sub-section we showed that by decreasing the size of the background window w there was little change in story ranking effectiveness (the r value was set to the default of 24 hours). In this sub-section, we examine the effect that the size of the recent time window has on story ranking effectiveness. Intuitively, these two parameters are to some extent dependant upon one another. For example, if w was set to two days and r was set to one day, then our approaches are in fact comparing the volume and relevance today's posts with yesterdays posts for a news story. Contrast this to a w size of 7, where we are instead comparing against the prior week worth of posts. To account for this we, examine how news story ranking effectiveness changes as we vary the size of the smaller window r for two settings of w, i.e. a large window of 10 days and a small window of 2 days.

Table A.4 reports news story ranking effectiveness of Votes, RWA and RWAN for the three datasets, r values ranging from one hour to 48 hours preceding t, where the background window w is set to 10 days. Table A.5 similarly reports story ranking effectiveness for a smaller w value of 2 days. From these two tables, we make the following observations. Firstly, when using the larger background time window w of 10 days, we see that on the $Tweets_{Dec2011}$ dataset, by decreasing the recent time window, i.e. only counting discussion closer to the query time t, leads to statistically significant decreases in effectiveness. Meanwhile, on the $Tweets_{Jan2012}$, under most settings, effectiveness either changes little or is similarly decreased. The only notable exception to this trend is when a recent window size of 24 hours is used on the $Tweets_{Jan2012}^{TopNews-Reuters}$ dataset. Here we observe a small but statistically significant increase in story ranking performance, leading to the highest observed performance. However, overall, when the a background window of 10 days is used, it appears that considering around one day worth of tweets is effective. Next, if we shrink the background window size to only 2 days, as shown in Table A.5, we observe a similar trend. On the $Tweets_{Dec2011}$ dataset, we see statistically significant decreases in effectiveness for r values lower than 24 hours. On the $Tweets_{Jan2012}$ datasets, effectiveness remains roughly constant as r is varied, with the same peak in performance when a recent window size of 2 days is used on the $Tweets_{Jan2012}^{TopNews-Reuters}$ dataset. Overall, we conclude that for the majority of settings tested here, using smaller window sizes than 24 hours decreases story ranking effectiveness. The reason for this is that even though using a small window size might enable better scoring of news stories as they break, the ranking effectiveness of longer running but still interesting stories will suffer, as related discussion in Twitter dies down.

A.2.3 News Story Representations

Next, we examine how different news story representations effect news story ranking with tweets. Notably, due to the manner in which the news stories to be ranked are collected from each news provider, only news story headlines are available (not all news stories have snippets describing them). Furthermore, unlike our blog stream experiments, we do not have other news story sources to use for query expansion. Instead, to generate alternate news story representations, we use a Wikipedia dump from

Approach	Recent Time Window Size w	Effectiveness					
		$Tweets^{TopNews-NYT}_{Dec2011}$		$Tweets^{TopNews-NYT}_{Jan2012}$		$Tweets^{TopNews-Reuters}_{Jan2012}$	
		NDCG	Success@1	NDCG	Success@1	NDCG	Success@1
Votes	48	0.7116▼ ▼	0.0444	**0.7388**	0.0892	0.7565▼	**0.2222**
Votes	24	**0.7442**	0.0666	0.7313	0.0178	**0.7650**	0.2063
Votes	12	0.7398	**0.2444**	0.7376	**0.1607**	0.7561▼	0.1587
Votes	6	0.7220▼ ▼	0.200.0000	0.7338	0.0892	0.7497▼	0.0793
Votes	3	0.7078▼ ▼	0.1333	0.7329	0.1250	0.7536	0.0634
Votes	1	0.6877▼ ▼	0.1333	0.7217	0.0714	0.7507	0.0476
RWA	48	0.7563▼	0.0444	0.7769	0.1607	0.7161▼ ▼	0.1269
RWA	24	**0.7666**	0.1333	**0.7820**	0.2678	0.7324	**0.1428**
RWA	12	0.7637	**0.2444**	0.7775	**0.2857**	0.7377	**0.1428**
RWA	6	0.7454▼	**0.2444**	0.7700▼	0.2142	0.7423▲	0.0793
RWA	3	0.7391▼ ▼	0.2222	0.7668▼	0.2321	**0.7454▲**	0.0634
RWA	1	0.7637	**0.2444**	0.7775	**0.2857**	0.7377	**0.1428**
RWAN	48	0.7005	0.0444	0.7339	0.0357	**0.7837▲**	0.1587
RWAN	24	0.7022	0.0444	0.7328	0.0357	0.7765	**0.2222**
RWAN	12	**0.7102**	0.0444	**0.7437**	0.0357	0.7722	0.1269
RWAN	6	0.6944	0.0666	0.7406	0.0357	0.7649▼	0.0793
RWAN	3	0.6829▼	**0.0888**	0.7286	**0.0535**	0.7662	0.0634
RWAN	1	0.6740▼ ▼	0.0666	0.7184	0.0350	0.7661	0.0793
Topic Count		45	45	72	57	63	63

Table A.4: News story ranking performance of Votes when the recent time window size r is varied. The background window size w is set to 10. The most effective NDCG and Success@1 performance observed under each approach and dataset is highlighted. ▲ and ▼ indicate statistically significant increases/decreases over the same approach with a default window size of 10 days under *t-test* $p < 0.05$. ▲▲ and ▼▼ indicate statistically significant increases/decreases under *t-test* $p < 0.01$.

Approach	Recent Time Window Size w	Effectiveness					
		$Tweets^{TopNews-NYT}_{Dec2011}$		$Tweets^{TopNews-NYT}_{Jan2012}$		$Tweets^{TopNews-Reuters}_{Jan2012}$	
		NDCG	Success@1	NDCG	Success@1	NDCG	Success@1
Votes	24 (w=10 days)	**0.7442**	0.0666	0.7313	0.0178	**0.7650**	0.2063
Votes	24	**0.7047▼ ▼**	0.2000	**0.7388**	0.0892	**0.7679**	**0.2063**
Votes	12	0.7010▼ ▼	**0.2444**	0.7313	0.0178	0.7533	0.1904
Votes	6	0.6868▼ ▼	0.1555	0.7376	**0.1607**	0.7511	0.0952
Votes	3	0.6817▼ ▼	0.1111	0.7338	0.0892	0.7498▼	0.0476
Votes	1	0.6824▼ ▼	0.1111	0.7329	0.1250	0.7487	0.0317
RWA	24 (w=10 days)	**0.7666**	0.1333	**0.7820**	0.2678	0.7324	**0.1428**
RWA	24	**0.7649**	0.1111	0.7217▼ ▼	0.0714	0.7279	0.1428
RWA	12	0.7518	**0.2444**	0.7769	0.1607	0.7380	**0.1746**
RWA	6	0.7293▼ ▼	0.2222	**0.7820**	0.2678	0.7421	0.1269
RWA	3	0.7215▼ ▼	0.2222	0.7775	**0.2857**	0.7451▲	0.0952
RWA	1	0.7166▼ ▼	0.1777	0.7700*	0.2142	**0.7474**	0.0793
RWAN	24 (w=10 days)	0.7022	0.0444	0.7328	0.0357	0.7765	**0.2222**
RWAN	24	0.6650▼ ▼	0.0000	**0.7668▲ ▲**	**0.2321**	**0.7853▲ ▲**	**0.1428**
RWAN	12	**0.6769▼**	0.0000	0.7565	0.1964	0.7800	0.0634
RWAN	6	0.6566▼ ▼	0.0444	0.7339	0.0357	0.7716	0.0476
RWAN	3	0.6600▼ ▼	**0.0666**	0.7328	0.0357	0.7684	0.0476
RWAN	1	0.6629▼ ▼	0.0444	0.7437	0.0357	0.7664	0.0634
Topic Count		45	45	71	56	63	63

Table A.5: News story ranking performance of Votes when the recent time window size is varied. The background window size w is set to 2 days. The most effective NDCG and Success@1 performance observed under each approach and dataset is highlighted. ▲ and ▼ indicate statistically significant increases/decreases over the same approach with a default window size of 10 days under *t-test* $p < 0.05$. ▲▲ and ▼▼ indicate statistically significant increases/decreases under *t-test* $p < 0.01$.

Approach	Collection Enrichment Enrichment of a	Effectiveness					
		$Tweets^{TopNews-NYT}_{Dec2011}$		$Tweets^{TopNews-NYT}_{Jan2012}$		$Tweets^{TopNews-Reuters}_{Jan2012}$	
		NDCG	Success@1	NDCG	Success@1	NDCG	Success@1
Votes	None	**0.7443**	**0.0888**	0.7313	0.0178	0.7650	0.2380
Votes	Wikipedia	0.7394	0.0666	**0.7376**	**0.2321**	**0.7711**	**0.2698**
RWA	None	0.7666	**0.1333**	**0.7820**	**0.2678**	**0.7324**	**0.1111**
RWA	Wikipedia	**0.7684**	0.0888	0.7666	0.1607	0.7307	**0.1111**
RWAN	None	**0.7022**	**0.0444**	0.7328	**0.0357**	**0.7765**	**0.1904**
RWAN	Wikipedia	**0.7022**	**0.0444**	**0.7329**	**0.0357**	**0.7765**	**0.1904**
Topic Count		45	45	71	56	63	63

Table A.6: News story ranking performance of Votes, RWA and RWAN with and without query expansion on Wikipedia. The most effective NDCG and Success@1 performance observed under each approach and dataset is highlighted. ▲ and ▼ indicate statistically significant increases/decreases over the same approach with a default window size of 10 days under *t-test* $p < 0.05$. ▲▲ and ▼▼ indicate statistically significant increases/decreases under *t-test* $p < 0.01$.

November 2011 to expand each news story headline with a fixed number of terms. In this case, we compare the un-expanded headline with the headline expanded using 10 terms from the top 3 tweets returned.

Table A.6 reports the news story ranking effectiveness of each of Votes, RWA and RWAN on the three tweet datasets, when expanding the headline of each news story with pseudo-relevance feedback. From Table A.6, we see that in terms of NDCG, applying collection enrichment from Wikipedia has little effect upon the news story ranking, while under Success@1, we see some variance for the $Tweets^{TopNews-NYT}_{Jan2012}$ dataset. This negative result is interesting, in that it exposes an aspect our approach when deployed in a Twitter setting. In particular, the most relevant tweets to a news story are those that exactly match the story itself. e.g. if we are ranking for the news story 'Romney and Obama Are in a Tight Contest, Poll Finds', then there are often relevant tweets of the form 'Romney and Obama Are in a Tight Contest, Poll Finds <SOME URL>' where <SOME URL> points to the source news article. PRF selects the top three tweets for each of these news stories, which are often almost identical to the news story title, and hence don't find any additional terms with which to expand. Indeed, it is for this reason that query expansion performance is largely static when searching for news stories on Twitter.

A.2.4 Ranking Depth

In this sub-section, we investigate the effect of the number of tweets retrieved $|R()|$ (that subsequently act as votes for the importance of each news story) has on the effectiveness of Votes, RWA and RWAN. On the blog streams, we previously observed increases in news story ranking effectiveness by decreasing the number of retrieved blog posts from 1000 to roughly 100. We experimentally test to see if this conclusion holds for tweet streams as well. We test our news story ranking approaches as they use the top 10,30 50, 100, 300, 500 and 1000 tweets as voting evidence.

Table A.7 reports the news story ranking effectiveness of Votes, RWA and RWAN on the three datasets when retrieving different numbers of tweets for each news story. Interestingly, we observe two distinct patterns, one for Votes and one for RWA and RWAN. In particular, Votes maintains or improves effectiveness as fewer of the top tweets are used as evidence, with the exception of 10 tweets on the $Tweets_{Dec2011}^{TopNews-NYT}$ dataset. Meanwhile, for our approaches that take into account the retrieval score (relevance) of the tweet, retrieving the full 1000 tweets leads to better performance. These patterns can be explained in terms of the relevance of the tweets returned in the top results for each story. Recall that irrelevant tweets act as random votes. Our Votes approach considers each tweet equally, and hence will suffer more than RWA and RWAN when irrelevant tweets are retrieved in the top 1000 because RWA and RWAN favour tweets near the top of the ranking, placing less weight on lower ranked tweets that may be irrelevant. Overall, we see the best story ranking effectiveness across all approaches when 1000 tweets are returned. Hence, we conclude that for tweets retrieving 1000 tweets is superior to retrieving only the 100 posts that was shown to be effective on blogs.

Approach	Ranking Depth $\lvert R()\rvert$	Effectiveness					
		$Tweets^{TopNews-NYT}_{Dec2011}$		$Tweets^{TopNews-NYT}_{Jan2012}$		$Tweets^{TopNews-Reuters}_{Jan2012}$	
		NDCG	Success@1	NDCG	Success@1	NDCG	Success@1
Votes	1000	0.7442	0.0666	0.7313	0.0178	0.7650	**0.2222**
Votes	500	0.7221▼▼	0.0444	0.7318	0.1428	0.7640	0.1904
Votes	300	0.7215▼▼	0.1333	0.7232	0.1071	0.7632	0.1111
Votes	100	**0.7460**	**0.2222**	0.7348	**0.1785**	0.7560	0.0793
Votes	50	0.7389	0.1777	0.7487	0.1607	0.7637	0.1269
Votes	30	0.7459	0.2000	**0.7500▲**	0.1607	0.7559	0.0317
Votes	10	0.7069▼▼	0.0888	0.7132	0.1071	**0.7714**	0.0476
RWA	1000	**0.7665**	0.1333	**0.7820**	0.2678	0.7324	0.1269
RWA	500	0.7501▼▼	0.1111	0.7819	**0.3035**	0.7378	**0.1587**
RWA	300	0.7516▼▼	0.1333	0.7796	0.2857	0.7444▲▲	0.1428
RWA	100	0.7630	0.2222	0.7645▼▼	0.0892	**0.7489▲▲**	0.1428
RWA	50	0.7548	0.2222	0.7679▼	0.1428	0.7477▲▲	0.1428
RWA	30	0.7576	**0.2444**	0.7582▼▼	0.1428	0.7376	0.0476
RWA	10	0.7407▼▼	0.2000	0.7453▼▼	0.1964	0.7385	0.0793
RWAN	1000	0.7022	0.0444	**0.7328**	0.0357	**0.7765**	0.1587
RWAN	500	0.7003	0.0444	0.7165▼▼	0.0357	0.7737	**0.1746**
RWAN	300	0.7085	0.0444	0.7218	0.0357	0.7711	0.1587
RWAN	100	0.7279▲▲	0.1111	0.7164▼	0.0714	0.7576▼▼	0.0793
RWAN	50	0.7283▲▲	**0.1555**	0.7226	**0.0892**	0.7566▼▼	0.0793
RWAN	30	**0.7293▲▲**	**0.1555**	0.7165▼	**0.0892**	0.7532▼▼	0.0793
RWAN	10	0.7095	0.0222	0.7120▼	**0.0892**	0.7509▼▼	0.0634
Topic Count		45	45	71	56	63	63

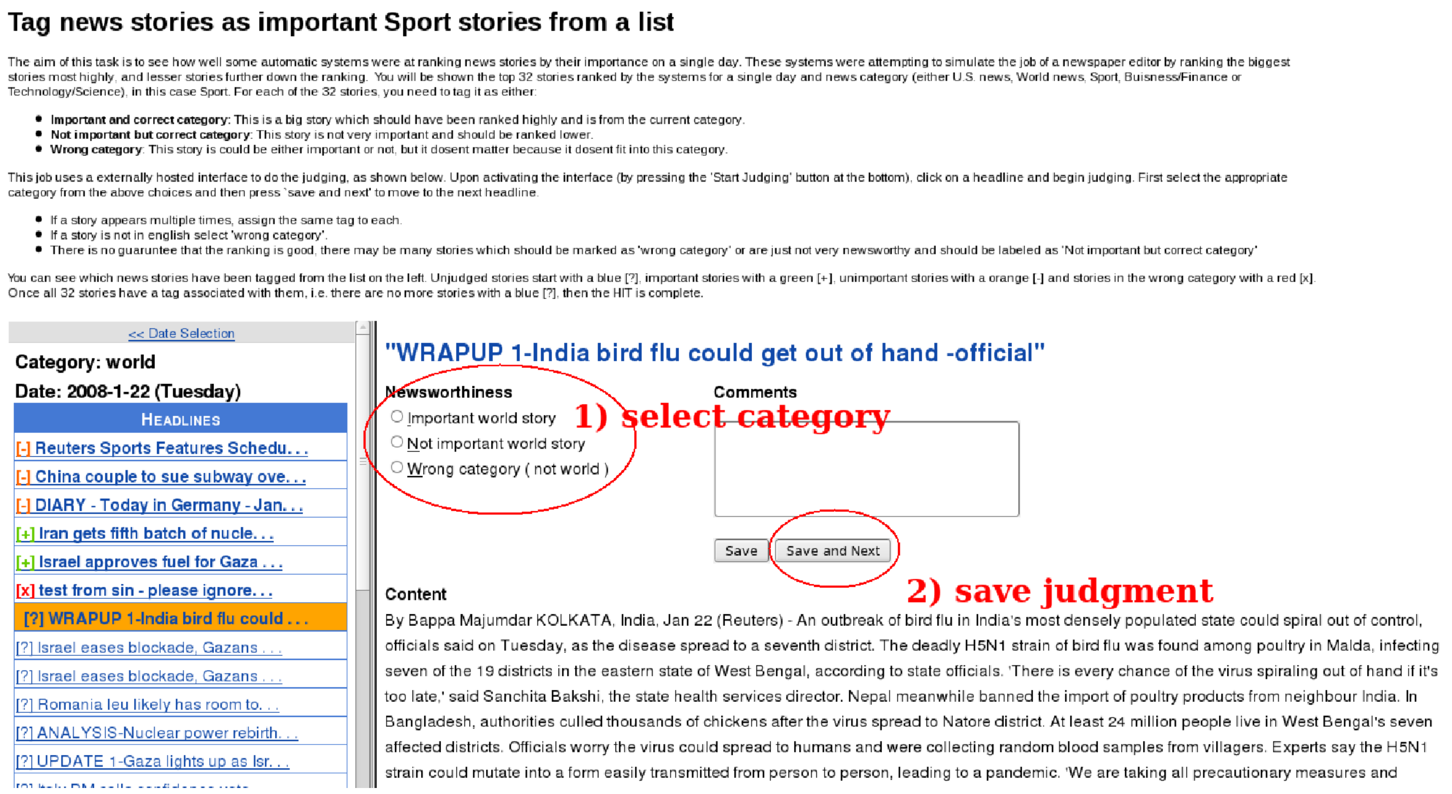
Table A.7: News story ranking performance of Votes, RWA and RWAN as ranking depth is varied. The most effective NDCG and Success@1 performance observed under each approach and dataset is highlighted. ▲ and ▼ indicate statistically significant increases/decreases over the same approach with a default window size of 10 days under *t-test* $p < 0.05$. ▲▲ and ▼▼ indicate statistically significant increases/decreases under *t-test* $p < 0.01$.

Appendix B

Worker Instructions for Crowdsourcing

In this appendix, we provide screenshots of the instructions provided to each worker for the crowd-sourced tasks described in Sections 5.4 to 5.7. In particular, we provide the worker instructions for identifying top events (see Section 5.4) in Section B.1, while the worker instructions for the classification of user queries (see Section 5.5) are shown in Section B.2. The worker instructions for blog post ranking (see Section 5.6) are provided in Section B.3, while the worker instructions for the document relevance assessment task (see Section 5.7) are shown in Section B.4.

B.1 Story Ranking Task Worker Instructions



Tag news stories as important Sport stories from a list

The aim of this task is to see how well some automatic systems were at ranking news stories by their importance on a single day. These systems were attempting to simulate the job of a newspaper editor by ranking the biggest stories most highly, and lesser stories further down the ranking. You will be shown the top 32 stories ranked by the systems for a single day and news category (either U.S. news, World news, Sport, Buisness/Finance or Technology/Science), in this case Sport. For each of the 32 stories, you need to tag it as either:

- **Important and correct category:** This is a big story which should have been ranked highly and is from the current category.
- **Not important but correct category**: This story is not very important and should be ranked lower.
- **Wrong category**: This story is could be either important or not, but it dosent matter because it dosent fit into this category.

This job uses a externally hosted interface to do the judging, as shown below. Upon activating the interface (by pressing the 'Start Judging' button at the bottom), click on a headline and begin judging. First select the appropriate category from the above choices and then press `save and next' to move to the next headline.

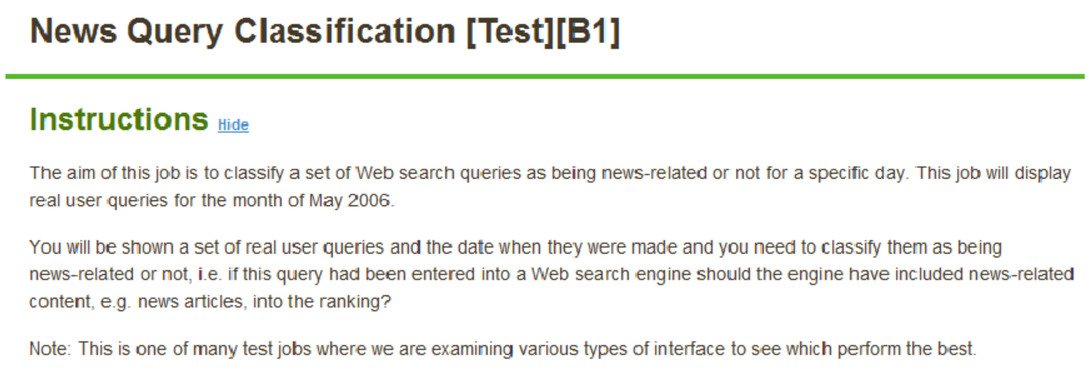
- If a story appears multiple times, assign the same tag to each.
- If a story is not in english select 'wrong category'.
- There is no guaruntee that the ranking is good, there may be many stories which should be marked as 'wrong category' or are just not very newsworthy and should be labeled as 'Not important but correct category'

You can see which news stories have been tagged from the list on the left. Unjudged stories start with a blue [?], important stories with a green [+], unimportant stories with a orange [-] and stories in the wrong category with a red [x]. Once all 32 stories have a tag associated with them, i.e. there are no more stories with a blue [?], then the HIT is complete.

<< Date Selection
Category: world
Date: 2008-1-22 (Tuesday)
Headlines
[-] Reuters Sports Features Schedu. . .
[-] China couple to sue subway ove. . .
[-] DIARY - Today in Germany - Jan. . .
[+] Iran gets fifth batch of nucle. . .
[+] Israel approves fuel for Gaza . . .
[x] test from sin - please ignore. . .
[?] WRAPUP 1-India bird flu could . . .
[?] Israel eases blockade, Gazans . . .
[?] Israel eases blockade, Gazans . . .
[?] Romania leu likely has room to. . .
[?] ANALYSIS-Nuclear power rebirth. . .
[?] UPDATE 1-Gaza lights up as Isr. . .
"WRAPUP 1-India bird flu could get out of hand -official"
Newsworthiness
○ Important world story
○ Not important world story
○ Wrong category (not world)
1) select category
Comments
Save
Save and Next
2) save judgment
Content

By Bappa Majumdar KOLKATA, India, Jan 22 (Reuters) - An outbreak of bird flu in India's most densely populated state could spiral out of control, officials said on Tuesday, as the disease spread to a seventh district. The deadly H5N1 strain of bird flu was found among poultry in Malda, infecting seven of the 19 districts in the eastern state of West Bengal, according to state officials. 'There is every chance of the virus spiraling out of hand if it's too late,' said Sanchita Bakshi, the state health services director. Nepal meanwhile banned the import of poultry products from neighbour India. In Bangladesh, authorities culled thousands of chickens after the virus spread to Natore district. At least 24 million people live in West Bengal's seven affected districts. Officials worry the virus could spread to humans and were collecting random blood samples from villagers. Experts say the H5N1 strain could mutate into a form easily transmitted from person to person, leading to a pandemic. 'We are taking all precautionary measures and

B.2 News Query Classification Worker Instructions



News Query Classification [Test][B1]

Instructions Hide

The aim of this job is to classify a set of Web search queries as being news-related or not for a specific day. This job will display real user queries for the month of May 2006.

You will be shown a set of real user queries and the date when they were made and you need to classify them as being news-related or not, i.e. if this query had been entered into a Web search engine should the engine have included news-related content, e.g. news articles, into the ranking?

Note: This is one of many test jobs where we are examining various types of interface to see which perform the best.

B.3 Blog Post Ranking Task Worker Instructions

1) What to do

The aim of this task is to find some blog posts which could resonably be returned by a blog search engine for a specific news story. You will be shown between 3-20 blog posts for a news story and you need to label each blog post as either:

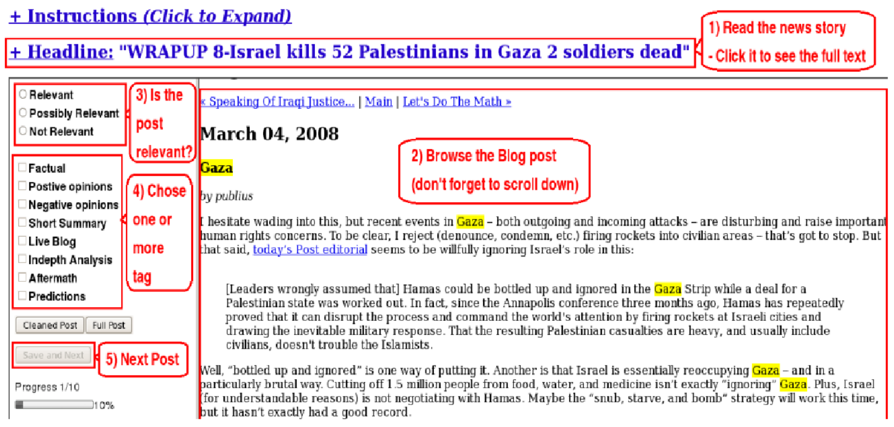
- **Relevant:** Story is discussed. Note that it must be the story in question and not some related story that is discussed.
- **Possibly relevant:** Post could be discussing the story. Chose this option when it looks like the story is being discussed but you can't be sure.
- **Not Relevant:** Story is not discussed. The blog post is either unrelated or discusses another (unrelated) story. This blog post should not be returned by a blog search engine for the story.

We also want you to tag each post. In difference to earlier versions of this task, we provide the tags for you. Pick any number from the following list:

- **Factual Account :** The post just describes the facts as is, e.g. Wikileaks has published over 200,000 diplomatic cables.
- **Opinionated Positive:** The post expresses a viewpoint endorsing some aspect of the story, e.g. Wikileaks is doing the right thing in releasing the cables, the public has a right to know.
- **Opinionated Negitive:** The post critisises some aspect of the story, e.g. Its disgraceful, the diplomatic cables should never have been leaked.
- **Opinionated Mixed:** The post expresses both positive and negitive opinions.
- **Short summary/Quick bites:** The post contains only a sentence or two about the story.
- **Live Blog:** The post continually updated at the time about the story.
- **Indepth analysis:** The post goes into significant detail about the story. Indeed, it is written in a similar way to a high quality news article.
- **Aftermath:** The post gives a round-up or retrospective account of the story.
- **Predictions:** The post was written before the story and discusses what might happen.

2) How to do it

Follow steps 1 to 5 in the picture below. Repeat steps 2 to 5 for each of the 20 blog posts (you can see your progress in the left frame).


+ Instructions *(Click to Expand)*
+ Headline: "WRAPUP 8-Israel kills 52 Palestinians in Gaza 2 soldiers dead"
1) Read the news story - Click it to see the full text
Relevant
Possibly Relevant
Not Relevant
3) Is the post relevant?
Factual
Postive opinions
Negative opinions
Short Summary
Live Blog
Indepth Analysis
Aftermath
Predictions
4) Chose one or more tag
Cleaned Post
Full Post
Save and Next
5) Next Post
Progress 1/10
10%
« Speaking Of Iraqi Justice... | Main | Let's Do The Math »
2) Browse the Blog post (don't forget to scroll down)

March 04, 2008

Gaza

by publius

I hesitate wading into this, but recent events in Gaza – both outgoing and incoming attacks – are disturbing and raise important human rights concerns. To be clear, I reject (denounce, condemn, etc.) firing rockets into civilian areas – that's got to stop. But that said, today's Post editorial seems to be willfully ignoring Israel's role in this:

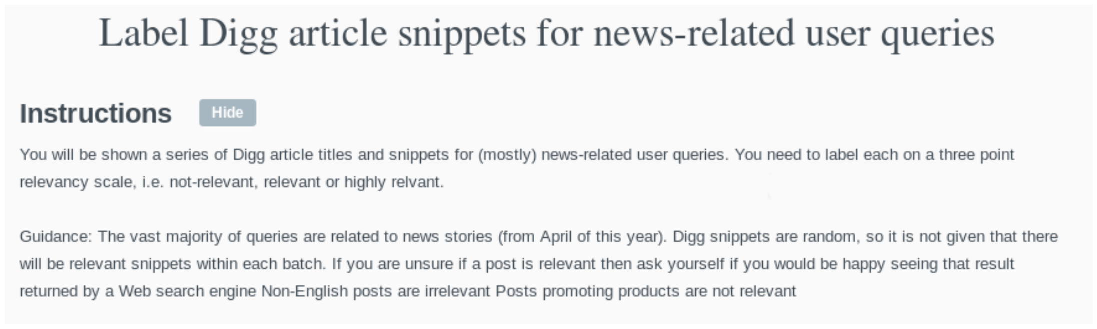
> [Leaders wrongly assumed that] Hamas could be bottled up and ignored in the Gaza Strip while a deal for a Palestinian state was worked out. In fact, since the Annapolis conference three months ago, Hamas has repeatedly proved that it can disrupt the process and command the world's attention by firing rockets at Israeli cities and drawing the inevitable military response. That the resulting Palestinian casualties are heavy, and usually include civilians, doesn't trouble the Islamists.

Well, "bottled up and ignored" is one way of putting it. Another is that Israel is essentially reoccupying Gaza – and in a particularly brutal way. Cutting off 1.5 million people from food, water, and medicine isn't exactly "ignoring" Gaza. Plus, Israel (for understandable reasons) is not negotiating with Hamas. Maybe the "snub, starve, and bomb" strategy will work this time, but it hasn't exactly had a good record.

3) Guidelines (this will be updated based on comments recieved)

- We have tried to clean most pages of javascript, ads, etc to cut down on loading times, if we have broken a page in doing so, press the 'full post' button below the tags to get the unmodified version.
- Any inappropriate pages (porn) should be marked as not relevant.
- If a page is not in english mark it as not relevant.
- A blog post can still be highly relevant even if it was posted before the story, i.e. expectations might still be interesting.
- You should NOT consider any comments in the post when judging.
- If a blog post does not appear, select not relevant and select save and next.
- **In some cases the progress bar may reset to 0 and go to the start after all posts have been judged - do not rejudge the posts, your work has been saved and you can press submit.**
- There are some HITs with less than 20 posts. This is normal.

B.4 Document Assessment Task Worker Instructions



Label Digg article snippets for news-related user queries

Instructions Hide

You will be shown a series of Digg article titles and snippets for (mostly) news-related user queries. You need to label each on a three point relevancy scale, i.e. not-relevant, relevant or highly relvant.

Guidance: The vast majority of queries are related to news stories (from April of this year). Digg snippets are random, so it is not given that there will be relevant snippets within each batch. If you are unsure if a post is relevant then ask yourself if you would be happy seeing that result returned by a Web search engine Non-English posts are irrelevant Posts promoting products are not relevant

Appendix C

Features for News Query Classification

In this appendix, we provide a detailed description of all of the features that we use in our news query classification experiments in Chapter 7. In particular, Section C.1 describes our query-only features, while Section C.2 details our frequency features. In Section C.3, we describe our retrieval features, while in Section C.4 we describe our burstiness features. Section C.5 details our importance features, while Section C.6 describes our stream-specific features.

C.1 Query-only Features

Initially, it is important to extract evidence from the query text itself, as this is the expression of the user's information need. Hence, we begin by extracting the basic textual features of the query. These features can roughly be classified into four categories, namely: length metrics; named entities; query composition; and query time. Firstly, *length metrics* are features which measure how long a query is. The average Web search query is between 2-3 terms in length (Bar-Ilan *et al.*, 2009), however, news queries may be longer than this if the user is searching for a specific story, e.g. 'labour leadership election winner'. Similar query-only features were also used by König *et al.* (2009) to tackle the related task of click-through prediction for news-related queries.

The second category of features we examine refers to *named entities*. It is intuitive that news stories are about or refer heavily to entities, e.g. people or organisations. Hence, the existence of such named entities within the query may be indicative of that query's news-relatedness. For the $NQC_{May2006}$ dataset, we use a dictionary of 1.4 million entities extracted from a 2008 Wikipedia dump to identify those entities within a query (Santos, Macdonald & Ounis, 2010*b*), using the presence of these as a feature. For completeness, it should be noted that as this entity dictionary comes from a Wikipedia dump made after the time of our queries, the entity features might encapsulate evidence that was not available at the time of the query. However, we do not see this as an issue since it is only used to create a simple static listing of all known named entities. However, this approach is potentially problematic if applied on the $NQC_{Apr2012}$ dataset. In particular, if were were to re-use this same Wikipedia dump for entity extraction, then important entities that have emerged during the preceding 4 years from the time that dump was made may be missing. Hence, we use a different method to extract named entities for $NQC_{Apr2012}$. In particular, we use the AlchemyAPI[1] named entity extraction service on each query to identify any named entity types and subtypes present. Notably, rather than expressing these as the number of People/Places/Organisations/Products, we instead express the entity information as a series of binary contains <entity> features, one for each entity type and subtype that AlchemyAPI identifies. For example, containsCity or containsPerson are type features, while containsUSState is an entity subtype feature.

Next, we look at the general *query composition*, i.e. its structure. For example, we check whether the query contains a universal resource locator (URL). As noted in our discussion on query types in Section 3.5.2, queries that contain a URL are often navigational in nature. Intuitively, this should act as a negative feature, as navigational queries are unlikely to be news-related (Broder, 2002). We also check whether the query contains the term 'news' or one of its variants, such that we can better identify news-related queries with current-news information needs.

Finally, we consider the *query time*. It is intuitive that the time a query is submitted will act as a prior on whether a query is news-related or not. For example, it is less likely that users will be searching for breaking news during the early hours of the morning. Hence, we add four binary time-oriented features, *isMorning*, *isEvening*, *isNightTime* and *isWeekend*, identifying the period during which the query was made.

On $NQC_{May2006}$ we extract a total of 9 length metric, named entity, and query composition features. On $NQC_{Apr2012}$ we extract 8 length metric, query composition and query time features, in addition to 173 contains named entity features.

C.2 Frequency Features

The second set of features that we examine are raw query-frequency features. These features represent our basic knowledge about the current popularity of the query in each stream. When looking at a constrained setting of a single hour or day, a high term or document frequency of one or more query-terms may indicate that users are reporting and/or discussing an interesting event relating to the query.

[1]`http://www.alchemyapi.com/`

Hence, the query is more likely to be news-related. The features in this feature set are inspired by earlier work by (Jones & Diaz, 2007) who used language modelling equivalents of the features described here to analyse the temporal profile of queries (see Section 3.5.1). König *et al.* (2009) also reported that such features were effective when tackling the similar task of click-through prediction for news-related queries.

Notably, frequency features encapsulate the usage of one or more query terms for a *time-window* over a stream. We use a standard set of notation to describe this. In particular, we denote the stream that the feature is extracted from as S. The time of the query is denoted t. The feature is calculated over a window of size w. Hence, the time-window of the stream is denoted $S_{t-w \to t}$. Furthermore, in some cases, we require a larger window describing the query term frequency is over a longer background period. We denote the size of the background window b and the window itself as $S_{t-b \to t}$. We report w in terms of hours and b in terms of days.

We examine four features extracted using the sum of the terms from the query to be classified and each of the seven streams:

- **Stream term frequency (*TF*$_{stream}$)**: The number of times each of the query terms appear within documents from the stream window $S_{t-w \to t}$, normalised by the expected background frequency of those terms for periods of length w.

- **Stream document frequency (*DF*$_{stream}$)**: The number of documents within which each of the query terms appear from the stream window $S_{t-w \to t}$, normalised by the expected background frequency for documents containing those terms for periods of length w.

- **Stream term frequency inverse document frequency (*TF-IDF*$_{stream}$)**: The number of times each of the query terms appear within documents from the stream window $S_{t-w \to t}$, normalised by the expected background frequency of those terms for periods of length w and multiplied by the background document frequency for the period of length b.

- **Stream document frequency inverse document frequency (*DF-IDF*$_{stream}$)**: The number of documents within which each of the query terms appear from the stream window $S_{t-w \to t}$, normalised by the expected background frequency of documents containing those terms for periods of length w and multiplied by the background document frequency for the period of length b.

Note that these are common IR document statistics re-purposed as features for a streaming environment, i.e. due to our real-time streaming setting, these features are calculated with respect to a time window (subset of documents) defined by $S_{t-w \to t}$. Furthermore, by varying the size of the window w, different features are generated. For these experiments, we use two values for w, i.e. one hour preceding each query and the 24 hours preceding the query. Finally, we normalise basic term and document frequencies by the expected background usage for time periods of that length. In this way, these features measure how much higher or lower is the discussion about the query within the stream than we would expect. The definition of each feature is provided below:

$$TF_{stream}(Q, S_{t-w \to t}) = \frac{\sum_{qt \in Q} \frac{tf(qt, S_{t-w \to t})}{Avg_{tf}(Q,S,t,w,b)}}{|Q|} \tag{C.1}$$

$$DF_{stream}(Q, S_{t-w \to t}) = \frac{\sum_{qt \in Q} \frac{df(qt, S_{t-w \to t})}{Avg_{df}(Q,S,t,w,b)}}{|Q|} \tag{C.2}$$

$$TF\text{-}IDF_{stream}(Q, S_{t-w \to t}) = \frac{\sum_{qt \in Q} \frac{tf(qt, S_{t-w \to t})}{Avg_{tf}(Q,S,t,w,b)} \cdot log(1 + \frac{|S_{t-b \to t}|}{df(qt, S_{t-b \to t})})}{|Q|} \tag{C.3}$$

$$DF\text{-}IDF_{stream}(Q, S_{t-b \to t}) = \frac{\sum_{qt \in Q} \frac{df(qt, S_{t-w \to t})}{Avg_{df}(Q,S,t,w,b)} \cdot log(1 + \frac{|S_{t-b \to t}|}{df(qt, S_{t-b \to t})})}{|Q|} \tag{C.4}$$

where Q is the query, $|Q|$ is the number of terms in Q, S is a document stream, qt is a query term in Q, $S_{t-w \to t}$ is the set of documents published in stream S during the time period $t - w \to t$ and $tf(qt, S_{t-w \to t})$ and $df(qt, S_{t-w \to t})$ are the term and document frequencies of qt in $S_{t-w \to t}$ respectively. b is a background window size larger than w, that is used to calculate the background TF and DF. We use a b size of 10 days in this work. $log(1 + \frac{|S_{t-b \to t}|}{df(qt, S_{t-b \to t})})$ is the inverse document frequency (IDF) component, where $|S_{t-b \to t}|$ is the total number of documents in $S_{t-b \to t}$, and $df(qt, S_{t-b \to t})$ is the total document frequency of term qt in $S_{t-b \to t}$. $Avg_{tf/df}(Q, S, t, w, b)$ is the tf/df normalising factor, that takes into account the expected background term and document frequency for periods of length w and is calculated as follows:

$$Avg_{tf}(Q, S, t, w, b) = \frac{\sum_{0 < \Delta t < \frac{b}{w}} TF_{stream}(Q, S_{t-w*(\Delta t+1) \to t-w*\Delta t})}{|\frac{b}{w}|} \tag{C.5}$$

$$Avg_{df}(Q, S, t, w, b) = \frac{\sum_{0 < \Delta t < \frac{b}{w}} DF_{stream}(Q, S_{t-w*(\Delta t+1) \to t-w*\Delta t})}{|\frac{b}{w}|} \tag{C.6}$$

where Δt is the index of a period of length w ranging from 1 to the number of periods of length w contained within the background window of length b.

Our previous example illustrated in Figure 7.2 showed how 12 different features can be generated from a single query made at a point in time. In particular, it shows how different features are generated for that query by leveraging evidence from each of the BBC news articles, Web search queries and blog posts, using two periods of time pre-dating each query (history), i.e. one hour before and 24 hours before and using two different types of features (DF and TF-IDF). Hence, 3 content sources * 2 time periods * 2 feature types = 12 individual features.

We extract generic query frequency features from both $NQC_{May2006}$ and $NQC_{Apr2012}$, resulting in 56 individual features in each case. For $NQC_{May2006}$, these features are extracted from each of the seven news and user-generated streams. Meanwhile for $NQC_{Apr2012}$, these features are extracted from all of the streams except Wikipedia, since the Wikipedia dump used pre-dates the query time and hence term frequencies would be out-of-date.

C.3 Retrieval Features

The third group of features that we use are query retrieval features. These features correspond to the sum of the retrieval scores for the top documents from each corpus returned for the query using various weighting models. As before, we limit ourselves to only the last hour and day of publications made before the query was issued. The intuition behind these features is that news queries should return highly relevant documents when searching from a news corpus. Within this feature set, we experiment with a range of effective retrieval models, covering probabilistic, Divergence From Randomness (DFR) and language modelling techniques (see Section 2.3). In particular, we use the well-known BM25 probabilistic model (Robertson *et al.*, 1994) with default settings, the parameter-free DPH model from the DFR framework (Amati., 2003) and a language modelling approach using Dirichlet smoothing with default parameters (Zhai & Lafferty, 2004). Specifically, we sum the retrieval scores for the top r documents using these three ranking models from each of the seven streams from the period of the hour (w=1) and the day (w=24) preceding each query. For example, the BM25 retrieval feature for a query Q, corpus C and a time period w is calculated as:

$$BM25(Q, S_{t-w\rightarrow t}) = \sum_{rank=1}^{r} bm25(Q, S_{t-w\rightarrow t})[rank] \tag{C.7}$$

where r is the number of top documents to consider, and $bm25(Q, S_{t-w\rightarrow t})[rank]$ is the BM25 retrieval score (see Equation 2.4) for the document at rank $rank$ returned when searching the recent document set $S_{t-w\rightarrow t}$ using query Q.

On $NQC_{May2006}$, we generate features where $1 \leq r \leq 10$, i.e. we consider up to the top 10 documents returned. Using all seven streams, two time periods, three retrieval models and ten r values, we total 420 query retrieval features. Subsequent analysis of the classification results on this dataset indicated that considering only a few highly ranked results resulted in redundant features. Hence, for the $NQC_{Apr2012}$ dataset, we instead test a wider range of retrieval ranks, i.e. $r \in$ [1,5,10,100,1000]. Using the eight streams, two time periods, three retrieval models and five r values, this results in 240 query retrieval features.

C.4 Burstiness Features

We hypothesise that terms of news queries will often coincide with terms that are currently undergoing a burst (Kleinberg, 2002) in newswire and user-generated sources. In particular, we first leverage two state-of-the-art burst detection techniques to identify bursty terms from our parallel news and user-generated content streams. Then we score each news query by the number of bursty terms that it contains, normalised by the query length.

An intuitive method for detecting bursts within a stream is to monitor a term's usage over time and to use a threshold above which a term is considered bursty. We employ a state-of-the-art instance of this style of burst detection proposed by Vlachos *et al.* (2004). In particular, the approach calculates a '*moving*' document frequency average over time. A term is considered bursty if the current document frequency (for a fixed time period) is markedly higher than the standard deviation of the moving average. We use this to create a list of bursty terms for the day of the query. The number of bursty terms that a query contains acts as our first burstiness feature. Note that we use the same time window notation as in Section C.2. In particular, whether a single query term is bursty is calculated as follows:

$$isBursty(qt,S,t,w,b,i) = \begin{cases} 1 & \text{if } \textit{DF-IDF}_{stream}(qt,S,t,w,b) - Mean_{DF-IDF_{stream}}(qt,S,t,w,b) \\ & \quad > (StdDevMean_{DF-IDF}(qt,S,t,w,b)) * i), \\ 0 & \text{otherwise.} \end{cases} \tag{C.8}$$

Again, qt is the query term, S is the document stream, t is the time at which the query was made, w is the size of time window to consider preceding t, while b is the larger background window from which to calculate the moving average. Note that in effect, the positive case of the above equation just compares the current deviation from the mean to the standard deviation from the mean, where 'current' is defined in terms of the most recent period of size w ending at t. If the score for a term is more than i times the standard deviation, then it is considered to be bursty. $Mean_{DF-IDF_{stream}}$ calculates the mean of the *DF-IDF*$_{stream}$ scores for time periods of size w in the background window of size b as follows:

$$Mean_{DF-IDF_{stream}}(qt, S, t, w, b) = \frac{\sum_{0<\Delta t<\frac{b}{w}} \textit{DF-IDF}_{stream}(Q, S_{t-w*(\Delta t+1)\rightarrow t-w*\Delta t})}{|\frac{b}{w}|} \tag{C.9}$$

where Δt is the index of a time period within b. Meanwhile the standard deviation from the mean over b ($StdDevMean_{DF-IDF}(qt, S, t, w, b)$) is calculated as follows:

$$StdDevMean_{DF-IDF}(qt, S, t, w, b) =$$

$$\sqrt{\frac{\sum_{0<\Delta t<\frac{b}{w}} (\textit{DF-IDF}_{stream}(qt, S_{t-w*(\Delta t+1)\rightarrow t-w*\Delta t}) - Mean_{DF-IDF_{stream}}(qt, S, t, w, b))^2}{|\frac{b}{w}| - 1}} \tag{C.10}$$

where *DF-IDF*$_{stream}$ is calculated as per Equation (B.4) and $Mean_{DF-IDF_{stream}}$ is calculated as per Equation (B.9). Importantly, some terms are more bursty than others. This is the reason for the parameter i, as it allows the generation of multiple features each defining to what degree a term is bursty, e.g. 3 times the mean or 6 times the mean. However, from a learning perspective, it is instead more advantageous to define a feature encapsulating the number of times over the standard deviation a term's current popularity is, rather than using the multiplier to create a threshold. Intuitively, this should be a more effective way to encapsulate the evidence, as FANS will learn how to incorporate burstiness into the model, rather than relying on arbitrarily defined thresholds. In practice, we define two features, namely: # BurstyTerms which is the number of terms in the query where $isBursty(qt, S, t, w, b, i)$ is true (where the threshold i was set to 2 times mean); and BurstMagnitude that sums for query terms, the number of times over the standard deviation from the mean their popularity is.

Each of these features encapsulates a time period defined by b. However, it is intuitive that bursty terms spanning multiple time periods are likely to provide stronger evidence. Chen *et al.* (2010) proposed an extension to Vlachos *et al*'s approach, where the bursty terms from multiple time periods are selected. We use this enhancement to create a smaller set of highly bursty terms, again leveraging these as features. In particular, for the $NQC_{May2006}$ dataset, we calculate this smaller set of terms over all seven streams and four scoring methods, contributing 28 features. This approach is not used on the $NQC_{Apr2012}$ dataset due to the large computational overhead involved when calculating features from multiple windows simultaneously, especially for streams with hundreds of thousands of documents like Twitter.

For $NQC_{May2006}$ we extract a total of 478 generic query burstiness features. 448 of these are # Bursty Terms and BurstMagnitude features, extracted for eight different background time periods

(b values), i.e. 1,2,4,8,16,32,64,128 hours before the query time and for each of the seven streams. Furthermore, unlike in the work of Vlachos *et al.* (2004), on $NQC_{May2006}$ we use each of the four generic query frequency features, defined in Equations (B.1) to (B.4), i.e. *TF*$_{stream}$, *DF*$_{stream}$, *TF-IDF*$_{stream}$ and *DF-IDF*$_{stream}$ for term scoring. Hence, 2 features * 8 time periods * 7 streams * 4 term scoring methods = 448 features. The 28 features from the approach by Chen *et al.* (2010)'s are added to the 448 features, forming this entire feature set.

In contrast, to reduce computational overheads and limit the size of our overall feature set, we extract fewer burstiness features from the $NQC_{Apr2012}$ dataset. In particular, we extract only the BurstMagnitude feature, for two time periods, i.e [6,14] hours before the query time and use only *DF-IDF*$_{stream}$ for term scoring. As with our generic query frequency features, burstiness features are not extracted from Wikipedia on $NQC_{Apr2012}$ for technical reasons (see Section 5.2.2). This results in 1 feature * 2 time periods * 7 streams * 1 term scoring method = 14 features. For both datasets w – the current time period – is fixed to one hour.

C.5 Importance Features

In Chapter 6 we introduced voting approaches for estimating the importance of a news story at a point in time. These approaches can also be deployed on our different content streams to estimate the importance of a query based upon each of those streams. In turn, these importance estimations may be useful to our learner for identifying news-related queries, since a high importance estimation indicates that the topic of the query is seeing elevated levels of discussion. We add a fifth feature set, referred to as *Importance*, that contains our top news identification approaches applied upon each content stream. In particular, we evaluate Votes, RWA and $RWA_{GaussBoost}$ when estimating the importance of queries. Notably, these approaches have parameters that can effect the importance estimation, as described in Section 6.2.1. By varying these parameters, different importance estimations, and hence features can be generated. For Votes, we use a default w value of 24 hours, and an r value of 6 hours. The underlying retrieval approach returns the top 100 documents from the stream in question using the DPH weighting model. For RWA, we test a wider range of parameter settings. We vary only one parameter at a time, using the same default setting as Votes for the other parameters. In particular, we independently test w values of [1,3,6,12,24,48] hours and retrieval depths $|R()|$ of [10,20,30,40,50,100,200,300,500,1000] documents. For $RWA_{GaussBoost}$, we use the same default settings as for Votes, however, we vary the Gaussian curve width parameter l for values ranging from 0.2 to 3.0 in increments of 0.2.

C.6 Stream-Specific Features

The final of our feature sets is stream-specific features. A stream-specific feature is extracted in a similar manner to a retrieval feature, i.e. scores for documents are aggregated together from a stream of content S bounded by a time window of size w that ends at time t. However, instead of scoring documents based on a generic feature like its retrieval score, we instead use a stream-specific feature, that is (typically) independent of the query. A single stream-specific feature is the aggregate of the features of the top documents retrieved for the query, in a similar way to the our query retrieval features. The idea is that these features will provide the learner with a different type of evidence to those features that use the retrieval scores of related documents. In general, the stream-specific features are calculated as follows:

$$feature(Q, S_{t-w\rightarrow t}) = \sum_{rank=1}^{r} f(R(Q, S_{t-w\rightarrow t})[rank]) \tag{C.11}$$

where r is the number of top documents to consider, $R(Q, S_{t-w\rightarrow t})[rank]$ returns the document at rank $rank$ retrieved using a document weighting model (see Section 2.3) when searching the recent document set $S_{t-w\rightarrow t}$ using query Q and $f(R(Q, S_{t-w\rightarrow t})[rank])$ returns the stream-specific feature f for document $R(Q, S_{t-w\rightarrow t})[rank]$. The document weighting model used to retrieve diggs is the parameter-free DPH model from the DFR framework (Amati., 2003), while to retrieve tweets we use the DFReeKLIM (Amati *et al.*, 2011) document weighting model that is specifically designed for ranking short texts.

We only extract stream-specific features from the $NQC_{Apr2012}$ dataset because the streams in the $NQC_{May2006}$ dataset provided only the raw text for each document. The Digg and Twitter streams from the $NQC_{Apr2012}$ dataset in contrast, contain metadata about each digg and tweet respectively. For example, each digg lists the number of times the linked article has been 'digged' before. As a result, for our experiments on $NQC_{Apr2012}$, we enrich our feature set with corpus specific features from these datasets in addition to a similar set of generic stream features as used previously.

From the Digg corpus, we extract three additional corpus specific features. Firstly, *# Poster Profile Views* counts for each of the top diggs retrieved for the query, number of times the person who made each digg has had their profile viewed. This feature provides an aggregate authority score for the top diggs, if many related diggs are made by popular users, then the query that those diggs were returned for will be more likely to be related to an important (newsworthy) topic. The second feature is *# Comments*, that counts the number of comments made for each of the top diggs retrieved. If diggs with many comments are retrieved for a query, then this is an indicator that the query topic is controversial, and hence is interesting and/or newsworthy. Finally, the third feature that we add is the *# Diggs*. This feature records the number of times each of the diggs returned for the query have previously been digged. This feature encapsulates how many digg users found documents relating to the query topic important enough to digg them.

In contrast to Digg, Twitter provides information about the tweet and its author (see Section 3.4.1). From the information available to us, we select the following six metadata items to form features. Firstly, *Retweet Count* measures the number of times the top retrieved tweets for a query have been retweeted. The more retweeted tweets are retrieved for a query, the more likely that the query topic is to be important/newsworthy (Suh *et al.*, 2010). Next, *User Statuses Count* provides the number of prior tweets made by the retrieved tweet author. As a corpus specific feature, this returns the number of prior posts by all users in the set of retrieved tweets. The idea is that queries that return tweets by active users, i.e. those with a high number of prior posts, are more likely to be news-related. Similarly, our fourth feature – *User Favourites Count* – is another measure of user activity, this time of the number of other tweets each author has favourites. Finally, our last three features incorporate information about the tweet author's social graph. In particular, *User Friends Count*, *User Followers Count* and *User Listed Count* each measure the number of friends, followers and lists that the user appears in, respectively. These features are once again aggregated over the top tweets retrieved for a query. The idea is that the more social connections that a user has, the more influential that user is (Cha *et al.*, 2010). Hence, if influential users are tweeting about the topic of the query then that news story is more likely to be related to current news.

Appendix D

News Query Classification Learners

In our news query classification experiements in Chapter 7, we use a linear logistic regression trees learner (Logitboost) (Landwehr *et al.*, 2003) to produce classification models. However, as described in Section 2.5, other learners may produce more effective models (at the cost of training time). In this appendix, we report the classification effectiveness when using a suite of alternative learning algorithms.

Table D.1 reports news query classification effectiveness when using features from all of our streams under different learning algorithms. Statistical significance over Logistic Regression (Multinomial) (paired t-test $p<0.05$) denoted ▲. From Table D.1, we see that the learner used can have a large impact upon the end news query classification effectiveness. In particular, simple Logistic Regression (Le Cessie & Van Houwelingen, 1992), produces the least effective classification model. The statistical NaiveBayes and tree-based J48 Tree and RandomForest learners produce the next most effective models. Finally, the support-vector machine (SVM) approach SMO and Logitboost that we used previously produce the best models over all. Indeed, under the combined F_1 measure, SMO produced the best classification model, resulting in 0.8344 F_1, although this is not significantly better than the next best alternative (Logitboost) that we used in our previous experiment. This justifies our use of the Logitboost classifier in Chapter 7.

Learner	Precision	Recall	F_1
Logistic Regression (Multinomial)	0.7177	0.6572	0.7012
Logitboost (Linear Logistic Regression Trees)	0.8304▲	0.8032▲	0.8138▲
NaiveBayes	0.7996▲	0.6912	0.7383
J48 Tree	0.7914	0.7989▲	0.7383▲
RandomForest	0.7840	**0.8429▲**	0.7998▲
SMO	**0.8516▲**	0.8240▲	**0.8344▲**

Table D.1: News Query classification effectiveness when using all features under a variety of learning algorithms. Statistical significance over Logistic Regression (Multinomial) (paired t-test $p<0.05$) denoted ▲.

Bibliography

Adar, E., Teevan, J., Dumais, S. T. & Elsas, J. L. (2009). The Web changes everything: Understanding the dynamics of Web content. *In* 'Proceedings of the 2nd ACM International Conference on Web Search and Data Mining'. 2.2.6

Agichtein, E., Brill, E. & Dumais, S. T. (2006). Improving Web search ranking by incorporating user behavior information. *In* 'Proceedings of the 29th International ACM SIGIR Conference on Research and Development in Information Retrieval'. 2.5

Aha, D. (1997). *Lazy learning*. Kluwer Academic Publishers. 2.5.1

Ahn, B., Van Durme, B. & Callison-Burch, C. (2011). WikiTopics: what is popular on Wikipedia and why. *In* 'Proceedings of the Association for Computational Linguistics: Human Language Technologies'. 3.6.1

Al-Maskari, A., Sanderson, M. & Clough, P. (2008). Relevance judgments between TREC and Non-TREC assessors. *In* 'Proceedings of the 31st International ACM SIGIR Conference on Research and Development in Information Retrieval'. 5.4.5

Allan, J. (2002). *Topic detection and tracking*. Springer. 2.6, 2.6.1

Allan, J., Carbonell, J., Doddington, G., Yamron, J., Yang, Y., Amherst, U. & Umass, J. A. (1998). Topic detection and tracking pilot study. *In* 'Proceedings of the DARPA Broadcast News Transcription and Understanding Workshop'. 2.6

Alonso, O. & Baeza-Yates, R. (2011). Design and implementation of relevance assessments using crowdsourcing. *In* 'Advances in Information Retrieval'. Vol. 6611. pp. 153–164. 5.6.2

Alonso, O., Rose, D. E. & Stewart, B. (2008). Crowdsourcing for relevance evaluation. *ACM SIGIR Forum* **42**(2), 9–15. 5.1, 5.2, 5.3, 5.3, 5.4.4, 5.6.4

Amati., G. (2003). Probabilistic Models for Information Retrieval based on Divergence from Randomness. PhD thesis. University of Glasgow. 2.3.3, 2.3.5, 2.3.5, 2.3.5, 2.7.1, 6.4.2, C.3, C.6

Amati, G., Ambrosi, E., Bianchi, M., Gaibisso, C. & Gambosi, G. (2007). FUB, IASI-CNR and University of Tor Vergata at TREC 2007 Blog Track. *In* 'Proceedings of the 16th Text REtrieval Conference'. 6.4.1

Amati, G., Amodeo, G., Bianchi, M., Marcone, G., Gaibisso, C., Celi, A., De Nicola, C. & Flammini, M. (2011). FUB, IASI-CNR, UNIVAQ at TREC 2011. *In* 'Proceedings of the 20th Text REtrieval Conference'. 3.4.3, 6.4.2, C.6

Arguello, J., Diaz, F. & Callan, J. (2011). Learning to aggregate vertical results into Web search results. *In* 'Proceedings of the 20th ACM Conference on Information and Knowledge Management'. 2.5

Arguello, J., Diaz, F. & Paiement, J.-F. (2010). Vertical selection in the presence of unlabeled verticals. *In* 'Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval'. 3.5.2

Arguello, J., Diaz, F., Callan, J. & Crespo, J.-F. (2009). Sources of evidence for vertical selection. *In* 'Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval'. 1.2, 2.8.2, 3.5.2, 4.8

Arguello, J., Elsas, J., Callan, J. & Carbonell, J. (2008). Document representation and query expansion models for blog recommendation. *In* 'Proceedings of the 2nd International AAAI Conference on Weblogs and Social Media'. 8.2.1

Aslam, J. & Pavlu, V. (2007). A practical sampling strategy for efficient retrieval evaluation. Technical report. North Eastern University. 5.4.1

Atwood, J. (2010). 'Is Amazon's Mechanical Turk a Failure?'. `http://www.codinghorror.com/blog/2007/04/is-amazons-mechanical-turk-a-failure.html`, accessed on 30/09/2012. 5.3

Baeza-Yates, R. & Ribeiro-Neto, B. (1999). *Modern Information Retrieval*. Addison Wesley. 2.1, 2.2.2, 2.2.3, 2.3, 3.2.1

Baeza-Yates, R., Hurtado, C. & Mendoza, M. (2005). Query recommendation using query logs in search engines. *In* 'Proceedings of the Current Trends in Database Technology - EDBT 2004 Workshops'. 10.2

Bailey, P., Craswell, N., Soboroff, I., Thomas, P., de Vries, A. P. & Yilmaz, E. (2008). Relevance assessment: Are judges exchangeable and does it matter. *In* 'Proceedings of the 31st International ACM SIGIR Conference on Research and Development in Information Retrieval'. 5.3, 5.6.5

Balog, K. & de Rijke, M. (2006). Finding experts and their details in e-mail corpora. *In* 'Proceedings of the 15th International World Wide Web Conference'. 2.7.1

Bandari, R., Asur, S. & Huberman, B. (2012). The pulse of news in social media: Forecasting popularity. *In* 'Proceedings of the 6th International AAAI Conference on Weblogs and Social Media'. 1.1

Bar-Ilan, J., Zhu, Z. & Levene, M. (2009). Topic-specific analysis of search queries. *In* 'Proceedings of the 2009 workshop on Web Search Click Data'. 1.1, 4.3, 4.6, 7.3, C.1

Beeferman, D. & Berger, A. L. (2000). Agglomerative clustering of a search engine query log. *In* 'Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data mining'. 10.2

Beitzel, S. M., Jensen, E. C. & Frieder, O. (2005). Improving automatic query classification via semi-supervised learning. *In* 'Proceedings of the 5th IEEE International Conference on Data Mining'. 2.8.2

Beitzel, S. M., Jensen, E. C., Chowdhury, A., Grossman, D. A. & Frieder, O. (2004). Hourly analysis of a very large topically categorized Web query log. *In* 'Proceedings of the 27th International ACM SIGIR Conference on Research and Development in Information Retrieval'. 3.5.1

Beitzel, S. M., Jensen, E. C., Lewis, D. D., Chowdhury, A. & Frieder, O. (2007). Automatic classification of Web queries using very large unlabeled query logs. *ACM Transactions on Information Systems* **25**(2), 9. 2.8.2

Bell, A. (1991). *The language of news media*. Blackwell Oxford. 4.5

Benjamin, C. A. (2009). Low-Cost and Robust Evaluation of Information Retrieval Systems. PhD thesis. University of Massachusetts Amherst. 9.3

Berger, A. & Lafferty, J. (1999). Information retrieval as statistical translation. *In* 'Proceedings of the 22nd International ACM SIGIR Conference on Research and Development in Information Retrieval'. 2.3.4

Blei, D., Ng, A. & Jordan, M. (2003). Latent dirichlet allocation. *Machine Learning Research* **3**, 993–1022. 3.4.4

Blood, R. (2002). *The weblog handbook: Practical advice on creating and maintaining your blog*. Basic Books. 3.2

Bollen, J., Pepe, A. & Mao, H. (2009). Modeling public mood and emotion: Twitter sentiment and socio-economic phenomena. *In* 'Proceedings of the 18th International World Wide Web Conference'. 3.4.1

Boyd, D., Golder, S. & Lotan, G. (2010). Tweet, tweet, retweet: Conversational aspects of retweeting on twitter. *In* 'Proceedings of the International Conference on System Sciences'. 3.4.1, 4.7.2

Brenes, D. J. & Gayo-avello, D. (2009). Stratified analysis of AOL query log. *Information Sciences* **179**, 1844–1858. 3.1, 3.5

Brin, S. & Page, L. (1998). The anatomy of a large-scale hypertextual Web search engine. *Computer networks and ISDN systems* **30**(1-7), 107–117. 2.2.4, 4.8

Broder, A. (2002). A taxonomy of Web search. *ACM SIGIR Forum* **36**(2), 3–10. 3.5.2, C.1

Broder, A. Z., Fontoura, M., Gabrilovich, E., Joshi, A., Josifovski, V. & Zhang, T. (2007). Robust classification of rare queries using Web knowledge. *In* 'Proceedings of the 30th International ACM SIGIR Conference on Research and Development in Information Retrieval'. 3.5.2

Büttcher, S. & Clarke, C. (2005). Efficiency vs. Effectiveness in Terabyte-scale Information Retrieval. *In* 'Proceedings of the 14th Text REtrieval Conference'. 2.3, 2.3.2

Callan, J. (2000). Distributed information retrieval. *In* W. B. Croft, ed., 'Advances in Information Retrieval'. Kluwer Academic Publishers. chapter 5, pp. 127–150. 2.8, 2.8.1, 2.8.3

Callan, J. P., Lu, Z. & Croft, W. B. (1995). Searching distributed collections with inference networks. *In* 'Proceedings of the 18th International ACM SIGIR Conference on Research and Development in Information Retrieval'. 2.8.1

Callison-Burch, C. (2009). Fast, Cheap, and Creative: Evaluating Translation Quality Using Amazon's Mechanical Turk. *In* 'Proceedings of the Conference on Empirical Methods in Natural Language Processing'. 5.3.1

Carbonell, J. (1990). *Machine learning: paradigms and methods*. Elsevier North-Holland, Inc. 2.5

Castillo, C. & Davison, B. D. (2010). Adversarial Web Search. *Foundations and Trends in Information Retrieval* **4**, 377–486. 3.2.2

Cha, M., Haddadi, H., Benevenuto, F. & Gummadi, K. (2010). Measuring user influence in Twitter: The million follower fallacy. *In* ‘Proceedings of the 4th International AAAI Conference on Weblogs and Social Media’. C.6

Chawla, N. V., Japkowicz, N. & Kotcz, A. (2004). Editorial: Special issue on learning from imbalanced data sets. *ACM SIGKDD 2004 Explorations Newsletter* **6**(1), 1–6. 7.3

Chen, W., Chen, C., Zhang, L.-j., Wang, C. & Bu, J.-j. (2010). Online detection of bursty events and their evolution in news streams. *Journal of Zhejiang University - Science C* **11**, 340–355. C.4

Chenliang, L., Jianshu, W., Qi, H., Yuxia, Y., Anwitaman, D., Aixin, S. & Bu-Sung, L. (2012). TwiNER: Unsupervised Named Entity Recognition in Targeted Twitter Stream. *In* ‘Proceedings of the 35nd International ACM SIGIR Conference on Research and Development in Information Retrieval’. 3.4.1

Chien, S. & Immorlica, N. (2005). Semantic similarity between search engine queries using temporal correlation. *In* ‘Proceedings of the 14th International World Wide Web Conference’. 3.5.1

Ciglan, M. & Nørvåg, K. (2010). WikiPop: Personalized event detection system based on Wikipedia page view statistics. *In* ‘Proceedings of the 19th ACM Conference on Information and Knowledge Management’. 3.6.1

Clarke, C. L., Kolla, M., Cormack, G. V., Vechtomova, O., Ashkan, A., Büttcher, S. & MacKinnon, I. (2008). Novelty and diversity in Information Retrieval evaluation. *In* ‘Proceedings of the 31st International ACM SIGIR Conference on Research and Development in Information Retrieval’. 2.3, 8.3.1

Cleverdon, C. W. (1991). The significance of the Cranfield tests on index languages. *In* ‘Proceedings of the 14th International ACM SIGIR Conference on Research and Development in Information Retrieval’. 2.4.2

Craswell, N. (2000). Methods for distributed Information Retrieval. PhD thesis. Australian National University. 2.8, 4.8

Craswell, N., Bailey, P. & Hawking, D. (2000). Server selection on the World Wide Web. *In* ‘Proceedings of the 5th ACM Conference on Digital libraries’. 2.8.1

Craswell, N., Hawking, D., Vercoustre, A., & P., W. (2004). Panoptic expert: Searching for experts not just for documents. *In* ‘Proceedings of the 7th Austraasian World Wide Web Conference’. 2.7, 2.7.1

Craswell, N., Jones, R., Dupret, G. & Viegas, E. (2009). Workshop on Web search click data 2009. *In* 'Proceedings of the 2009 Workshop on Web Search Click Data'. 3.1, 3.5, 4.3, 5.5.1

Croft, W. B. & Harper, D. (1988). Using probabilistic models of Information Retrieval without relevance information. *Journal of Documentation* **35**, 285–295. 2.3.2

Croft, W. B. & Lafferty, J. (2003). *Language Modeling for Information Retrieval*. Vol. 13. Kluwer Academic Publishers. 2.3.4

Crum, C. (2010). 'Google reveals factors for ranking tweets'. `http://www.webpronews.com/google-reveals-factors-for-ranking-tweets-2010-01`, accessed on 30/09/2012. 3.4.2

Dai, N., Shokouhi, M. & Davison, B. D. (2011). Learning to rank for freshness and relevance. *In* 'Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval'. 2.5

Davis, M. (2006). 'Scale up vs. scale out'. `http://weblogs.java.net/blog/malcolmdavis/archive/2006/07/scale_up_vs_sca.html`, accessed on 30/09/2012. 2.2.5

Dean, J. & Ghemawat, S. (2004). Mapreduce: Simplified data processing on large clusters. *In* 'Proceedings of the 10th Symposium on Open Systems Design and Implementation'. 2.2.5, 2.2.5

Diakopoulos, N. & Shamma, D. (2010). Characterizing debate performance via aggregated Twitter sentiment. *In* 'Proceeding of the 28th Conference on Human Factors in Computing Systems'. 3.4.1

Diamond, T. (2001). Information retrieval using dynamic evidence combination. PhD thesis. Syracuse University. 4.6

Diaz, F. (2009). Integration of news content into Web results. *In* 'Proceedings of the 2nd ACM International Conference on Web Search and Data Mining'. 2.8.2, 3.5.2, 5.2.2, 7.4, 7.5, 7.6, 7.6.1

Diaz, F. & Jones, R. (2004). Using temporal profiles of queries for precision prediction. *In* 'Proceedings of the 27th International ACM SIGIR Conference on Research and Development in Information Retrieval'. 3.5.1

Diaz, F. & Metzler, D. (2006). Improving the estimation of relevance models using large external corpora. *In* 'Proceedings of the 29th International ACM SIGIR Conference on Research and Development in Information Retrieval'. 2.3.5

Downs, J., Holbrook, M., Sheng, S. & Cranor, L. (2010). Are your participants gaming the system? screening mechanical turk workers. *In* 'Proceeding of the 28th Conference on Human Factors in Computing Systems'. 5.3

D'Souza, D., Zobel, J. & Thom, J. (2004). Is CORI effective for collection selection? an exploration of parameters, queries, and data. *In* 'Proceedings of the Australian Document Computing Symposium'. 2.8.1

Duan, Y., Jiang, L., Qin, T., Zhou, M. & Shum, H. (2010). An empirical study on learning to rank of tweets. *In* 'Proceedings of the 23rd International Conference on Computational Linguistics'. 3.4.2, 4.7.2, 8.2.2

Eastman, C. M. & Jansen, B. J. (2003). Coverage, Relevance, and Ranking: The Impact of Query Operators on Web Search Engine Results. *ACM Transactions on Information Systems* **21**, 383–411. 3.5.1

Efron, M. (2011). Information search and retrieval in microblogs. *American Society for Information Science and Technology*. 3.4.1, 4.7.2

Eickhoff, C. & de Vries, A. (2011). How Crowdsourcable is Your Task?. *In* 'Proceedings of the Workshop on Crowdsourcing for Search and Data Mining'. 5.3

Elias, P. (1975). Universal codeword sets and representations of the integers. *IEEE Transactions on Information Theory* **21**(2), 194–203. 2.2.4

Ernsting, B., Weerkamp, W. & Rijke, M. D. (2007). Language Modeling Approaches to Blog Post and Feed Finding. *In* 'Proceedings of the 16th Text REtrieval Conference'. 3.2.2

Fang, H. & Zhai, C. (2007). Probabilistic models for expert finding. *Advances in Information Retrieval* pp. 418–430. 2.7.1

Fiona McCann (2012). 'State media versus social media: What's happening in Damascus?'. `http://storyful.com/stories/36037`, accessed on 30/09/2012. 1.2

Fleiss, J. L. (1971). Measuring nominal scale agreement among many raters. *Psychological Bulletin* **76**(5), 378–382. 5.3.2, 5.3.2, 5.4.5

Freund, L., Berzowska, J., Lee, J., Read, K. & Schiller, H. (2011). Digging into Digg : genres of online news. *In* 'Proceedings of the ACM iConference'. 3.3.1

Fürnkranz, J. (1999). Separate-and-conquer rule learning. *Artificial Intelligence Review* **13**(1), 3–54. 2.5.1

Galtung, J. & Ruge, M. H. (1965). The structure of foreign news: the presentation of the Congo, Cuba and Cypris crises in four Norwegian newspapers. *The Journal of Peace Research* **2**(1), 64–90. 4.5

Ghemawat, S., Gobioff, H. & Leung, S.-T. (2003). The Google file system. *ACM SIGOPS* **37**(5), 29–43. 2.2.5

Halevy, A., Norvig, P. & Pereira, F. (2009). The unreasonable effectiveness of data. *IEEE Intelligent Systems* **24**(2), 8–12. 2.2.5

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P. & Witten, I. (2009). The WEKA data mining software: an update. *ACM SIGKDD 2009 Explorations Newsletter* **11**(1), 10–18. 2.5.3

Hansell, S. (2007). 'Google keeps tweaking its search engine'. `http://www.nytimes.com/2007/06/03/business/yourmoney/03google.html`, accessed on 30/09/2012. 1.1, 1.2, 4.6

Harter, S. P. (1975). An algorithms for probabilistic indexing. *American Society for Information Science* **26**(4), 280–289. 2.3.2

Hassan Sayyadi, Matthew Hurst, A. M. (2009). Event detection and tracking in social streams. *In* 'Proceedings of the 3th International AAAI Conference on Weblogs and Social Media'. 3.2.4

He, B., Macdonald, C. & Ounis, I. (2008). Retrieval sensitivity under training using different measures. *In* 'Proceedings of the 31st International ACM SIGIR Conference on Research and Development in Information Retrieval'. 2.4.4

He, B., Macdonald, C., He, J. & Ounis, I. (2008). An effective statistical approach to blog post opinion retrieval. *In* 'Proceeding of the 17th ACM Conference on Information and Knowledge Management'. 8.2.1

He, C., Hong, D. & Si, L. (2011). A weighted curve fitting method for result merging in federated search. *In* 'Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval'. 2.8.3

Hellmann, R., Griesbaum, J. & Mandl, T. (2010). *Quality in Blogs: How to Find the Best User Generated Content*. 3.2.2

Hoelscher, C. (1998). How Internet experts search for information on the Web. *In* 'Proceedings of the World Conference of the World Wide Web, Internet, and Intranet'. 3.5

Howe, J. (2009). *Crowdsourcing: Why the Power of the Crowd Is Driving the Future of Business*. Three Rivers Pr. 5.3

Howe, J. (2010). 'The rise of Crowdsourcing'. `http://www.wired.com/wired/archive/14.06/crowds.html`, accessed on 30/09/2012. 5.1, 5.3

Huang, L., Wang, L. & Li, X. (2008). Achieving both high precision and high recall in near-duplicate detection. *In* 'Proceeding of the 17th ACM Conference on Information and Knowledge Management'. 2.2.6

Ipeirotis, P. G. (2011). Crowdsourcing using Mechanical Turk: Quality Management and Scalability. *In* 'Proceedings of the Workshop on Crowdsourcing for Search and Data Mining'. 5.3.1, 5.3.2, 9.5.5

Isard, M., Budiu, M., Yu, Y., Birrell, A. & Fetterly, D. (2007). Dryad: distributed data-parallel programs from sequential building blocks. *In* 'Proceedings of the EuroSys Conference'. 2.2.5

Ivakhnenko, A. G. (1975). Principles of neurodynamics. *Cybernetics and Systems Analysis* **11**, 841–842. 10.1007/BF01071380. 2.5.1

Jansen, B. J. & Pooch, U. W. (2001). A review of Web searching studies and a framework for future research. *American Society for Information Science and Technology* **52**, 235–246. 3.5.1

Jansen, B. J., Booth, D. L. & Spink, A. (2008). Determining the informational, navigational, and transactional intent of Web queries. *Information Processing and Management* **44**, 1251–1266. 3.5.2

Jansen, B. J., Spink, A. & Saracevic, T. (2000). Real life, real users, and real needs: a study and analysis of user queries on the web. *Information Processing and Management* **36**, 207–227. 3.5.1

Jansen, B. J., Spink, A., Bateman, J. & Saracevic, T. (1998). Real life Information Retrieval: A study of user queries on the Web. *ACM SIGIR Forum* **32**, 5–17. 3.5, 3.5.1

Jansen, B., Zhang, M., Sobel, K. & Chowdury, A. (2009). Twitter power: Tweets as electronic word of mouth. *American Society for Information Science and Technology* **60**(11), 2169–2188. 3.4.1

Järvelin, K. & Kekäläinen, J. (2002). Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems* **20**(4), 422–446. 2.4.1.3, 2.4.1.3, 6.4.2

Jensen, F. (1996). *An introduction to Bayesian networks*. Vol. 36. UCL press London. 2.5.1

Johnson, R. E. (1997). Frameworks = (components + patterns). *Communications of the ACM* **40**(10), 39–42. 2.2.5

Jones, R. & Diaz, F. (2007). Temporal profiles of queries. *ACM Transactions on Information Systems* **25**(3), 14. 3.5.1, 7.6.2, C.2

Kang, C., Wang, X., Chen, J., Liao, C., Chang, Y., Tseng, B. & Zheng, Z. (2011). Learning to re-rank Web search results with multiple pairwise features. *In* 'Proceedings of the 4th ACM International Conference on Web Search and Data Mining'. 2.5

Kim, M., Xie, L. & Christen, P. (2012). Event diffusion patterns in social media. *In* 'Proceedings of the 6th International AAAI Conference on Weblogs and Social Media'. 3.2.4

Kittur, A., Chi, E. H. & Suh, B. (2008). Crowdsourcing user studies with mechanical turk. *In* 'Proceeding of the 26th Conference on Human Factors in Computing Systems'. 5.3.1, 5.5.3

Kleinberg, J. (2002). Bursty and hierarchical structure in streams. *In* 'Proceedings of the 8th ACM SIGKDD international Conference on Knowledge Discovery and Data Mining'. 3.5.1, 6.2.4, C.4

Kolari, P., Finin, T. & Joshi, A. (2006). SVMs for the blogosphere: Blog identification and splog detection. *In* 'AAAI Spring Symposium on Computational Approaches to Analyzing Weblogs'. Vol. 4. p. 1. 3.2.2

Kolari, P., Java, A. & Finin, T. (2006). Characterizing the splogosphere. *In* 'Proceedings of the 3rd Annual Workshop on Weblogging Ecosystem: Aggregation, Analysis and Dynamics'. 3.2.2

König, A. C., Gamon, M. & Wu, Q. (2009). Click-through prediction for news queries. *In* 'Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval'. 3.2.4, 3.5.2, 7.4, C.1, C.2

Kotsiantis, S., Zaharakis, I. & Pintelas, P. (2007). Supervised machine learning: A review of classification techniques. *Frontiers in Artificial Intelligence and Applications* **160**, 3. 2.5.1

Kulkarni, A., Teevan, J., Svore, K. M. & Dumais, S. T. (2011). Understanding temporal query dynamics. *In* 'Proceedings of the 4th International AAAI Conference on Weblogs and Social Media'. 3.5.1

Kumaran, G. & Carvalho, V. R. (2009). Reducing long queries using query quality predictors. *In* 'Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval'. A.1.1

Kwak, H., Lee, C., Park, H. & Moon, S. (2010). What is Twitter, a Social Network or a News Media. *In* 'Proceedings of the 19th International World Wide Web Conference'. 1.1, 3.4.1, 4.5, 4.7.2

Kwok., K. L. & Chan, M. S. (1998). Improving two-stage ad-hoc retrieval for short queries. *In* 'Proceedings of the 21st International ACM SIGIR Conference on Research and Development in Information Retrieval'. 2.3.5

Landis, J. R. & Koch, G. G. (1977). The measurement of observer agreement for categorical data. *Biometrics* **33**(1), 159–174. 5.5.5

Landwehr, N., Hall, M. & Frank, E. (2003). Logistic model trees. *In* 'Proceedings of the 14th European Conference on Machine Learning'. 2.5.1, 7.3, D

Le Cessie, S. & Van Houwelingen, J. (1992). Ridge estimators in logistic regression. *Applied statistics* pp. 191–201. D

Lee, R. & Sumiya, K. (2010). Measuring geographical regularities of crowd behaviors for Twitter-based geo-social event detection. *In* 'Proceedings of the Workshop on Advances in Geographic Information Systems'. 3.4.4

Lee, Y., Jung, H., Song, W. & Lee, J. (2010). Mining the Blogosphere for Top News Stories Identification. *In* 'Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval'. 3.2.3

Lee, Y., Na, S., Kim, J., Nam, S., Jung, H. & Lee, J. (2008). KLE at TREC 2008 Blog Track: Blog Post and Feed Retrieval. *In* 'Proceedings of the 17th Text REtrieval Conference'. 3.2.2

Leidner, J. L. (2010). 'Thomson Reuters Releases TRC2 News Corpus Through NIST'. `http://jochenleidner.posterous.com/thomson-reuters-releases-research-collection`, accessed on 30/09/2012. 6.4.1, 8.3.1

Lerman, K. (2006). Social Networks and Social Information Filtering on Digg . *Computing Research Repository*. 3.3.2

Lerman, K. (2007). User Participation in Social Media: Digg Study. *Computing Research Repository* **abs/0708.2**, 255–258. 3.3.2

Lerman, K. & Galstyan, A. (2008). Analysis of Social Voting Patterns on Digg . *Computing Research Repository* **abs/0806.1**, 7–12. 3.3.2

Lerman, K. & Ghosh, R. (2010). Information Contagion: an Empirical Study of the Spread of News on Digg and Twitter Social Networks. *Computing Research Repository*. 3.3.2

Li, X., Wang, Y. & Acero, A. (2008). Learning query intent from regularized click graphs. *In* 'Proceedings of the 31st International ACM SIGIR Conference on Research and Development in Information Retrieval'. 2.8.2

Li, Y., Zheng, Z. & Dai, H. (2005). KDD CUP-2005 report: Facing a great challenge. *ACM SIGKDD 2005 Explorations Newsletter* **7**(2), 91–99. 2.8.2

Lih, A. (2004). Wikipedia as Participatory Journalism: Reliable Sources? Metrics for Evaluating Collaborative Media as a News Source. *Nature*. 3.6.1

Lin, Y.-F., Wang, J.-H., Lai, L.-C. & Kao, H.-Y. (2010). Top Stories Identification From Blog to News In TREC 2010 Blog Track. *In* 'Proceedings of the 19th Text REtrieval Conference'. 3.2.3

Lin, Y., Sundaram, H., Chi, Y., Tatemura, J. & Tseng, B. L. (2007). Splog Detection Using Self-similarity Analysis on Blog Temporal Dynamics. *In* 'Proceedings of the International Adversarial Information Retrieval on the Web Workshop series'. 3.2.2

Lioma, C., Macdonald, C., Plachouras, V., Peng, J., He, B. & Ounis, I. (2006). University of Glasgow at TREC 2006: Experiments in Terabyte and Enterprise Tracks with Terrier. *In* 'Proceedings of the 15th Text REtrieval Conference'. 6.4.1

Liu, T.-Y. (2009). Learning to rank for information retrieval. *Foundations Trends Information Retrieval* **3**, 225–331. 2.1, 2.5, 2.5.2, 3.4.2

Liu, X., Croft, W. & Koll, M. (2005). Finding experts in community-based question-answering services. *In* 'Proceedings of the 14th ACM Conference on Information and Knowledge Management'. 2.7.1

Long, R., Wang, H., Chen, Y., Jin, O. & Yu, Y. (2011). Towards Effective Event Detection, Tracking and Summarization on Microblog Data. *In* 'Proceedings of the 12th International Conference on Web-age Information Management'. 3.4.4

Lovins, J. B. (1968). Development of a stemming algorithm. *Mechanical Translation and Computational Linguistics* **11**, 22–31. 2.2.3

Luhn, H. (1957). A statistical approach to mechanized encoding and searching of literary information. *IBM Journal of Research and Development*. 2.2.2

Macdonald, C. (2009). The Voting Model for People Search. PhD thesis. University of Glasgow. 2.1, 2.7, 2.7.1, 2.7.1, 6.2, 6.2.1, 6.4.1, A.1.2

Macdonald, C. & Ounis, I. (2006*a*). Combining fields in known-item email search. *In* 'Proceedings of the 29th International ACM SIGIR Conference on Research and Development in Information Retrieval'. 2.7.1

Macdonald, C. & Ounis, I. (2006*b*). The TREC Blogs06 Collection : Creating and Analysing a Blog Test Collection. 6.4.1

Macdonald, C. & Ounis, I. (2009). Searching for expertise: Experiments with the Voting Model. *Computer* **52**(7), 729–748. 6.2.1

Macdonald, C. & Ounis, I. (2011). Learning models for ranking aggregates. *In* 'Proceedings of 33rd European Conference on Information Retrieval'. 2.7.1, 6.3

Macdonald, C., He, B., Plachouras, V. & Ounis, I. (2005). University of Glasgow at TREC 2005: Experiments in Terabyte and Enterprise Tracks with Terrier. *In* 'Proceedings of the 14th Text REtrieval Conference'. 2.3.5

Macdonald, C., Ounis, I. & Soboroff, I. (2009). Is spam an issue for opinionated blog post search?. *In* 'Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval'. 3.2.2

Macdonald, C., Santos, R. & Ounis, I. (2012). The whens and hows of learning to rank. *Information Retrieval*. 2.5.2, 8.2.2

Macdonald, C., Soboroff, I. & Ounis, I. (2009). Overview of TREC-2009 Blog track. *In* 'Proceedings of the 18th Text REtrieval Conference'. 2.4.4, 3.2.3, 5.2.1, 8.2.1

Manning, C. D., Raghavan, P. & Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press. 2.2.5, 2.3.2, 4.7.2

McCreadie, R. (2010). Leveraging user-generated content for news search. *In* 'Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval'. 1.5

McCreadie, R., Macdonald, C. & Ounis, I. (2009*a*). Comparing distributed indexing: To MapReduce or not?. *In* 'Proceedings of the Workshop on Large-Scale Distributed Systems for Information Retrieval'. 1.5, 2.2.5

McCreadie, R., Macdonald, C. & Ounis, I. (2009*b*). On Single-Pass Indexing with MapReduce. *In* ‘Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval’. 2.2.5

McCreadie, R., Macdonald, C. & Ounis, I. (2010*a*). Crowdsourcing a news query classification dataset. *Proceedings of the Workshop on Crowdsourcing for Search Evaluation*. 1.5

McCreadie, R., Macdonald, C. & Ounis, I. (2010*b*). Insights on the horizons of news search. *In* ‘Proceedings of the 3rd Annual Workshop on Search in Social Media’. 1.5

McCreadie, R., Macdonald, C. & Ounis, I. (2010*c*). News article ranking: Leveraging the wisdom of bloggers. *In* ‘Proceedings of the 9th International Conference on Computer-Assisted Information Retrieval’. 1.5, 6.4.1, 6.4.2, A.1.1

McCreadie, R., Macdonald, C. & Ounis, I. (2011*a*). Crowdsourcing Blog Track Top News Judgments at TREC. *In* ‘Proceedings of the Workshop on Crowdsourcing for Search and Data Mining’. 1.5

McCreadie, R., Macdonald, C. & Ounis, I. (2011*b*). A learned approach for ranking news in real-time using the blogosphere. *In* ‘Proceedings of the 18th International Symposium on String Processing and Information Retrieval’. 1.5

McCreadie, R., Macdonald, C. & Ounis, I. (2011*c*). Mapreduce indexing strategies: Studying scalability and ef?ciency. *Information Processing and Management*. 1.5, 2.2.5

McCreadie, R., Macdonald, C. & Ounis, I. (2012). Identifying top news using crowdsourcing. *Information Retrieval* pp. 1–31. 1.5

McCreadie, R., Macdonald, C., Ounis, I., Peng, J. & Santos, R. (2009). University of Glasgow at TREC 2009: Experiments with Terrier. *In* ‘Proceedings of the 18th Text REtrieval Conference’. 8.2.1

McCreadie, R., Soboroff, I., Lin, J., Macdonald, C., Ounis, I. & McCullough, D. (2012). On Building a Reusable Twitter Corpus. *In* ‘Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval’. 3.4.3

Mejova, Y., Ha Turc, V., Foster, S., Harris, C., Arens, B. & Srinivasan, P. (2009). TREC Blog and TREC Chem: A View from the Corn Fields. *In* ‘Proceedings of the 18th Text REtrieval Conference’. 3.2.3

Metzler, D. & Cai, C. (2011). USC/ISI at TREC 2011: Microblog Track. 3.4.3

Metzler, D. A. (2007). Automatic feature selection in the markov random field model for information retrieval. *In* 'Proceedings of the 16th ACM Conference on Information and Knowledge Management'. 2.5.2, 8.2.1, 8.3.1

Mishne, G. & de Rijke, M. (2006). A study of blog search. *In* 'Proceedings of 28th European Conference on Information Retrieval'. 3.2.1, 3.2.4, 4.7.1

Munk, S. (2009). 'Michael Jackson death covered first by Twitter crashes servers'. `http://www.electricpig.co.uk/2009/06/26/michael-jackson-death-covered-first-by-twitter-crashes-servers/`, accessed on 30/09/2012. 1.2

Nagmoti, R., Teredesai, A. & De Cock, M. (2010). Ranking approaches for microblog search. *In* 'Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology'. 3.4.2, 4.7.2

Nam, S.-H., Na, S.-H., Lee, Y., & Lee, J.-H. (2009). Diffpost: Filtering non-relevant content based on content difference between two consecutive blog posts. pp. 791–795. 2.2.6

Newspaper Association of America (2010). 'Newspaper Web sites attract more than 70 million visitors in June; over one-third of all Internet users visit newspaper Web sites'. `http://www.naa.org/News-and-Media/Press-Center/Archives/2009/Newspaper-websites-attract-more-than-70-million-visitors.aspx`, accessed on 30/09/2012. 4.5

Nigam, K., Mccallum, A. K., Thrun, S. & Mitchell, T. (2000). Text classification from labeled and unlabeled documents using em. *Machine Learning* **39**, 103–134. 4.6, 7.2

Norman, J. (2011). 'Google processes 1,000,000,000 search queries per day (march 5, 2011)'. `http://www.historyofinformation.com/index.php?id=3276`, accessed on 30/09/2012. 1.1, 3.5, 4.3

O'Brien, T. (2009). 'Twitter breaks news of plane crash in the Hudson'. `http://www.switched.com/2009/01/15/twitter-breaks-news-of-plane-crash-in-the-hudson/`, accessed on 30/09/2012. 1.1, 1.2

Olston, C., Reed, B., Srivastava, U., Kumar, R. & Tomkins, A. (2008). Pig Latin: A not-so-foreign language for data processing. *In* 'Proceedings of ACM SIGMOD 2008 International Conference on Management of Data (SIGMOD '08)'. 2.2.5

Oracle BEA (2008). ‘Capacity planning process’. `http://docs.oracle.com/cd/E13214_01/wli/docs102/capplanguide/process.html`, accessed on 30/09/2012. 2.2.5

O’Reilly, T. & Milstein, S. (2009). *The Twitter Book*. O’Reilly Media, Inc. 1.1

Osborne, M., Petrović, S., McCreadie, R., Macdonald, C. & Ounis, I. (2012). Bieber no more: First Story Detection using Twitter and Wikipedia. *In* ‘Proceedings of Workshop on Time-aware Information Access’. 3.6.1, 7.6.3

Ounis, I., Amati, G., Plachouras, V., He, B., Macdonald, C. & Lioma, C. (2006). Terrier: A High Performance and Scalable Information Retrieval Platform. *In* ‘Proceedings of 2nd International Workshop on Open Source Information Retrieval’. 2.2.4, 6.4.1, 6.4.2, 8.3.1, 8.3.2

Ounis, I., de Rijke, M., Macdonald, C., Mishne, G. & Soboroff, I. (2006). Overview of the TREC-2006 Blog Track. *In* ‘Proceedings of the 15th Text REtrieval Conference’. 3.2.2

Ounis, I., Macdonald, C. & Soboroff, I. (2010). Overview of the TREC 2010 Blog Track. *In* ‘Proceedings of the 19th Text REtrieval Conference’. 3.2.3, 5.2.1, 5.2.3, 5.4.1, 5.6.1, 6.2, 8.3.1, 10.2

Ounis, I., Macdonald, C. & Soboroff, I. (2011). Overview of the TREC-2011 Microblog Track. *In* ‘Proceedings of the 20th Text REtrieval Conference’. 2.3, 3.4.3, 5.1, 5.2.3, 5.2.3

Ozmutlu, S., Spink, A. & Ozmutlu, H. C. (2004). A day in the life of Web searching: An exploratory study. *Information Processing and Management* **40**(2), 319–345. 4.3, 5.2.2, 5.5.1

Page, L., Brin, S., Motwani, R. & Winograd, T. (1999). The PageRank Citation Ranking: Bringing Order to the Web.. Technical report. Stanford InfoLab. 2.5

Pak, A. & Paroubek, P. (2010). Twitter as a corpus for sentiment analysis and opinion mining. *In* ‘Proceedings of International Conference on Language Resources and Evaluation’. 3.4.1

Petkova, D. & Croft, W. (2006). Hierarchical language models for expert finding in enterprise corpora. *In* ‘Proceedings of the IEEE International Conference on Tools with Artificial Intelligence’. 2.7.1

Petkova, D. & Croft, W. (2007). Proximity-based document representation for named entity retrieval. *In* ‘Proceedings of the 16th ACM Conference on Information and Knowledge Management’. 2.7.1

Petrovic, S., Osborne, M. & Lavrenko, V. (2010). Streaming first story detection with application to Twitter. *In* ‘Proceedings of The 11th Annual Conference of the North American Chapter of the Association for Computational Linguistics’. 3.4.4, 3.6.1

PHP Group (2009). 'PHP: Hypertext preprocessor'. `http://www.php.net/`, accessed on 30/09/2012. 2.2.6

Pike, R., Dorward, S., Griesemer, R. & Quinlan, S. (2005). Interpreting the data: Parallel analysis with Sawzall. *Scientific Programming* **13**(4), 277–298. 2.2.5

Ponte, J. M. & Croft, W. B. (1998). A language modeling approach to Information Retrieval. *In* 'Proceedings of the 21st International ACM SIGIR Conference on Research and Development in Information Retrieval'. 2.3.4

Porter, M. F. (1980). An algorithm for suffix stripping. *Program* **14**(3), 130–137. 2.2.3

Powell, A. L. & French, J. C. (2003). Comparing the performance of collection selection algorithms. *ACM Transactions on Information Systems* **21**(4), 412–456. 2.8.1

Pu, H., lung Chuang, S. & Yang, C. (2002). Subject categorization of query terms for exploring Web users' search interests. *Journal of The American Society for Information Science and Technology* **53**, 617–630. 3.5.2

Qu, Y., Huang, C., Zhang, P. & Zhang, J. (2011). Microblogging after a major disaster in China: A case study of the 2010 Yushu earthquake. *In* 'Proceedings of the ACM Conference on Computer Supported Cooperative Work'. 3.4

Quinlan, J. (1993). *C4. 5: programs for machine learning*. Morgan Kaufmann. 2.5.1

Randolph, J. J. (2005). Free-marginal multirater kappa (multirater free): An alternative to Fleiss' fixed-marginal multirater kappa. *In* 'Proceedings of the Joensuu Learning and Instruction Symposium'. 5.3.2

Rickns, M. (2010). 'Twitter to start pushing advertising to users'. `http://www.pcworld.com/businesscenter/article/194092/twitter_to_start_pushing_advertising_to_users.html`, accessed on 30/09/2012. 4.7.2

Rijsbergen, C. J. V. (1979). *Information Retrieval*. Butterworth-Heinemann. Newton, MA, USA. 2.4.1.1

Ritter, A., Clark, S., Etzioni, O. *et al.* (2011). Named entity recognition in tweets: An experimental study. *In* 'Proceedings of the Conference on Empirical Methods in Natural Language Processing'. 3.4.1

Robertson, S. & Walker, S. (1994). Some simple effective approximations to the 2-Poisson model for probabilistic weighted retrieval. *In* 'Proceedings of the 17th International ACM SIGIR Conference on Research and Development in Information Retrieval'. 2.8.1

Robertson, S. E. (1977). The probability ranking principle in IR. *Journal of Documentation* **4**(33), 294–304. 2.3

Robertson, S. E., van Rijsbergen, C. J. & Porter, M. F. (1981). Probabilistic models of indexing and searching. *In* 'Proceedings of the 3rd International ACM SIGIR Conference on Research and Development in Information Retrieval'. 2.3.2, 2.3.2

Robertson, S. E., Walker, S., Hancock-Beaulieu, M., Gull, A. & Payne, L. (1992). Okapi at TREC-1. *In* 'Proceedings of the 1st Text REtrieval Conference'. 6.4.1

Robertson, S. E., Walker, S., Jones, S., Hancock-Beaulieu, M. & Gatford, M. (1994). Okapi at TREC-3. *In* 'Proceedings of the 3rd Text REtrieval Conference'. 2.3.2, 2.7.1, 6.4.1, C.3

Rose, D. E. & Levinson, D. (2004). Understanding user goals in Web search. *In* 'Proceedings of the 13th International World Wide Web Conference'. 3.5.2

Rosenfeld, B., Feldman, R. & Ungar, L. (2008). Using sequence classification for filtering Web pages. *In* 'Proceeding of the 17th ACM Conference on Information and Knowledge Management'. 2.2.6

Sakaki, T., Okazaki, M. & Matsuo, Y. (2010). Earthquake shakes Twitter users: real-time event detection by social sensors. *In* 'Proceedings of the 19th International World Wide Web Conference'. 3.4.4

Salton, G. (1971). *The SMART Retrieval System—Experiments in Automatic Document Processing*. Prentice-Hall, Inc.. Upper Saddle River, NJ, USA. 2.3.1, 2.3.5, 2.3.5

Sanderson, M. & Dumais, S. (2007). Examining repetition in user search behavior. *In* 'Advances in Information Retrieval'. Vol. 4425 of *Lecture Notes in Computer Science*. Springer. pp. 597–604. 10.2

Santos, R. L., McCreadie, R., Macdonald, C. & Ounis, I. (2010). University of Glasgow at TREC 2010: Experiments with Terrier in Blog and Web tracks. *In* 'Proceedings of the 19th Text REtrieval Conference'. 1.5, 9.5.2

Santos, R. L. T., Macdonald, C. & Ounis, I. (2010*a*). Exploiting query reformulations for Web search result diversification. *In* 'Proceedings of the 19th International World Wide Web Conference'. 10.2

Santos, R. L. T., Macdonald, C. & Ounis, I. (2010*b*). Voting for related entities. *In* 'Proceedings of the 9th International Conference on Computer-Assisted Information Retrieval'. 6.4.1, C.1

Santos, R., Macdonald, C., McCreadie, R., Ounis, I. & Soboroff, I. (2012). *Information Retrieval on the Blogosphere*. Vol. 6. Foundations and Trends in Information Retrieval. 3.2, 10.2

Savoy, J. (2007). Why do successful search systems fail for some topics. *In* 'Proceedings of the ACM symposium on Applied computing'. 4.7

Schmid, H. (1995). Treetagger: A language independent part-of-speech tagger. *Institut für Maschinelle Sprachverarbeitung, Universität Stuttgart* p. 43. 6.4.1

Scott, D. (2007). *The new rules of marketing and PR: how to use news releases, blogs, podcasting, viral marketing, & online media to reach buyers directly*. John Wiley & Sons, Inc. New York, NY, USA. 1.1

Sear, J. (2012). 'If you agree with our editorial line, we'll pretend you're representative of "the public"'. `http://blogs.crikey.com.au/purepoison/2012/01/13/if-you-agree-with-our-editorial-line-well-pretend-youre-representative-of-the-public/`, accessed on 30/09/2012. 4.5, 6.1

Shamma, D. A., Kennedy, L. & Churchill, E. F. (2009). Tweet the debates: understanding community annotation of uncollected sources. *In* 'Proceedings of the 1st SIGMM workshop on Social media'. 1.2

Shaw, J. A. & Fox, E. A. (1994). Combination of Multiple Searches. *In* 'Proceedings of the 3rd Text REtrieval Conference'. 3.2.4

Shen, D., Pan, R., Sun, J., Pan, J., Wu, K., Yin, J. & Yang, Q. (2005). Q 2 C@ UST: our winning solution to query classification in KDDCUP 2005. *ACM SIGKDD 2005 Explorations Newsletter* **7**(2), 110. 2.8.2

Shokouhi, M. (2007). Central-rank-based collection selection in uncooperative distributed information retrieval. *In* 'Proceedings of 29th European Conference on Information Retrieval'. 2.8.1, 2.8.1, 2.8.1

Shokouhi, M. (2011). Detecting seasonal queries by time-series analysis. *In* 'Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval'. 10.2

Shokouhi, M. & Zobel, J. (2009). Robust result merging using sample-based score estimates. *ACM Transactions on Information Systems* **27**(3), 14. 2.8.3

Si, L. & Callan, J. (2003*a*). Relevant document distribution estimation method for resource selection. *In* 'Proceedings of the 26th International ACM SIGIR Conference on Research and Development in Information Retrieval'. 2.1

Si, L. & Callan, J. (2003*b*). A semisupervised learning method to merge search engine results. *ACM Transactions on Information Systems* **21**(4), 457–491. 2.8.3

Si, L. & Callan, J. P. (2003*c*). Relevant document distribution estimation method for resource selection. *In* 'Proceedings of the 26th International ACM SIGIR Conference on Research and Development in Information Retrieval'. 2.8, 2.8.1, 2.8.2

Silverstein, C., Marais, H., Henzinger, M. & Moricz, M. (1999). Analysis of a very large Web search engine query log. *In* 'ACM SIGIR Forum'. Vol. 33. ACM. pp. 6–12. 2.3.5, 3.5, 3.5.1

Singhal, A. (2010). 'Relevance meets the real-time Web'. `http://googleblog.blogspot.com/2009/12/relevance-meets-real-time-web.html`, accessed on 30/09/2012. 4.7.2

Singhal, A., Buckley, C. & Mitra, M. (1996). Pivoted document length normalization. *In* 'Proceedings of the 19th International ACM SIGIR Conference on Research and Development in Information Retrieval'. 2.3.1

Sirhan, H. (2011). '1,200 tweets per second (and other interesting Twitter stats)'. `http://www.freshnetworks.com/blog/2011/03/twitter-numbers-and-statistics/`, accessed on 30/09/2012. 7.7

Snow, R., O'Connor, B., Jurafsky, D. & Ng, A. Y. (2008). Cheap and fast—but is it good?: evaluating non-expert annotations for natural language tasks. *In* 'Proceedings of the Conference on Empirical Methods in Natural Language Processing'. 5.3.1, 5.4.3

Soboroff, I., McCullough, D., Lin, J., Macdonald, C., Ounis, I. & McCreadie, R. (2012). Evaluating Real-Time Search over Tweets. *In* 'Proceedings of the 6th International AAAI Conference on Weblogs and Social Media'. 3.4, 3.4.3, 5.2.3

Spärck-Jones, K. (1972). A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation* **28**, 11–21. 2.3.1

Sparck-Jones, K. & van Rijsbergen, C. J. (1975). Report on the need for and provision of an "ideal" judgements retrieval test collection.. Technical report. University of Cambridge. 2.4.2, 2.4.3

Spink, A., Jansen, B. J., Wolfram, D. & Saracevic, T. (2002). From E-Sex to E-Commerce: Web Search Changes. *IEEE Computer* **35**, 107–109. 3.5.2

Spink, A., Ozmutlu, S., Ozmutlu, H. C. & Jansen, B. J. (2002). U.S. Versus European Web Searching Trends. *ACM Sigir Forum* **36**, 32–38. 3.5

Spink, A., Wolfram, D., Jansen, M. B. J. & Saracevic, T. (2001). Searching the web: The public and their queries. *Journal of The American Society for Information Science and Technology* **52**, 226–234. 3.5.2

Subasic, I. & Castillo, C. (2010). The effects of query bursts on Web search. *In* 'Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology'. 3.5.1

Suh, B., Hong, L., Pirolli, P. & Chi, E. (2010). Want to be Retweeted? Large Scale Analytics on Factors Impacting Retweet in Twitter Network. *In* 'Proceedings of the IEEE Second International Conference on Social Computing'. C.6

Sussman, M. (2009). 'The state of the Blogosphere 2009'. `http://technorati.com/blogging/article/state-of-the-blogosphere-2009-introduction/`, accessed on 30/09/2012. 3.2.1, 4.7.1

Szab, G. & Huberman, B. A. (2008). Predicting the popularity of online content. *Communications of the ACM* **53**, 80–88. 3.3.2

Technorati (2011). 'State of the Blogosphere 2011: Introduction and Methodology'. `http://technorati.com/social-media/article/state-of-the-blogosphere-2011-introduction/`, accessed on 30/09/2012. 7.7

Teevan, J., Ramage, D. & Morris, M. (2011). # twittersearch: a comparison of microblog search and Web search. *In* 'Proceedings of the 4th International AAAI Conference on Weblogs and Social Media'. 3.4.2

Thelwall, M. (2006). Bloggers during the London attacks: Top information sources and topics. *In* 'Proceedings of the 3rd Annual Workshop on Weblogging Ecosystem: Aggregation, Analysis and Dynamics'. 3.2.1, 4.7.1

Tsagkias, M., de Rijke, M. & Weerkamp, W. (2011). Linking online news and social media. *In* 'Proceedings of the 4th ACM International Conference on Web Search and Data Mining'. 3.2.4

Tumasjan, A., Sprenger, T., Sandner, P. & Welpe, I. (2010). Predicting elections with twitter: What 140 characters reveal about political sentiment. *In* 'Proceedings of the 4th International AAAI Conference on Weblogs and Social Media'. 3.4.1

Turtle, H. R. & Croft, W. B. (1989). Inference networks for document retrieval. *In* 'Proceedings of the 13th International ACM SIGIR Conference on Research and Development in Information Retrieval'. 2.8.1

van Rijsbergen, C. (1979). *Information Retrieval, 2nd edition*. Butterworths, London. 2.1, 2.2.4, 2.3

Vapnik, V. (2000). *The nature of statistical learning theory*. Springer Verlag. 2.5.1

Vlachos, M., Meek, C., Vagena, Z. & Gunopulos, D. (2004). Identifying similarities, periodicities and bursts for online search queries. *In* 'Proceedings of ACM SIGMOD 2004 International Conference on Management of Data (SIGMOD '04)'. 3.5.1, C.4, C.4

Voorhees, E. (2003). Common Evaluation Measures. *In* 'Proceedings of the 12th Text REtrieval Conference'. 2.4.1.2, 2.4.1.3

Voorhees, E., Harman, D., of Standards, N. I. & (US), T. (2005). *TREC: Experiment and evaluation in information retrieval*. MIT press USA. 2.3, 2.4.4, 4.7

W3C (2009). 'Document object model'. `http://www.w3.org/DOM/`, accessed on 30/09/2012. 2.2.6

Wavelength Media (2009). 'What makes a story newsworthy?'. `http://www.mediacollege.com/journalism/news/newsworthy.html`, accessed on 30/09/2012. 4.5, 6.1

Weerkamp, W. & Rijke, M. D. (2008). External Query Expansion in the Blogosphere. *In* 'Proceedings of the 17th Text REtrieval Conference'. 3.2.2

Weng, J. & Lee, B. (2011). Event detection in twitter. *In* 'Proceedings of the 5th International AAAI Conference on Weblogs and Social Media'. 3.4.4

Witten, I. H. & Frank, E. (2005). *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann. 7.3

Wolfram, D., Spink, A., Jansen, B. J. & Saracevic, T. (2001). Vox populi: The public searching of the web. *Journal of The American Society for Information Science and Technology* **52**, 1073–1074. 3.5.2

Wu, B. & Davison, B. (2005). Identifying link farm spam pages. *In* ‘Proceedings of the 14th International World Wide Web Conference’. 8.5.3

Wu, Q., Burges, C. J. C., Svore, K. M. & Gao, J. (2008). Ranking, Boosting, and Model Adaptation. Technical report. Microsoft Research. 3.2.4

Wu, Q., Burges, C., Svore, K. & Gao, J. (2010). Adapting boosting for Information Retrieval measures. *Information Retrieval* **13**, 254–270. 2.5.2

Xu, J. & Croft, W. B. (1999). Cluster-based language models for distributed retrieval. *In* ‘Proceedings of the 22nd International ACM SIGIR Conference on Research and Development in Information Retrieval’. 2.8.1

Xu, J. & Croft, W. B. (2000). Improving the effectiveness of Information Retrieval with local context analysis. *ACM Transactions on Information Systems* **18**(1), 79–112. 2.6.1

Xu, X., Liu, Y., Xu, H., Yu, X., Peng, Z., Cheng, X., Xiao, L. & Nie, S. (2010). ICTNET at Blog track TREC 2010. *In* ‘Proceedings of the 19th Text REtrieval Conference’. 3.2.3, 6.3, 6.4.1, 6.6.4

Xu, Y., Jones, G. J. & Wang, B. (2009). Query dependent pseudo-relevance feedback based on wikipedia. *In* ‘Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval’. 4.7

Yang, Y. (2004). ‘CMU TEAM-A in TDT 2004 Topic Tracking’. `http://www.itl.nist.gov/iad/mig/tests/tdt/2004/papers/CMU_A_tracking_TDT2004.ppt`, accessed on 30/09/2012. 2.6.1

Yang, Y., Ault, T., Pierce, T. & Lattimer, C. W. (2000). Improving text categorization methods for event tracking. *In* ‘Proceedings of the 23rd International ACM SIGIR Conference on Research and Development in Information Retrieval’. 2.6.1

Yang, Y., Carbonell, J. G., Brown, R. D., Pierce, T., Archibald, B. T. & Liu, X. (1999). Learning approaches for detecting and tracking news events. *IEEE Intelligent Systems* **14**(4), 32–43. 2.6.1

Yano, T., Cohen, W. W. & Smith, N. A. (2009). Predicting response to political blog posts with topic models. *In* ‘Proceedings of The 10th Annual Conference of the North American Chapter of the Association for Computational Linguistics’. 10.2

Yu, H., Han, J. & Chang, K. C.-C. (2002). PEBL: Positive example based learning for Web page classification using SVM. *In* 'Proceedings of ACM SIGKDD 2002 International Conference on Knowledge Discovery and Data Mining'. 2.5, 2.5.1

Yuheng, H., Ajita, J., Duncan, S. D. & Fei, W. (2012). What were the Tweets about? Topical Associations between Public Events and Twitter Feeds. *In* 'Proceedings of the 6th International AAAI Conference on Weblogs and Social Media'. 3.4.4

Yuwono, B. & Lee, D. (1997). Server ranking for distributed text retrieval systems on the internet. *In* 'Proceedings of the 5th International Conference on Database Systems for Advanced Applications'. 2.8.1

Zeng, H.-J., He, Q.-C., Chen, Z., Ma, W.-Y. & Ma, J. (2004). Learning to cluster Web search results. *In* 'Proceedings of the 27th International ACM SIGIR Conference on Research and Development in Information Retrieval'. 2.5

Zhai, C. & Lafferty, J. (2004). A study of smoothing methods for language models applied to information retrieval. *ACM Transactions on Information Systems* **22**(2), 179–214. 2.3.4, C.3

Zhu, X. & Gauch, S. (2000). Incorporating quality metrics in centralized/distributed Information Retrieval on the World Wide Web. *In* 'Proceedings of the 23rd International ACM SIGIR Conference on Research and Development in Information Retrieval'. 3.2.2

Zobel, J. & Moffat, A. (2006). Inverted files for text search engines. *ACM Computing Surveys* **38**(2), 6. 2.2.4