On the Usefulness of Query Features for Learning to Rank

Craig Macdonald
School of Computing Science
University of Glasgow
Glasgow, G12 8QQ, UK
craig.macdonald@gla.ac.uk

Rodrygo L.T. Santos
School of Computing Science
University of Glasgow
Glasgow, G12 8QQ, UK
rodrygo@dcs.gla.ac.uk

Iadh Ounis
School of Computing Science
University of Glasgow
Glasgow, G12 8QQ, UK
iadh.ounis@gla.ac.uk

ABSTRACT

Learning to rank studies have mostly focused on query-dependent and query-independent document features, which enable the learning of ranking models of increased effectiveness. Modern learning to rank techniques based on regression trees can support *query features*, which are document-independent, and hence have the same values for all documents being ranked for a query. In doing so, such techniques are able to learn sub-trees that are specific to certain types of query. However, it is unclear which classes of features are useful for learning to rank, as previous studies leveraged anonymised features. In this work, we examine the usefulness of four classes of query features, based on topic classification, the history of the query in a query log, the predicted performance of the query, and the presence of concepts such as persons and organisations in the query. Through experiments on the ClueWeb09 collection, our results using a state-of-the-art learning to rank technique based on regression trees show that all four classes of query features can significantly improve upon an effective learned model that does not use any query feature.

Categories & Subject Descriptors: H.3.3 [Information Storage & Retrieval]: Information Search & Retrieval

Keywords: Learning to Rank, Query Features

1. INTRODUCTION

Learning to rank deployments leverage various query-dependent (e.g. term weighting models, proximity) and query-independent (e.g. URL length, PageRank, content quality) features, with which to re-rank in an effective manner a *sample* of documents obtained from a single feature (usually BM25 [12]). Such features are combined into a *learned model*, obtained by minimising the loss function of a learning to rank technique (see [12] for an overview).

While learning to rank techniques are often differentiated as pointwise, pairwise, and listwise, depending on their loss function [12], we focus on a different viewpoint. In particular, we separate techniques that learn a linear combination of features (possibly after a kernel transformation) from techniques based on regression trees, such as gradient boosted regression trees [21] and LambdaMART [23]. For such *tree-based learners*, the partial score for a given document is obtained by following a traversal of a regression tree. At each node, a particular feature value is compared with a learned threshold, to define a path towards the partial score for that document. Typically, an *ensemble* consisting of multiple trees is used to obtain the final score for each document [7].

In addition to the classical query-dependent and query-independent document features, a third class of features has seen comparatively less research in the learning to rank literature. In particular, a *query feature* is a quantifier for some aspect of the query, and – in contrast to document features – has the same value across all documents in the sample. Query features can be used by tree-based learners, serving as decision points between the various branches of a learned ranking model, thereby permitting the learner to 'customise' the learned model with respect to an aspect of the query [21]. An example query feature might be the number of terms in the query [21], which may trigger branches of the learned model's trees that are better suited for long queries. Chapelle et al. [7] mention the 'adult score' of the top retrieved documents as another example query feature.

Research encapsulating query features within learning to rank is in its infancy. For instance, none of the standard LETOR datasets for learning to rank [12] deploy query features. This problem was addressed by the Yahoo! Learning to Rank Challenge, whose dataset contains 36 query features [7]. However, to the best of our knowledge, no previous work has analysed the usefulness of different types of query features. As a matter of fact, the features included in the Yahoo! Learning to Rank Challenge are anonymised, which precludes any analysis on the effectiveness of individual query features. On the other hand, query features have been used for customising rankings on a per-query basis. For instance, Geng et al. [11] used query features to cluster similar queries before learning common ranking models. Santos et al. [20] used query features to learn the appropriate amount of diversification to apply for a query. However, in all of these works, query features were used to improve the understanding of the query *before* the ranking stage, as opposed to being leveraged as an integral part of the learning to rank process.

To further the understanding of the usefulness of query features for learning to rank, we conduct the first empirical analysis of query features in a state-of-the-art learning to rank deployment. In the remainder of this paper, Section 2 describes the experimental methodology underlying our investigation. Our findings are discussed in Section 3, and Section 4 presents our final remarks.

Features	Type	Total
Weighting models (DPH [2], PL2 [1], BM25 [18], LM, MQT [20])	QD	21
Fields-based models (PL2F [13])	QD	1
URL and link analysis features (e.g. PageRank, Absorbing Model [17], EdgeReciprocity)	QI	13
Quality features (e.g., fraction of stopwords, table text [3])	QI	8
Click feature (click count)	QI	1
Spam feature (Cormack's fusion score [6])	QI	1
Term-dependence models (MRF [14], pBiL [16])	QD	2
TOTAL		47

Table 1: All query-dependent (QD) and query-independent (QI) document features used in this work.

Source	Class	Feature	Description	Total
query	QPP	AvICTF [4]	Pre-retrieval performance predictor	1
query	QPP	AvIDF [4]	Pre-retrieval performance predictor	1
query	QPP	AvPMI [4]	Pre-retrieval performance predictor	1
query	QPP	EnIDF [4]	Pre-retrieval performance predictor	1
query	QPP	Gamma1 [4]	Pre-retrieval performance predictor	1
query	QPP	Gamma2 [4]	Pre-retrieval performance predictor	1
query	QPP	TermCount	Number of unique terms	1
query	QPP	TokenCount	Number of tokens	1
query	QPP	N-GramScore	Likelihood of ngram query in anchor or title fields	8
query	QCI	AcronymSenses	Number of acronym senses	1
documents	QCI	WPDisambSenses [19]	Number of disambiguation senses per document	18
documents	QCI	WPDisambCount	Number of disambiguation pages retrieved	6
query	QCI	EntityCount	Number of named entities in the query	4
query	QLM	NGramScore	Likelihood of ngram query in query log	3
clicks	QLM	ClickCount	Number of clicks	3
clicks	QLM	ClickEntropy [5]	Click entropy at the URL level	1
clicks	QLM	HostEntropy [22]	Click entropy at the host level	1
clicks	QLM	ResultCount	Number of displayed results in a session	3
clicks	QLM	SessionDuration	Session duration in seconds	3
documents	QTC	DocEntityCount	Number of retrieved entities (products, persons, organisation, locations)	18
documents	QTC	DocEntityEntropy	Entity entropy of centroid document	18
documents	QTC	DocEntityPairwiseCosine	Entity distance over pairs of top documents	54
documents	QTC	WPCategoryCount	Number of retrieved categories	6
documents	QTC	WPCategoryEntropy	Category entropy of centroid document	6
documents	QTC	WPPairwiseCosine	Categorical distance over pairs of top documents	18
TOTAL				178

Table 2: All query features used in this work. Unreferenced query features are described by Santos et al. [20].

2. EXPERIMENTAL METHODOLOGY

In the following experiments, our goal is to assess the effectiveness of a series of query features when added to an effective machine learned ranking model that does not employ any query features. Our experiments use the ClueWeb09 (category B) collection, which comprises 50 million English Web documents, and is aimed to represent the first tier index of a commercial search engine. We index this collection using the Terrier information retrieval platform [15],[1] with a weak Porter stemmer. To form a sizable query set for our investigation, we use 98 topics and relevance assessments from the TREC 2009-2010 Web tracks, and 140 from the TREC 2009 Million Query track. For each query, we use the Divergence from Randomness DPH weighting model [2] to produce an initial sample of 5000 documents. For documents in the sample, we calculate a total of 47 standard query-dependent (e.g. term weighting models, proximity features) and query-independent document features (e.g. link analysis, URL length, content quality), similar to those deployed within the LETOR datasets [12]. Table 1 lists the 47 query-dependent (QD) and query-independent (QI) document features used within our experiments.

To re-rank the documents in the sample, we use the LambdaMART learning to rank technique [23] (as implemented by the Jforests package [9][2]), a representative of current state-of-the-art learning to rank techniques, as per its top-class performance in the 2011 Yahoo! Learning to Rank Challenge [7]. A ranking model learned by LambdaMART based on a loss function targeting NDCG, and using the aforementioned 47 document features, forms the baseline for the investigation in this paper. In particular, we investigate whether query features can enhance this baseline, and analyse the usefulness of different classes of query features.

Following Santos et al. [20], who investigated query features for selective diversification, in addition to the 47 document features, we calculate 178 query features, which are classified according to the tasks that inspired them: query concept identification (QCI); query performance prediction (QPP); query log mining (QLM); and query topic classification (QTC). More generally, these features examine the statistics of the query within the corpus, how many times it appears within the MSN 2006 query log,[3] or how entities (persons, products, organisations or locations) occur in the query or the corresponding top-ranked sample documents. Table 2 details the used query features for each class. This table also lists the main evidence source for each feature, namely the query itself, user click behaviour in the query log

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

[2]`http://code.google.com/p/jforests/`

[3]`http://tiny.cc/wscd09`

for that query, or a number of top-ranked sample documents for the query. In particular, query features that examine the sample documents are instantiated multiple times, examining different numbers of top-ranked sample documents. For example, a query feature named DocEntityCount-org-50 considers the mean number of organisation entities identified in the top 50 retrieved sample documents. The distribution of entities across the top retrieved documents results in further summary features, such as average, maximum, and standard deviation.

Finally, our experiments are conducted using a 5-fold cross validation across the 238 TREC topics (98 Web track + 140 Million Query track topics), where each fold has separate training, validation and test query sets. Our experimental results report NDCG@20 across the test sets from each fold.

3. EXPERIMENTAL RESULTS

For each query feature, we add it to the 47 document features of the baseline ranking, and learn a ranking model for each fold. Across the 5 folds, we found that all but 3 query features exhibited improvements over the LambdaMART baseline that has no query features. Of the improving features, 46 brought significant improvements in NDCG@20 (paired t-test, $p < 0.05$). Table 3 shows the effectiveness of the four best features for each class (QCI, QPP, QLM and QTC) of query features, as well as the mean NDCG@20 of all query features in each class, and the number of query features exhibiting significant improvements compared to the baseline ($p < 0.05$). On analysing the table, we observe that the most useful query features result in statistically significant increases over the baseline that does not deploy any query features.

In particular, the best query topic classification (QTC) feature results in an 8% increase in NDCG@20 ($0.2832 \rightarrow 0.3109$). This query feature examines the number of organisations mentioned in the top 100 sample documents [20], suggesting that QTC query features are useful for triggering sub-trees that promote entity homepages.

Of the other query feature classes, query performance predictors (QPP) show promise in increasing effectiveness, with the Gamma1 pre-retrieval predictor [20] being the most effective. By deploying such a query performance predictor as a query feature, the learner is able to customise different tree branches for easy and difficult queries. The second most useful query performance predictor is simply the query length, in tokens. This feature likely helps the learner to decide the importance of proximity document weighting features (e.g. MRF [14] or pBiL [16]). The last two features concern the importance of 1-grams and 4-grams within an anchor text index, and hence are likely to differentiate between navigational and informational queries, resulting in different learned models for queries of different intents.

For query concept identification (QCI), the deployed query features include the number of acronyms and the number of entities (locations, organisations, products, and persons) in the query. Three of the four highest performing features consider the ambiguity of the query: for instance, the highest-performing query feature counts the number of disambiguation pages ranked in the top 3 Wikipedia pages for the query. The fourth feature denotes the presence of known person entities in the query, again suggesting that queries for entities require adapted ranking models.

Of our query log mining (QLM) features, three of the four most useful query features considered the complexity of the user sessions for such a query. For instance, longer mean user session duration for a query may indicate a more difficult query, with a similar interpretation if more results are viewed. Moreover, a query whose sessions exhibit a high variance in durations may be ambiguous. Hence, both these query features and the QCI Wikipedia disambiguation query features suggest that different sub-trees for queries with different levels of ambiguity can be learned. Finally, the 1-gram score for the query is an indication of query popularity, suggesting that more popular (head) queries can have customised ranking models – indeed, head queries are likely to be more navigational in nature.

Comparing the mean performance across all features in a class (see Mean rows in Table 3), we find that all classes of query features exhibit similar improvements above the baseline, suggesting that each could bring further evidence to the LambdaMART learning to rank technique. However, the QTC and QLM classes exhibited a higher fraction of features leading to significant effectiveness improvements. Overall, the results in Table 3 provide concrete examples of several types of features that can be successfully integrated with document features to result in learned models with significantly improved effectiveness.

Finally, we combine the best feature from each class with the 47 document features. The performance of the combined learned model with 47+4 features is shown in Table 3. As the performance does not improve over the 47-feature baseline, we conclude that LambdaMART can easily overfit for multiple query features. This is explainable in that the learner has far less queries than documents (144 training queries vs 720,000 training documents). While more queries should prevent this overfitting, feature selection approaches (e.g. adapting [10] to query features) may also be beneficial - we leave this to future work.

4. CONCLUSIONS

While query features are mentioned in the recent learning to rank literature, there has been no empirical investigation into *which* types of query feature are useful to improving learned models. In this paper, we investigated the benefit of several classes of query features, when integrated with document features using the state-of-the-art LambdaMART learning to rank technique. We found that almost all query features could improve performance, with over a quarter of the 178 features exhibiting significant improvements. In particular, our results show that query features can be successfully employed to customise ranking models for queries with different popularity, length, difficulty, ambiguity, and related entities. Future work will examine other possible classes of query features, and ways in which multiple query features can be combined.

5. REFERENCES

[1] G. Amati. *Probability models for information retrieval based on Divergence From Randomness.* PhD thesis, Univ. of Glasgow, 2003.

[2] G. Amati, E. Ambrosi, M. Bianchi, C. Gaibisso, and G. Gambosi. FUB, IASI-CNR and Univ. of Tor Vergata at TREC 2007 Blog track. In *TREC*, 2007.

[3] M. Bendersky, W. B. Croft, and Y. Diao. Quality-biased ranking of web documents. In *WSDM*, 2011.

[4] D. Carmel and E. Yom-Tov. Estimating the query difficulty for information retrieval. *Synthesis Lectures*

Class	Feature	NDCG@20
Baseline (no query features)		0.2832
QPP		
Rank 1	Gamma1	0.3033$^\blacktriangle$
Rank 2	TermCount	0.30217$^\triangle$
Rank 3	1-GramScore on anchor field	0.29857
Rank 4	4-GramScore on anchor field	0.29727
Mean	-	0.2955 (2/16)
QTC		
Rank 1	DocEntityCount-org-100	**0.3109**$^\blacktriangle$
Rank 2	DocEntityPairwiseCosine-std-products-10	0.30635 $^\blacktriangle$
Rank 3	WPTopPairwiseCosine-PL2-avg-50	0.30612 $^\blacktriangle$
Rank 4	WPTopPairwiseCosine-PL2-max-3	0.30605 $^\triangle$
Mean	-	0.2950 (36/120)
QCI		
Rank 1	WPDisambCount-3	0.3049$^\blacktriangle$
Rank 2	WPDisambSenses-avg-50	0.30282$^\triangle$
Rank 3	WPDisambSenses-std-50	0.30196$^\triangle$
Rank 4	PersonCount	0.29688
Mean	-	0.2917 (3/29)
QLM		
Rank 1	SessionDuration-std	0.3085$^\blacktriangle$
Rank 2	SessionDuration-avg	0.30647$^\triangle$
Rank 3	ResultCount-med	0.30520$^\blacktriangle$
Rank 4	NGramScore-query-1	0.30364$^\triangle$
Mean	-	0.2964 (5/14)
Top query feature from each class combined		0.2809

Table 3: NDCG@20 of the four best ranked for each class of query features, along with the mean performance of all features of each class. The symbol $\triangle$ ($\blacktriangle$) denotes a significant improvement according to the paired *t*-test for $p < 0.05$ ($p < 0.01$), with respect to the baseline that uses no query features. Fractions in parenthesis for each class denote the number of query features with significant improvement over the baseline.

on Information Concepts, Retrieval, and Services, 2(1):1–89, 2010.

[5] P. Clough, M. Sanderson, M. Abouammoh, S. Navarro, and M. Paramita. Multiple approaches to analysing query diversity. In *SIGIR*, 2009.

[6] G. V. Cormack, M. D. Smucker, and C. L. A. Clarke. Efficient and effective spam filtering and re-ranking for large Web datasets. *Inf. Retr.*, 14(5):441–465, 2011.

[7] O. Chapelle, Y. Chang. Yahoo! learning to rank challenge overview. *J. Machine Learning: Workshop and Conference Proceedings*, 14:1–24, 2011.

[8] S. Cronen-Townsend, Y. Zhou, and W. B. Croft. Predicting query performance. In *SIGIR*, 2002.

[9] Y. Ganjisaffar, R. Caruana, and C. Lopes. Bagging gradient-boosted trees for high precision, low variance ranking models. In *SIGIR 2011.*

[10] X.-B. Geng, T.-Y. Liu, T. Qin, H. Li. Feature selection for ranking. In *SIGIR 2007.*

[11] X.-B. Geng, T.-Y. Liu, T. Qin, A. Arnold, H. Li, H.-Y. Shum. Query dependent ranking using k-nearest neighbor. In *SIGIR 2008.*

[12] T.-Y. Liu. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, 3(3):225–331, 2009.

[13] C. Macdonald, V. Plachouras, B. He, C. Lioma, and I. Ounis. Univ. of Glasgow at WebCLEF 2005: Experiments in per-field normalisation and language specific stemming. In *CLEF*, 2005.

[14] D. Metzler and W. B. Croft. A Markov random field model for term dependencies. In *SIGIR*, 2005.

[15] I. Ounis, G. Amati, V. Plachouras, B. He, C. Macdonald, and C. Lioma. Terrier: A high performance and scalable information retrieval platform. In *OSIR at SIGIR*, 2006.

[16] J. Peng, C. Macdonald, B. He, V. Plachouras, and I. Ounis. Incorporating term dependency in the DFR framework. In *SIGIR*, 2007.

[17] V. Plachouras, I. Ounis, and G. Amati. The static absorbing model for the Web. *J. Web Eng.*, 4(2):165–186, 2005.

[18] S. E. Robertson, S. Walker, S. Jones, M. Hancock-Beaulieu, and M. Gatford. Okapi at TREC-3. In *TREC*, 1994.

[19] M. Sanderson. Ambiguous queries: Test collections need more sense. In *SIGIR*, 2008.

[20] R. L. T. Santos, C. Macdonald, and I. Ounis. Selectively diversifying Web search results. In *CIKM 2010.*

[21] S. Tyree, K. Weinberger, K. Agrawal, and J. Paykin. Parallel boosted regression trees for Web search ranking. In *WWW 2011.*

[22] Y. Wang and E. Agichtein. Query ambiguity revisited: clickthrough measures for distinguishing informational and ambiguous queries. In *NAACL-HLT 2010*, 2010.

[23] Q. Wu, C. J. C. Burges, K. M. Svore, and J. Gao. Ranking, boosting, and model adaptation. Technical Report MSR-TR-2008-109, Microsoft, 2008.