From Puppy to Maturity: Experiences in Developing Terrier

Craig Macdonald, Richard McCreadie,
Rodrygo L.T. Santos, and Iadh Ounis
terrier@dcs.gla.ac.uk
School of Computing Science
University of Glasgow
G12 8QQ, Glasgow, UK

ABSTRACT

The Terrier information retrieval (IR) platform, maintained by the University of Glasgow, has been open sourced since 2004. Open source IR platforms are vital to the research community, as they provide state-of-the-art baselines and structures, thereby alleviating the need to 'reinvent the wheel'. Moreover, the open source nature of Terrier is critical, since it enables researchers to build their own unique research on top of it rather than treating it as a black box. In this position paper, we describe our experiences in developing the Terrier platform for the community. Furthermore, we discuss the vision for Terrier over the next few years and provide a roadmap for the future.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval

1. INTRODUCTION

The origins of the Terrier open source information retrieval (IR) platform[1] within the University of Glasgow can be traced back to 2001, when it was first created in Java to provide a common basis for research students to use for their PhD research. Since then, the platform has grown to be a scalable and mature open source platform released under the Mozilla Public License (MPL)[2], aimed at researchers and practitioners, permitting the rapid and effective research and development of information retrieval technologies.

Two of the key goals of Terrier are to be flexible and extensible, such that the platform can act as a corner-stone upon which both the academic community and practitioners can build. To this end, Terrier follows a modular design, whereby different components of the indexing and retrieval process can be customised. For instance, when working with Twitter, it can be advantageous to use a dedicated tokeniser that removes URLs and mentions from the text. By combining a modular design with an open source nature, Terrier can be adapted for a potentially unlimited number of use-cases.

However, it is also important to provide as much functionality out-of-the-box as possible. Indeed, Terrier strives to provide state-of-the-art efficient indexing and effective retrieval mechanisms. For example, due to its modularity, Terrier allows various ways of changing the ranking of documents, providing a huge variety of weighting models, including Okapi BM25 [17], language modelling [10, 23] and a vast number of models from the Divergence from Randomness framework [2]. It also includes field-based document weighting models for tackling more structured documents, such as BM25F [22] or PL2F [13]. As a result, Terrier is well known within IR evaluation forums such as TREC and FIRE as the basis of many effective systems.

Over the last decade, the Terrier platform has been developed and enhanced by a wide range of academics world-wide. Indeed, contributions made to Terrier have been driven by the desire to explore new research directions, and will remain so in the future. For instance, Terrier is currently being extended to tackle the new challenges of real-time search tasks, e.g. incremental indexing, live search and reproducibility of experimentation in such dynamic search scenarios. However, open source platforms require significant investments in time and manpower to develop and maintain. While such investments may not directly lead to research outcomes, they are of utmost importance for the research community. For this reason, in this paper, we detail our experiences in developing Terrier and provide a roadmap for future releases of the platform. In this way, the contributions of this position paper are two-fold: we describe the growth of the Terrier platform along the past decade and its latest developments, followed by a roadmap for the next generation of the open source search platform.

This paper is structured as follows. Section 2 discusses the philosophy that guides the development of Terrier. Section 3 describes recent developments of the platform, whereas Section 4 provides a roadmap for future developments. Finally, Section 5 provides our concluding remarks.

2. BUILDING FOR SUCCESS

Our overriding belief behind Terrier is that an information retrieval (IR) system 'should just work... out-of-the-box.' Many users will come to Terrier, and their first experience using the platform is key—we want to ensure that the crucial first experience facilitates the aim that they have for using the platform. In particular, we identify four dimensions that are key to the user experience, namely:

- Effectiveness: The platform should be effective by default. Moreover, it should provide easy access to the accepted state-of-the-art techniques, permitting experiments to be conducted with minimal development cost.

[1] `http://terrier.org`
[2] `http://www.mozilla.org/MPL/`

SIGIR 2012 Workshop on Open Source Information Retrieval.
ACM August 16, 2012, Portland, Oregon, USA.

- Efficiency: Scientific experiments in information retrieval have advantages over other scientific fields, in that experimental ground truths (i.e. relevance assessments) are generally reusable. While the efficiency of an IR research platform is not paramount, we believe that the system should be generally efficient so as to facilitate fast experiment iterations.
- Scalability: The size of IR test corpora has grown 500-fold since the first open source release of Terrier, as illustrated in Figure 1. Regardless of the scale of hardware resources available to a researcher, we believe that the IR system should be able to index and retrieve without challenging configuration.
- Adaptability: It must be possible to adapt the system to new requirements, whether this encompasses new retrieval strategies, new corpora, or different experimental paradigms.

These dimensions (which we collectively denote EESA) underlie the decisions behind the development of the Terrier platform, and the plans for its future. In particular, with Terrier, we aim to ensure that indexing and retrieval can be effectively performed by making efficient use of whatever resource is available, be it a single machine or a large cluster.

An important choice in Terrier has centred around the effectiveness/efficiency/adaptability tradeoff. In particular, IR researchers may not know a priori which particular weighting model they intend to use during retrieval time. Indeed, some retrieval approaches decide on the choice of weighting model on a per-query basis (e.g. [8]). For this reason, we do not believe that efficient retrieval approaches (e.g. score-at-a-time [3]) that tie the index to a particular retrieval approach are suitable for an experimental IR platform, even if they can produce marked efficiency improvements.

On the other hand, there are software engineering challenges in maintaining and improving a large platform such as Terrier. For instance, since Terrier 3.0 we have been following a testing regime that ensures that changes do not impact on correctness (unit tests) and effectiveness (end-to-end tests). In the following, we identify some specific improvements made to the Terrier over the last few years and how these are related to the EESA dimensions, before going on to identify a roadmap for further developments of Terrier.

3. RECENT IMPROVEMENTS

In this section, we highlight recent improvements in Terrier, covering both its indexing and retrieval architectures.

3.1 Indexing

Since the inception of Terrier, the ability to quickly produce compressed index structures representing collections of documents has been critical. However, over the last decade, the scale of the document collections of interest, the specifications of commodity hardware used for indexing and the search tasks that are being investigated have dramatically changed. For instance, Figure 1 illustrates how the size of IR test collections has grown over a 17 year period. These changes have and continue to introduce new indexing challenges that have driven the continual enhancement of Terrier's indexing processes.

The basis for much of the indexing functionality within Terrier stems from the development of single-pass indexing [9], which was released as part of Terrier 2.0. The idea behind single-pass indexing is that a central index structure can be built a document-at-a-time in a 'single-pass' over the collection. Moreover, this index can be created under tight memory constraints by compressing the inverted files and periodically writing partial posting-lists to disk, facilitating indexing on a single machine. Single-pass indexing is a fast and efficient means to index smaller (by today's standards) collections, spanning 100 million documents or less.

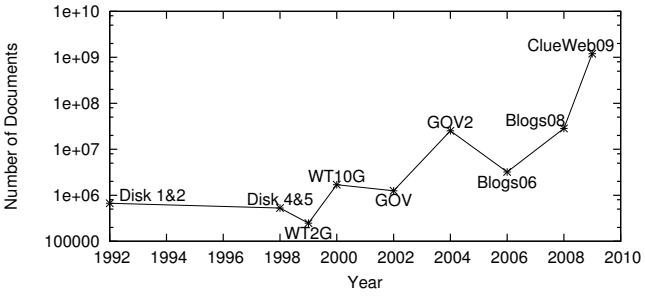

Figure 1: Evolution of the size of TREC corpora.

However, for larger collections such as the TREC ClueWeb-09 corpus that is comprised of 1.2 *billion* documents,[3] single-pass indexing is a slow process requiring weeks of processor time to complete on a single machine. MapReduce is a programming paradigm for the processing of large amounts of data by distributing work tasks over multiple processing machines [6]. The central concept underpinning MapReduce is that many data-intensive tasks are based around performing a *map* operation with a simple function over each 'record' in a large dataset. Terrier 2.2 became the first open source IR platform to implement large-scale parallelised indexing using MapReduce [14] and its Java implementation Hadoop.[4] This has enabled Terrier to scale its indexing process to efficiently tackle collections spanning billions of documents or more, given a suitable cluster of machines.

Furthermore, it is not just the scaling of existing IR processes that is of interest. To facilitate new search tasks, the indexing process needs to be flexible and extensible. Indeed, this is especially true as the focus of IR continues to move from traditional web documents to more specialised domains, such as the search of social media and other user-generated content sources [21]. To this end, the open source and modular nature of Terrier allows for the addition of indexing capability for new collections with specialised structured documents, in addition to custom tokenisation, stopword removal and stemming. For example, we released a custom package for Terrier 3.0 to support the processing of JSON tweets outside the normal release cycle.[5] Moreover, the entire indexing process can be extended to tackle entirely new search tasks. For instance, the ImageTerrier[6] project enhanced Terrier with image retrieval functionality.

Recent developments have focused on enhancing the deployment and demonstration capabilities of Terrier. From an indexing perspective, this involves efficiently storing document metadata for later display to the user. Terrier 3.5 supports automated metadata extraction and snippet gen-

[3] `http://boston.lti.cs.cmu.edu/Data/clueweb09`
[4] `http://hadoop.apache.org`
[5] `http://ir.dcs.gla.ac.uk/wiki/Terrier/Tweets11`
[6] `http://www.imageterrier.org`

Figure 2: Terrier user interface for Twitter search.

eration that can be saved within a specialist meta index structure. This functionality can be paired with the Terrier front-end search interface to facilitate custom search applications, such as Twitter search, as illustrated in Figure 2.

3.2 Retrieval

When retrieving from large corpora in constrained memory situations, it is possible that the system does not have sufficient memory available to decompress the entire posting list for common query terms. From Terrier 3.0, we redesigned the retrieval mechanism such that the posting list for each term is decompressed in a streaming fashion, using an iterator design pattern. Moreover, this had an additional benefit in permitting support for Document-at-a-Time retrieval (DAAT) strategies. Indeed, DAAT retrieval strategies are advantageous in that the number of document score accumulators is markedly reduced compared to Term-at-a-Time (TAAT) scoring, ensuring an efficient retrieval process.

On the internationalisation front, we have been working on extending Terrier to index and retrieve from East Asian corpora. In particular, in Terrier 3.5, we refactored the way documents are processed, such that text parsing and tokenisation are now fully separated operations. As a result of this refactoring, Terrier now supports pluggable tokenisers for different languages, adding to the overall adaptability of the platform. In a first test of the new tokenisation architecture, Terrier delivered out-of-the-box state-of-the-art retrieval performance on news and web corpora for both Chinese and Japanese [20].

4. ROADMAP FOR TERRIER

In the following, we highlight new functionalities developed for Terrier, which we plan to release in future open source versions. Each of these functionalities is key to improving one or more dimensions of EESA within the Terrier platform. In particular, the massive scale and heterogeneity of current corpora and the increasingly complex information needs of search users limits the effectiveness of traditional ranking approaches based on a single feature. Instead, effective retrieval is increasingly moving towards machine-learned ranking functions combining multiple features [11].

Feature-based retrieval (effectiveness and efficiency).

Terrier will support the extraction of query-independent features at indexing time, as well as the efficient extraction of query-dependent features with a single pass over the inverted index at retrieval time. The latter is enabled by an improved matching mechanism that keeps track of the actual postings for documents that might end up among the top retrieved. Another valuable source of evidence, which conveys how a document is described by the rest of the Web, is the anchor text of the incoming hyperlinks to this document. Integrating anchor-text extraction to the indexing pipeline of Terrier will provide a unified solution for leveraging this rich evidence as an additional feature for effective retrieval.

Non-global configuration (adaptability).

An important direction for improving the adaptability of Terrier is to enable multiple instances of its indexing and retrieval pipelines to run concurrently. A crucial development in this direction is a configuration system that admits non-global, instance-specific setups. For instance, a typical search scenario that may require multiple instances of a retrieval process is search result diversification. In particular, effective diversification can be attained by ensuring that the produced ranking covers multiple aspects of an ambiguous query, represented as query reformulations [18, 19].

Dynamic pruning (scalability and adaptability).

Dynamic pruning strategies, such as WAND [4], can increase efficiency by omitting the scoring of documents that can be guaranteed not to make the top-k retrieved set—a feature known as safeness. These pruning strategies rely on maintaining a threshold score that documents must overcome in order to be considered in the top-k documents. Each term is associated with an upper bound stating the maximal contribution of the weighting model to any document's relevance score. By comparing upper bounds on the scores of the terms that have not been scored to the threshold, i.e. the current k-th document, the pruning strategy decides on to whether to skip the scoring of documents during retrieval. In addition, in WAND, skipping is also supported by underlying posting list iterators in order to reduce disk IO and further increase efficiency. However, traditionally in the literature, the upper bounds are pre-calculated at indexing time and stored in the index. As a result, the generated index is only suited for efficient retrieval using one particular weighting model, where all the returned search results of queries are ranked using a single weighting model. Instead, in Terrier, we avoid tying up the generated index to a particular weighting model, by deploying safe upper bound approximations for various retrieval models, which can be calculated on-the-fly at query execution time [12]. By doing so, Terrier supports the recent trend of deploying selective retrieval approaches, where the retrieval strategy varies from a query to another [7, 19].

Distributed and real-time search (scalability).

While a MapReduce indexing process can efficiently index the 1.2B documents of the ClueWeb09 corpus, retrieval from a monolithic single index shard is not efficient. Document-partitioned distributed indexing [5] ensures that efficient retrieval can be attained regardless of the index size. Recently, search in real-time scenarios such as live search over tweets have become popular [21]. Real-time search poses different challenges to traditional retrospective retrieval tasks. In particular, documents are not contained within a static corpus, but rather arrive over time in a streaming fashion.

Moreover, both indexing and retrieval operations occur in parallel, hence index structures must always be searchable, thread safe and up-to-date. Furthermore, from a research perspective, there is a need to support the 'replaying' of a stream, facilitating reproducible experimentation. The development of real-time search within Terrier is well advanced and is targeted for merging into the next major open source release. Terrier's real-time infastructure is also being further developed as part of the SMART EC project to enable low latency distributed indexing and retrieval [1].[7]

Crowdsourcing for relevance assessment (effectiveness).

Researchers rely on document relevance assessments for queries to gauge the effectiveness of their systems. Evaluation forums such as TREC and CLEF play a key role by providing relevance assessments for many common tasks. However, these venues cannot cover all of the collections and tasks currently investigated in IR, resulting in the burden of relevance assessment generation falling upon individual researchers. This is an important problem, as relevance assessment generation is often a time-consuming, difficult and potentially costly process. For many IR-related tasks, crowdsourcing has been shown to be a fast and cheap method to generate relevance assessments in a semi-automatic manner [16], by outsourcing to a large group of non-expert workers. CrowdTerrier is a soon to be released open source extension to Terrier that leverages crowdsourcing to provide researchers with an out-of-the-box tool to achieve fast and cheap relevance assessment [15].

Plugin Expansions (adaptability).

The growth of the Terrier platform into exciting new areas such as crowdsourcing entails increased functionality, but also platform complexity. To avoid software bloat, we are moving from a monolithic release structure, to a system of periodic core releases and timely plugin expansions. This will enable Terrier to continue to grow and evolve to tackle future challenges in the dynamic field of IR in line with the interests of the community.

5. CONCLUSIONS

In this paper, we described the philosophy that has guided the development of the Terrier IR open source platform since its first release in 2004. We described some recent developments in the Terrier IR platform, as well as a comprehensive roadmap for its forthcoming releases, intended to ensure that the platform remains extensible and effective, while providing a robust, modern and efficient grounding for building next generation search engine technologies. The last decade has witnessed a dramatic shift in the scale and nature of experiments IR researchers are increasingly being required to conduct to test and evaluate new search technologies. Our vision for Terrier is to continue empowering researchers and practitioners in IR with up-to-date, effective and scalable tools, allowing them to build and evaluate the next generation IR applications. We hope that many will join us in working together towards such an objective.

Acknowledgements

The authors express their gratitude to all members of the Terrier community who have contributed to the platform over the last decade.

[7] `http://www.smartfp7.eu/content/twitter-indexing-demo`

6. REFERENCES

[1] M.-D. Albakour, C. Macdonald, I. Ounis, A. Pnevmatikakis, and J. Soldatos. SMART: An open source framework for searching the physical world. In *Proc. of OSIR*, 2012.

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

[3] V. N. Anh and A. Moffat. Pruned query evaluation using pre-computed impacts. In *Proc. of SIGIR*, pages 372–379, 2006.

[4] A. Z. Broder, D. Carmel, M. Herscovici, A. Soffer, and J. Zien. Efficient query evaluation using a two-level retrieval process. In *Proc. of CIKM*, pages 426–434, 2003.

[5] F. Cacheda, V. Plachouras, and I. Ounis. A case study of distributed information retrieval architectures to index one terabyte of text. *IP&M*, 41(5):1141–1161, 2005.

[6] J. Dean and S. Ghemawat. MapReduce: Simplified data processing on large clusters. In *Proc. of OSDI*, pages 137–150, 2004.

[7] X. Geng, T.-Y. Liu, T. Qin, A. Arnold, H. Li, and H.-Y. Shum. Query-dependent ranking using k-nearest neighbor. In *Proc. of SIGIR*, pages 115–122, 2008.

[8] B. He and I. Ounis. A query-based pre-retrieval model selection approach to information retrieval. In *Proc. of RIAO*, pages 706–719, 2004.

[9] S. Heinz and J. Zobel. Efficient single-pass index construction for text databases. *J. Am. Soc. Inf. Sci. Technol.*, 54(8):713–729, 2003.

[10] D. Hiemstra. *Using Language Models for Information Retrieval*. PhD thesis, University of Twente, 2001.

[11] T.-Y. Liu. Learning to rank for information retrieval. *Found. Trends Inf. Retr.*, 3(3):225–331, 2009.

[12] C. Macdonald, I. Ounis, and N. Tonellotto. Upper-bound approximations for dynamic pruning. *ACM TOIS*, 29(4):1–28, 2011.

[13] C. Macdonald, V. Plachouras, B. He, C. Lioma, and I. Ounis. University of Glasgow at WebCLEF 2005: experiments in per-field normalisation and language specific stemming. In *Proc. of CLEF*, pages 898–907, 2006.

[14] R. McCreadie, C. Macdonald, and I. Ounis. MapReduce indexing strategies: Studying scalability and efficiency. *IP&M*, 2011.

[15] R. McCreadie, C. Macdonald, and I. Ounis. Crowdterrier: Automatic crowdsourced relevance assessments with terrier. In *Proc. of SIGIR*, 2012.

[16] R. McCreadie, C. Macdonald, and I. Ounis. Identifying top news using crowdsourcing. *Information Retrieval*, 2012.

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

[18] R. L. T. Santos, C. Macdonald, and I. Ounis. Exploiting query reformulations for web search result diversification. In *Proc. of WWW*, pages 881–890, 2010.

[19] R. L. T. Santos, C. Macdonald, and I. Ounis. Selectively diversifying Web search results. In *Proc. of CIKM*, pages 1179–1188, 2010.

[20] R. L. T. Santos, C. Macdonald, and I. Ounis. University of Glasgow at the NTCIR-9 intent task. In *Proc. of NTCIR*, 2011.

[21] I. Soboroff, D. McCullough, J. Lin, C. Macdonald, I. Ounis, and R. McCreadie. Evaluating real-time search over tweets. In *Proc. of ICWSM*, 2012.

[22] H. Zaragoza, N. Craswell, M. J. Taylor, S. Saria, and S. E. Robertson. Microsoft Cambridge at TREC 13: Web and Hard tracks. In *Proc. of TREC*, 2004.

[23] C. Zhai and J. Lafferty. A study of smoothing methods for language models applied to information retrieval. *ACM TOIS*, 22(2):179–214, 2004.