Terrier: A High Performance and Scalable Information Retrieval Platform

Iadh Ounis, Gianni Amati*, Vassilis Plachouras, Ben He,
Craig Macdonald, Christina Lioma
Department of Computing Science
University of Glasgow Scotland, UK
{ounis,gianni,vassilis,ben,craigm,xristina}@dcs.gla.ac.uk

ABSTRACT

In this paper, we describe Terrier, a high performance and scalable search engine that allows the rapid development of large-scale retrieval applications. We focus on the open-source version of the software, which provides a comprehensive, flexible, robust, and transparent test-bed platform for research and experimentation in Information Retrieval (IR).

Categories and Subject Descriptors

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

General Terms

Algorithms, Design, Experimentation, Performance, Theory

Keywords

Terrier, Information Retrieval platform, Open Source

1. INTRODUCTION

Terrier [12], Terabyte Retriever, is a project that was initiated at the University of Glasgow in 2000, with the aim to provide a flexible platform for the rapid development of large-scale Information Retrieval applications, as well as a state-of-the-art test-bed for research and experimentation in the wider field of IR.

The Terrier project explored novel, efficient and effective search methods for large-scale document collections, combining new and cutting-edge ideas from probabilistic theory, statistical analysis, and data compression techniques. This led to the development of various retrieval approaches using a new and highly effective probabilistic framework for IR, with the practical aim of combining efficiently and effectively various sources of evidence to enhance the retrieval performance.

In particular, we have developed several new hyperlink structure analysis methods [15], various selective combination of evidence approaches [13, 16], new document length normalisation methods [1, 7, 8], many automatic query expansion and re-formulation techniques [1, 11], as well as a comprehensive set of query performance predictors [6, 9], to support a wide range of retrieval tasks and applications.

In addition, an effective in-house, highly configurable and scalable crawler called Labrador[1] has been developed. The crawler was used for the deployment of Terrier in various industrial applications. Various powerful compression techniques have been explored and implemented, making Terrier particularly suitable for large-scale collections. Finally, the project investigated distributed architectures and retrieval [4], which allowed the system to be used both in a centralised and distributed setting.

This paper focuses on a core version of Terrier, an open source product, which has been available to the general public since November 2004 under the MPL license[2]. The open source version of Terrier is written in Java, allowing the system to run on various operating systems platforms, and a wide variety of hardware. Terrier, the latest version of which is 1.0.2, is available to download from `http://ir.dcs.gla.ac.uk/terrier/`.

2. MOTIVATIONS & AIMS

In any experimental science field, as is the case in IR, it is crucial to have a system allowing large-scale experimentation to be conducted in a flexible, robust, transparent and reproducible way. Terrier addresses this important issue by providing a test-bed framework for driving research and facilitating experimentation in IR.

The major leading IR groups in the world have always had an experimental platform to support their research. Most of the existing IR platforms are designed towards supporting a particular IR method/approach, making them difficult to customise and tailor to new methods and approaches.

Terrier is the first serious answer in Europe to the dominance of the United States on research and technological solutions in IR. Our objective was to include the state-of-the-art IR methods and techniques, and offer the most efficient and effective IR technologies, into a transparent, easily extensible and modular open source software. The open source product of Terrier was therefore designed as a tool to evaluate, test and compare models and ideas, and to build systems for large-scale IR.

The system includes efficient and effective state-of-the-art retrieval models based on a new Divergence From Randomness (DFR) framework for deriving parameter-free proba-

*Also affiliated to Fondazione Ugo Bordoni, Rome

Copyright is held by the author/owner(s).
SIGIR Open Source Workshop '06 Seattle, Washington, USA.

[1] `http://ir.dcs.gla.ac.uk/labrador/`
[2] `http://www.mozilla.org/MPL/`

bilistic models for IR [1]. The DFR models are information-theoretic models, for both query expansion and document ranking, a versatile feature that is not possessed by any other IR model. In addition, for easy cross-comparison of different retrieval models, Terrier includes a wide variety of models, ranging from numerous forms of the classical TF-IDF weighting scheme to the recent language modelling approach, through the well-established Okapi's BM25 probabilistic ranking formula.

Terrier provides various indexing and querying APIs, and allows rapidly experimenting with new concepts/ideas on various collections, and in different configuration settings. This important feature for a research platform is made possible by the modular architecture of the system, its various easy-to-use configuration options, as well as its use of the most recent software engineering methods. The system is very easy to work with, further helped with a comprehensive documentation[3], and is easy to extend and adapt to new applications. This is demonstrated by the use of the platform in such diverse search tasks as desktop search, web search, e-mail & expertise search, XML retrieval, multilingual and blogs search.

Finally, Terrier includes various retrieval performance evaluation tools, which produce an extensive analysis of the retrieval effectiveness of the tested models/concepts on given test collections. Systems and their variants can be instantiated in parallel, an important feature for conducting numerous experiments at the same time, thus fostering extensive experimentation in a structured and systematic way.

The remainder of this paper is structured as follows. We describe the main indexing features of Terrier in Section 3, and their associated data structures in Section 4. Section 5 provides an overview of the retrieval features of Terrier, which make the system particularly suitable for research purposes. In Section 6, we describe a very interesting functionality of Terrier, namely query expansion, and outline how it can be used for other IR activities and applications. The retrieval performance evaluation package of Terrier is presented in Section 7, while some "out-of-the-box" applications of Terrier are described in Section 8. Finally, we conclude the paper with lessons learnt while building the Terrier platform.

3. INDEXING FEATURES

Figure 1 outlines the indexing process in Terrier. Terrier is designed to allow many different ways of indexing a corpus of documents. Indexing is a four stage process, and at each stage, plugins can be added to alter the indexing process. This modular architecture allows flexibility in the indexing process at several stages: in the handling of a corpus of documents, in handling and parsing each individual document, in the processing of terms from documents, and in writing the index data structures. In addition, Terrier allows the direct parsing and indexing of compressed collections.

The open source version of Terrier comes with support for indexing test collections from the well-established TREC[4] international evaluation forum. This permits researchers to quickly set up and run Terrier with a test collection, lessening the learning curve for the platform. The advantage is that any other storage method for a collection of documents

[3] http://ir.dcs.gla.ac.uk/terrier/docs.html
[4] http://trec.nist.gov

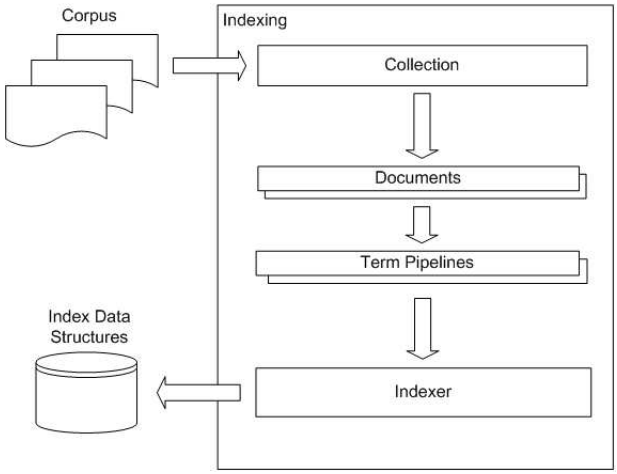

Figure 1: Overview of the indexing architecture of Terrier. A corpus of documents is handled by a Collection plugin, which generates a stream of Document objects. Each Document generates a stream of terms, which are transformed by a series of Term Pipeline components, after which the Indexer writes to disk.

can easily be supported by adding a new Collection plugin. For example, reading a stream of documents from an email server would only require implementing a new Collection plugin to connect to the email server.

Terrier comes with various document parsers embedded. These include the ability to index HTML documents, Plain text documents, Microsoft Word, Excel and Powerpoint documents, as well as Adobe PDF files. To add support for another file format to Terrier, a developer must only add another Document plugin which is able to extract the terms from the document.

Each term extracted from a document has three fundamental properties: firstly, the actual String textual form of the term; secondly, the position at which the term occurs in the document, and thirdly, the fields in which the term occurs (fields can be arbitrarily defined by the Document plugin, but typically are used to define which HTML tags a term occurs in). Terrier adds configurability at this stage of the indexing: terms are passed through a 'Term Pipeline', which allows the terms to be transformed in various ways. Terrier comes with some pre-defined Term Pipeline plugins, such as two variants of Porter's stemming algorithm, and a stopwords removal component. Term Pipeline plugins introduce flexibility to the processing and transformation of terms, which can be manifested for example by n-gram indexing, adding stemming and stopword removal in various languages, acronym expansion, or use of a thesaurus, to name but a few.

The last component in the Term Pipeline chain is always the Indexer. The Indexer is responsible for writing the index using the appropriate data structures. By using a Block Indexer, it is possible to create an index that stores the positions of terms in each document. Each document is divided into blocks - the size of a block defines the accuracy with which term positions are recorded. By default, the size of a block is 1 and, in that case, the exact positions of the

term occurrences are recorded. This allows for proximity and phrase operators to be used in queries during retrieval. However, a block could also be defined as a semantic entity, so as to allow structured retrieval. The following section describes the index data structures generated by Terrier when indexing a corpus of documents.

4. INDEX STRUCTURES

A Terrier index consists of 4 main data structures, in addition to some auxiliary files:

- **Lexicon:**
  The lexicon stores the term and its term id (a unique number for each term), along with the global statistics of the term (global term frequency and document frequency of the term) and the offsets of the postings list in the Inverted Index.

- **Inverted Index:**
  The inverted index stores the postings lists of a term. In particular, for each term, the inverted index stores: the document id of the matching document; and the term frequency of the term in that document. The fields in which the term occurred are encoded using a bit set. If the index has been made with positional information, the postings list will also contain the block ids in the document that the term occurs in. Positional information allows phrasal and proximity search to be performed.

  The postings list in Terrier is highly compressed. In particular, the document ids are encoded in the inverted index using Gamma encoding, the term frequencies are encoded using Unary encoding, and the Block ids, if present, are encoded using Gamma encoding [20].

- **Document Index:**
  The Document Index stores the document number (an external unique identifier of the document), the document id (internal unique identifier of the document); the length of the document in terms of tokens; and the offset of the document in the Direct Index.

- **Direct Index:**
  The Direct Index stores the terms and term frequencies of the terms present in each document. The aim of the Direct Index is to facilitate easy and efficient query expansion, as described in Section 6. However, the direct index is also extremely useful for applying clustering to a collection of documents.

  The direct index contents are compressed in an orthogonal way to the inverted index. Term ids are written using Gamma encoding, term frequencies use Unary encoding and the fields in which the terms occur are encoded using a bit set. Block ids, if present, are encoded using Gamma encoding.

Table 1 details the format of each file in the index structures. Moreover, Table 2 shows the sizes of Terrier's index data structures after indexing the WT2G collection. Fields are not recorded, while index sizes with and without term positions are stated. For comparison purposes, the index sizes of Terrier without compression, and two other open source products, Indri[5] and Zettair[6] for the same collection are also provided. Like Terrier, Indri and Zettair are deployed using their default settings.

Index Structure	Contents
Lexicon	Term (20) Term id (4) Document Frequency (4) Term Frequency (4) Byte offset in inverted file (8) Bit offset in inverted file (1)
Inverted Index	Document id gap (gamma code) Term Frequency (unary code) Fields (# of fields bits) Block Frequency (unary code) [Block id gap (gamma code)]
Document Index	Document id (4) Document Length (4) Document Number (20) Byte offset in direct file (8) Bit offset in direct file (1)
Direct Index	Term id gap (gamma code) Term frequency (unary code) Fields (# of fields bits) Block frequency (unary code) [Block id gap (gamma code)]
Lexicon Index	Offset in lexicon (8)
Collection Statistics	# of documents # of tokens # of unique terms # of pointers

Table 1: Details on the format and compression used for each index data structure. The numbers in parenthesis are the size of each entry in bytes, unless otherwise denoted. Gamma and unary codes are variable length encodings.

5. RETRIEVAL FEATURES

One of the main aims of Terrier is to facilitate research in the field of IR. Figure 2 provides an outline of the retrieval process in Terrier. Terrier's retrieval features have been selected in order to be useful for a wide range of IR research. Indeed, Terrier offers great flexibility in choosing a weighting model (Section 5.1), as well as in altering the score of the retrieved documents (Section 5.2). Moreover, Terrier offers an advanced query language (Section 5.3). Another very important retrieval feature of Terrier is the automatic query expansion, which is described in Section 6.

5.1 Weighting Models

The core functionality of matching documents to queries and ranking documents takes place in the Matching module. Matching employs a weighting model to assign a score to each of the query terms in a document. In order to facilitate the cross-comparison of weighting models, a range of weighting models is supplied, including BM25, TF-IDF, and document weighting models from the Divergence From Ran-

[5] `http://www.lemurproject.org/indri/`
[6] `http://www.seg.rmit.edu.au/zettair/`

Index Structure	Index Sizes	% Of WT2G
Terrier 1.0.2		
Direct Index	87MB	4%
Inverted Index	72MB	3%
Lexicon	40MB	2%
Document Index	8.8MB	0.4%
Terrier if no compression is used		
Direct Index	481MB	22%
Inverted Index	481MB	22%
Lexicon	40MB	2%
Document Index	8.8MB	0.4%
Terrier 1.0.2 with Term Positions		
Direct Index	366MB	17%
Inverted Index	349MB	17%
Lexicon	40MB	2%
Document Index	8.8MB	0.4%
Indri 2.2		
Direct Index	386MB	18%
Inverted Index	417MB	19%
Zettair 0.6.1		
Inverted Index	430MB	20%

Table 2: Index sizes for Terrier (with and without term positions), Indri and Zettair for the WT2G collection (2.1GB). Terrier's uncompressed index size is provided for comparison purposes.

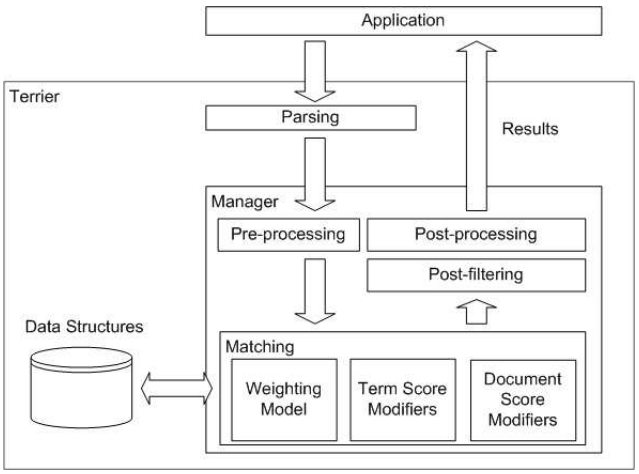

Figure 2: Overview of the retrieval architecture of Terrier. The application communicates with the Manager, which in turn runs the desired Matching module. Matching assigns scores to the documents using the combination of weighting model and score modifiers.

domness (DFR) framework [1]. The DFR approach supplies parameter-free probabilistic models, based on a simple idea:

> *"The more the divergence of the within-document term-frequency from its frequency within the collection, the more the information carried by the term t in the document d".*

The open source 1.0.2 version of Terrier includes eight document weighting models from the DFR framework, all of which perform robustly on standard test collections. Ponte-Croft's language model [18] is also supported.

5.2 Altering Document Scores

The score of an individual term in a document can be altered by applying a TermScoreModifier. For example, a TermInFieldModifier can be applied in order to ensure that the query terms occur in a particular field of a document. If a query term does not appear in a particular field, then the TermInFieldModifier resets the score of the document. In addition, a FieldScoreModifier boosts the score of a document in which a query term appears in a particular field.

Similarly, changing the score of a retrieved document is achieved by applying a DocumentScoreModifier. One such modifier is the PhraseScoreModifier, which employs the position information saved in the index of Terrier, and resets the score of the retrieved documents in which the query terms do not appear as a phrase, or within a given number of blocks. Generally, a DocumentScoreModifier is ideal for applying query-independent evidence, such as evidence from hyperlink structure analysis, or from the URL of documents. Moreover, the selective application of different retrieval techniques based on evidence from the hyperlink structure [13] can be applied as a DocumentScoreModifier.

After the application of any TermScoreModifiers or DocumentScoreModifiers, the set of retrieved documents can be further altered by applying post processing or post filtering. Post processing is appropriate to implement functionalities that require changing the original query. An example of post processing is Query Expansion (QE), which is described in detail in Section 6. The application of QE could be enabled on a per-query basis, before retrieval, depending on the output of a pre-retrieval performance predictor [6]. Another possible example of post processing could be the application of clustering.

Post filtering is the final step in Terrier's retrieval process, where a series of filters can remove already retrieved documents, which do not satisfy a given condition. For example, in the context of a Web search engine, a post filter could be used to reduce the number of retrieved documents from the same Web site, in order to increase diversity in the results.

5.3 Query Language

Terrier includes a powerful query language that allows the user to specify additional operations on top of the normal probabilistic queries. Such operations may specify that a particular query term should or should not appear in the retrieved documents. Other available operations include requiring that terms appear in particular fields, phrase queries, or proximity queries. An overview of the available query language operations is given below.

- `t1 t2` retrieves documents with either `t1` or `t2`.
- `t1^2.3` sets the weight of query term `t1` to 2.3.

- `+t1 -t2` retrieves documents with `t1` but not t2.
- ``t1 t2'' retrieves documents where the terms `t1` and `t2` occur next to each other.
- ``t1 t2''∼n retrieves documents where the terms `t1`, `t2` occur within n blocks.
- `+(t1 t2)` specifies that both terms `t1` and `t2` are required.
- `field:t1` retrieves docs where `t1` must appear in the specified field.
- `control:on/off` enables or disables a given control. For example, query expansion is enabled with `qe:on`.

The query language operations correspond to TermScoreModifier or DocumentScoreModifier modules, which are appropriately configured when the query is parsed. Retrieval parameters can also be specified by the user, when permitted, to change or enable retrieval functionalities. For example, a query for which a particular query term should appear in the title of the retrieved documents, automatically enables a TermInFieldModifier.

6. QUERY EXPANSION

Terrier includes automatic pseudo-relevance feedback, in the form of Query Expansion (QE). The method works by taking the top most informative terms from the top-ranked documents of the query, and adding these new related terms into the query. This operation is made possible by the presence of the direct index. The direct index allows the terms and their frequencies to be determined for each document in the index. The new query is reweighted and rerun - providing a richer set of retrieved documents. Terrier provides several term weighting models from the DFR framework which are useful for identifying informative terms from top-ranked documents. In addition, for easy cross-comparison, Terrier also includes some well-established pseudo-relevance feedback techniques, such as Rocchio's method [19].

Automatic query expansion is highly effective for many IR tasks. The decision of applying QE depends on the type of application, and the difficulty of queries. Therefore, we have developed several tools for determining possible indicators that predict the query difficulty, and the selective application of query expansion [2, 6, 9]. The vocabulary and size of the collections may be largely different. As a consequence, automatic means to tune the inherent parameters of the query expansion are also necessary in a good IR system. For this purpose, we devised a parameter-free QE mechanism, called Bo1 [1, 11], as a default in Terrier. The system is also readily available to support several low-cost methods for a selective application of query expansion. Our method of QE guarantees the most effective and efficient mechanism for performing QE, even with only using 3 retrieved documents and a handful of additional query terms. In addition, QE can be also used to automatically generate realistic samples of queries [6].

Data structures, in particular the direct file, tools and methods adopted for performing QE in Terrier, also constitute an advanced basis for other types of IR activities and applications, such as document and term clustering or question answering. Indeed, the query expansion methodology is based on a filter, activated on the information theoretic weights of the terms, which selects the highly informative terms that appear in at least X retrieved documents. As an illustration, the optimal value for X is 2 in adhoc retrieval. The clustering of the results would require a higher value for X, while question answering may require a different information theoretic definition for X. In conclusion, QE is a good source of inspiration for new and interesting IR applications and research directions.

Collection	Topics/Task	Performance
WT2G	Adhoc Topics 401-450	0.2663 MAP
WT10G	Adhoc Topics 451-550	0.1886 MAP
W3C	Known Item Topics 26-150	0.4960 MRR

Table 3: Performance of "out-of-the-box" Terrier on standard TREC collections and tasks. Performance is measured using the standard measure for each task: for Adhoc, Mean Average Precision (MAP); for Known Item, Mean Reciprocal Rank (MRR).

7. EVALUATION PACKAGE

When performing IR research using a standard test collection, it is essential to evaluate the retrieval performance of the applied approaches. The open source version of Terrier was designed to allow users to rapidly design and test new retrieval functions and IR models. For this reason, we implemented the evaluation tools in the TREC style with a wide range of known IR measures. In particular, Terrier includes and integrates the main evaluation functionalities found in the trec_eval tool, including, among others, calculating Mean Average Precision, Precision @ rank N, interpolated Precision, and R-Precision. It is also possible to show the evaluation measures on a per-query basis.

The evaluation tools of Terrier allow easy testing of new retrieval methodologies or approaches. Once the new approach has been integrated into Terrier, users can test it, together with as many models as possible, against a test collection. As soon as the system has terminated all the runs, for each model, the file of the evaluation results can be created and a summary is displayed.

8. STANDARD DEPLOYMENTS & SAMPLE APPLICATIONS

Terrier 1.0.2 comes with two applications. Firstly, Terrier comprises a complete application suite for performing research using test collections. Terrier is easy to deploy on the standard TREC test collections, e.g. the TREC CDs, WT2G, WT10G, .GOV etc. This facilitates research on these standard test collections. Table 3 shows "out-of-the-box" retrieval performance scores of Terrier on standard test collections, using title-only queries. The default settings include the removal of English stopwords and the application of Porter's stemming algorithm. During retrieval, the InL2 DFR weighting model is applied as is.

Secondly, Terrier comes with a proof-of-concept Desktop Search application, shown in Figure 3. This shows how the retrieval functionalities of Terrier can be easily deployed to an application suitable for day-to-day retrieval tasks. All the retrieval functionalities are easily accessible from the Desktop Search application, including Terrier's query language and query expansion.

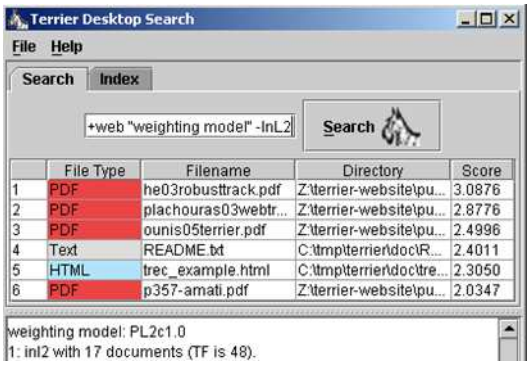

Figure 3: Screenshot of the proof-of-concept Terrier Desktop Search. It can index various types of common document formats, including Microsoft Office documents, HTML and PDF files.

9. RESEARCH FACILITATED & LESSONS LEARNED

The development of a test-bed framework allowing for the rapid experimentation of new IR concepts/ideas has boosted our research, yielding some significant insights into the behaviour of IR techniques on diverse, multilingual, and large-scale collections. The system allowed us to successfully and significantly participate in various TREC tracks such as the Web tracks (2001-2004), the Robust tracks (2003, 2004), the Terabyte tracks (2004-2005), and recently the Enterprise track (2005). In addition, we participated in various tracks of CLEF[7] in 2003-2005.

More importantly, in addition to being a vehicle for the evaluation of our new search methods, the Terrier platform allowed us to easily simulate, assess, and improve state-of-the-art IR methods and techniques. This led to a better understanding of the underlying issues related to the tackling of different search tasks on various and diverse large collections. In particular, we realised the need to build parameter-free models, which avoid the constraint of using costly parameter-tuning approaches based on relevance assessment. The proposed DFR framework is not only an efficient and effective IR model, but also a mechanism by which we can explain and complement other existing IR approaches/techniques [1].

One of the main lessons learnt from building the Terrier platform during the previous years is the fact that building a sustainable, flexible and robust platform for information retrieval is equally important as creating new IR models. Indeed, research in an experimental field such as IR cannot be conducted without a platform facilitating experimentation and evaluation.

In developing a cutting-edge technology from a laboratory setting to a product release, which was deployed in various industrial applications and settings, we gained a much better understanding of the challenges faced in tackling real large-scale collections, not necessarily as clean as the TREC or CLEF collections, and how to provide practical and effective solutions under a high query workload.

Many benefits were achieved by releasing a core version of the project as an open source software. By facilitating a transparent and reproducible experimentation of various retrieval methods, we believe that we achieved a greater impact and visibility in the scientific community. We have also built a community of users/developers around Terrier, which helped improving the quality of the software and attracted more researchers to the project. Since its release in November 2004, the Terrier software has been downloaded thousands of times from across the world, including major commercial search engine companies.

Finally, the Terrier project has attracted many undergraduate, postgraduate, and research students, and benefited from their contributions. Building an IR platform is both a research and engineering process, requiring team work and a critical mass of developers. Terrier is currently the main development platform for both undergraduate and postgraduate students in our research group, allowing them to employ a state-of-the-art framework in their learning and research.

10. CONCLUSIONS

In this paper, we described the open source version of Terrier, a robust, transparent, and modular framework for research and experimentation in Information Retrieval. The platform was developed as part of an ongoing larger project, ultimately aiming at becoming the European Search Engine, with the state-of-the-art search technology for various search tasks and applications.

Terrier is a mature and state-of-the-art platform, including highly effective retrieval models, based on the DFR framework. The performance of these models is at least comparable to, if not better than, that of the most recent models, including on very large-scale test collections [14, 11]. In addition, Terrier's DFR models are information-theoretic models for both query expansion and document ranking, a versatile and original feature that is not possessed by any other IR model. Terrier is continuously expanded with new models resulting from our research. Indeed, new research, such as models for document structure and combination of evidence [17], length normalisation methods [7], query performance prediction [6, 9], NLP techniques [10], information extraction [3], is being readily fed to Terrier. Furthermore, Terrier is being used for research in enterprise and expert search [11]. Retrieval from large-scale test collections has also led us to study optimal distributed architectures for retrieval systems [4, 5].

From an IR perspective, technological challenges may render an IR model obsolete very quickly (as has happened with the Boolean or Vector Space model). We believe that a theoretical framework for Information Retrieval, when implemented within a mature, robust and extensible IR system, is destined to last. Our strong belief is that Terrier, as the accumulator of innovative ideas in IR, and with its design features and underlying principles, open source being one of them, will make easier technology transfers.

11. ACKNOWLEDGEMENTS

The development of Terrier required a significant manpower and infrastructure. We would like to thank those many students and programmers who participated in developing Terrier. We would also like to thank the UK Engineering and Physical Sciences Research Council (EPSRC), project grant number GR/R90543/01, as well as the Leverhulme Trust, project grant number F/00179/S, who have

[7] http://www.clef-campaign.org

partially supported the development of Terrier. Finally, we thank the Department of Computing Science at the University of Glasgow, and Keith van Rijsbergen, for supporting the Terrier platform at various stages of its development.

12. REFERENCES

[1] G. Amati. *Probabilistic Models for Information Retrieval based on Divergence from Randomness*. PhD thesis, Department of Computing Science, University of Glasgow, 2003.

[2] G. Amati, C. Carpineto, and G. Romano. Query Difficulty, Robustness, and Selective Application of Query Expansion. In *ECIR '04: Proceedings of the 26th European Conference on Information Retrieval (ECIR'04)*, pages 127-137, Sunderland, UK, 2004. Springer.

[3] G. Amati. Information Theoretic Approach to Information Extraction. In Proceedings of the 7th international conference on Flexible Query Answering Systems (FQAS 2006), pages 519–529, Milan, Italy, 2006.

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

[5] F. Cacheda, V. Carneiro, V. Plachouras, and I. Ounis. Performance Analysis of Distributed Information Retrieval Architectures Using an Improved Network Simulation Model. *Information Processing & Management* (to appear).

[6] B. He, and I. Ounis. Inferring Query Performance Using Pre-retrieval Predictors. In *Proceedings of the 11th Symposium on String Processing and Information Retrieval (SPIRE 2004)*, Padova, Italy, 2004.

[7] B. He, and I. Ounis. A study of the Dirichlet priors for term frequency normalisation. In *SIGIR '05: Proceedings of the 28th annual international ACM SIGIR conference on Research and Development in information retrieval*, pages 465–471, Salvador, Brazil, 2004.

[8] B. He, and I. Ounis. Term Frequency Normalisation Tuning for BM25 and DFR Models. In *ECIR '05: Proceedings of the 27th European Conference on Information Retrieval (ECIR'05)*, pages 200–214, Santiago de Compostela, Spain, 2005. Springer.

[9] B. He, and I. Ounis. Query Performance Prediction. In *Information Systems*, Elsevier, 2006 (In press).

[10] C. Lioma, and I. Ounis. Examining the Content Load of Part-of-Speech Blocks for Information Retrieval. In *Joint Conference of the International Committee on Computational Linguistics and the Association for Computational Linguistics, (COLING/ACL 2006)*, Sydney, Australia, 2006.

[11] C. Macdonald, B. He, V. Plachouras, and I. Ounis. University of Glasgow at TREC2005: Experiments in Terabyte and Enterprise Tracks with Terrier. In *TREC '05: Proceedings of the 14th Text REtrieval Conference (TREC 2005)*, November, 2005, NIST.

[12] I. Ounis, G. Amati, Plachouras V., B. He, C. Macdonald, and D. Johnson. Terrier Information Retrieval Platform. In *Proceedings of the 27th European Conference on IR Research (ECIR 2005)*, volume 3408 of *Lecture Notes in Computer Science*, pages 517–519. Springer, 2005.

[13] V. Plachouras, and I. Ounis. Usefulness of hyperlink structure for query-biased topic distillation. In *SIGIR '04: Proceedings of the 27th annual international ACM SIGIR conference on Research and Development in information retrieval*, pages 48–455, Sheffield, UK, 2004.

[14] V. Plachouras, B. He, and I. Ounis. University of Glasgow at TREC2004: Experiments in Web, Robust and Terabyte tracks with Terrier. In *TREC '04: Proceedings of the 13th Text REtrieval Conference (TREC 2004)*, November 2004, NIST.

[15] V. Plachouras, I. Ounis, and G. Amati. The Static Absorbing Model for the Web. *Journal of Web Engineering*, 4(2):165–186, 2005.

[16] V. Plachouras, F. Cacheda, and I. Ounis. A Decision Mechanism for the Selective Combination of Evidence in Topic Distillation. *Information Retrieval*, 9(2):139–163, 2006.

[17] V. Plachouras. *Selective Web Information Retrieval*. PhD thesis, Department of Computing Science, University of Glasgow, 2006.

[18] J. M. Ponte and W. B. Croft. A language modelling approach to information retrieval. In *SIGIR '98: Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 275–281, Melbourne, Australia, 1998.

[19] J. Rocchio. Relevance feedback in information retrieval. In *The Smart Retrieval system - Experiments in Automatic Document Processing*, In Salton, G., Ed. Prentice-Hall Englewood Cliffs. NJ.313-323, 1971.

[20] I.H. Witten, A. Moffat, and T.C. Bell. *Managing Gigabytes: Compressing and Indexing Documents and Images*. Morgan Kaufmann Publishers, 1999.