University of Glasgow at TREC2004: Experiments in Web, Robust and Terabyte tracks with Terrier

Vassilis Plachouras, Ben He, and Iadh Ounis

University of Glasgow
G12 8QQ Glasgow, UK

Abstract

With our participation in TREC2004, we test Terrier, a modular and scalable Information Retrieval framework, in three tracks. For the mixed query task of the Web track, we employ a decision mechanism for selecting appropriate retrieval approaches on a per-query basis. For the robust track, in order to cope with the poorly-performing queries, we use two pre-retrieval performance predictors and a weighting function recommender mechanism. We also test a new training approach for the automatic tuning of the term frequency normalisation parameters. In the Terabyte track, we employ a distributed version of Terrier and test the effectiveness of techniques, such as using the anchor text, query expansion and selecting an optimal weighting model for each query. Overall, in all three tracks we participated, Terrier and the tested Divergence From Randomness models were shown to be stable and effective.

1 Introduction

With our participation in TREC2004, we test our Information Retrieval (IR) framework, Terrier, in a variety of different settings. Terrier is a modular and scalable framework, for the rapid development of large-scale IR applications. It provides indexing and retrieval functionalities, as well as a number of parameter-free weighting models, based on the Divergence From Randomness (DFR) framework [2]. Terrier stands for TErabyte RetrIEveR, and further information can be found at http://ir.dcs.gla.ac.uk/terrier.

We have submitted official runs to three tracks of TREC2004. For the Web track, we test the selective application of different retrieval approaches on a per-query basis. In the Robust track, we employ two novel pre-retrieval performance predictors in a weighting function recommender mechanism, in order to use the optimal weighting functions/models for the poorly-performing queries. We also refine the automatic tuning of the term frequency normalisation parameters, by creating samples of queries, instead of using relevance information. In both the Web and Robust tracks, we use a centralised version of Terrier. For the Terabyte track, we use Terrier in a distributed setting, in order to handle the test collection .GOV2, and evaluate retrieval techniques, which have been effective in the context of previous ad-hoc and Web retrieval TREC tasks. In all three tracks we have participated, Terrier performed extremely well and the tested DFR models were shown to be effective in different settings.

The remainder of the paper is organised as follows. Section 2 contains a description of the Terrier framework. In Section 3, we describe our approach for the mixed query task of the Web track. Section 4 presents our experiments for the Robust track. In Section 5, we describe our participation in the Terabyte track, and we close with some concluding remarks in Section 6.

2 Terrier Information Retrieval Framework

Among its various features, Terrier offers a number of DFR-based models for document weighting, as well as classical models, such as BM25 [19] and tf-idf, and recent models, such as Ponte-Croft's language model [17]. Terrier also provides DFR-based and classical term weighting models for query expansion. The relevance score of a document d for a particular query Q is given by:

$$\text{score}(d, Q) = \sum_{t \in Q} \big(qtfn \cdot w(t, d)\big) \tag{1}$$

where $w(t, d)$ is the weight of the document d for a query term t and $qtfn$ is the normalised frequency of term t in the query. It is given by qtf/qtf_{max}, where qtf is the original frequency of term t in the query, and qtf_{max} is the maximum qtf of all the composing terms of the query. In Table 1, we provide the formulas for the different models $w(t, d)$ we have used in our experiments for TREC2004.

Model	Formula
BB2	$w(t,d) = \frac{F+1}{N_t \cdot (tfn+1)}\big(-\log_2(N-1) - \log_2(e) + f(N+F-1, N+F-tfn-2) - f(F, F-tfn)\big)$
BL2	$w(t,d) = \frac{1}{tfn+1}\big(-\log_2(N-1) - \log_2(e) + f(N+F-1, N+F-tfn-2) - f(F, F-tfn)\big)$
PB2	$w(t,d) = \frac{F+1}{N_t \cdot (tfn+1)}\big(tfn \cdot \log_2 \frac{tfn}{\lambda} + (\lambda + \frac{1}{12 \cdot tfn} - tfn) \cdot \log_2 e + 0.5 \cdot \log_2(2\pi \cdot tfn)\big)$
PL2	$w(t,d) = \frac{1}{tfn+1}\big(tfn \cdot \log_2 \frac{tfn}{\lambda} + (\lambda + \frac{1}{12 \cdot tfn} - tfn) \cdot \log_2 e + 0.5 \cdot \log_2(2\pi \cdot tfn)\big)$
I(n)B2	$w(t,d) = \frac{F+1}{N_t \cdot (tfn+1)}\big(tfn \cdot \log_2 \frac{N+1}{N_t+0.5}\big)$
I(n)L2	$w(t,d) = \frac{1}{tfn+1}\big(tfn \cdot \log_2 \frac{N+1}{N_t+0.5}\big)$
I(F)B2	$w(t,d) = \frac{F+1}{N_t \cdot (tfn+1)}\big(tfn \cdot \log_2 \frac{N+1}{F+0.5}\big)$
I(F)L2	$w(t,d) = \frac{1}{tfn+1}\big(tfn \cdot \log_2 \frac{N+1}{F+0.5}\big)$
I(n_e)B2	$w(t,d) = \frac{F+1}{N_t \cdot (tfn+1)}\big(tfn \cdot \log_2 \frac{N+1}{n_e+0.5}\big)$
I(n_e)L2	$w(t,d) = \frac{1}{tfn+1}\big(tfn \cdot \log_2 \frac{N+1}{n_e+0.5}\big)$
I(n_e)C2	$w(t,d) = \frac{F+1}{N_t \cdot (tfn_e+1)}\big(tfn_e \cdot \log_2 \frac{N+1}{n_e+0.5}\big)$

Table 1: Terrier DFR-based document weighting models used in the experiments for TREC2004.

The notation in Table 1 is explained below:

- tf is the within-document frequency of term t in document d.
- F is the term frequency of term t in the whole collection.
- N is the number of documents in the collection.
- N_t is the document frequency of term t.
- n_e is given by $N \cdot \big(1 - (1 - \frac{N_t}{N})^F\big)$.
- λ is given by $\frac{F}{N}$ and $F \ll N$.
- The relation f is given by the Stirling formula:

$$f(n, m) = (m + 0.5) \cdot \log_2\big(\frac{n}{m}\big) + (n - m) \cdot \log_2 n \tag{2}$$

- tfn is the normalised term frequency. It is given by the *normalisation 2*:

$$tfn = tf \cdot \log_2\big(1 + c \cdot \frac{avg\_l}{l}\big) \tag{3}$$

  where c is a parameter, l is the document length, which corresponds to the number of tokens in a document, and avg_l is the average document length in the collection.

- tfn_e is the normalised term frequency, which is given by the modified version of the normalisation 2:

$$tfn_e = tf \cdot \log_e\left(1 + c \cdot \frac{avg\_l}{l}\right) \quad (4)$$

The only free parameter of the DFR framework is the term frequency normalisation parameter c from Equations (3) and (4). The tuning of such a parameter is a crucial issue in IR, because it has an important impact on the retrieval performance [7, 2]. A classical tuning method is the pivoted normalisation [21], which fits the document length distribution to the length distribution of relevant documents. However, since the document length distribution is collection-dependent, the pivoted normalisation suffers from the collection-dependency problem. Indeed, the optimal parameter settings of diverse document collections are different [7].

In our experiments with Terrier, the parameter c is automatically tuned, according to a method proposed by He and Ounis [12]. This method assumes a constant optimal normalisation effect with respect to the document length distribution of the collection, and it assigns the parameter value such that it gives this constant. Thus, it is a collection-independent approach. The proposed method in [12] uses real queries and the corresponding relevance information for training. Moreover, this tuning method can also be applied to Okapi's BM25 [14].

Terrier provides various DFR-based models for query expansion, based on extracting the most informative terms from a set of top-ranked documents. In Table 2, we present the term weighting models $w(t)$ used in our experiments for TREC2004.

Model	Formula
KL	$w(t) = P_x \cdot \log_2 \frac{P_x}{P_c}$
Bo1	$w(t) = tf_x \cdot \log_2 \frac{1+P_n}{P_n} + \log_2(1 + P_n)$
Bo2	$w(t) = tf_x \cdot \log_2 \frac{1+P_f}{P_f} + \log_2(1 + P_f)$
CS	$w(t) = l_x \cdot D + 0.5 \cdot \log_2(\pi \cdot l_x \cdot \frac{1-tf_x}{token_c})$

Table 2: Terrier DFR-based term weighting models used in the experiments for TREC2004.

The notation in Table 2, is explained below:

- l_x is the sum of the length of the exp_doc top-ranked documents, and exp_doc is a parameter of the query expansion methodology.
- tf_x is the frequency of the query term in the top-ranked documents.
- $token_c$ is the total number of tokens in the whole collection.
- P_n is given by $\frac{F}{N}$, where F is the term frequency of the query term in the whole collection and N is the number of documents in the whole collection.
- P_f is given by $\frac{tf_x \cdot l_x}{token_c}$.
- D is given by:

$$P_x \cdot \log_2 \frac{P_x}{P_c} + P_x \cdot \log_2 \frac{1 - P_x}{1 - P_c} \quad (5)$$

where $P_x = tf_x / l_x$ and $P_c = \frac{F}{token_c}$.

The normalised query term frequency $qtfn$ of an expanded query term is given by:

$$qtfn = \frac{w(t)}{w_{max}(t)} \quad (6)$$

where $w(t)$ is the weight of term t and $w_{max}(t)$ is the maximum $w(t)$ of the expanded query terms.

3 Web Track

Our experiments for the Web track of TREC2004 continue the evaluation of a decision mechanism for the dynamic application of appropriate retrieval approaches on a per-query basis. We use Terrier, a modular Information Retrieval framework and its associated DFR-based weighting models, as described in Section 2.

We have submitted runs for the mixed query task of the Web track. In this task, there are 225 topics, which can be either topic distillation, named page finding, or homepage finding topics. The queries are created from the title of each topic. However, the system is not aware of the actual type of each query, during retrieval. This task is more similar to the operational setting of a Web search engine, which receives user queries without explicit evidence of the query type. Our aim is to use a decision mechanism for selecting an appropriate retrieval approach for each query, based on evidence from the hyperlink structure and the anchor text of the set of retrieved documents. More specifically, the decision mechanism is focused on identifying when to favour the entry points or homepages of relevant web sites.

3.1 Decision Mechanism

The decision mechanism we use employs two characteristics of the set of retrieved documents, in order to select an appropriate retrieval approach for each query.

The first characteristic is the *usefulness of the hyperlink structure*, which estimates whether there are non-random patterns of hyperlinks within the set of retrieved documents [16]. If we detect such patterns, then we assume that there are clusters of documents about the query topic. Therefore, it is preferred to favour the entry points, or the central nodes of these clusters.

We define the usefulness of the hyperlink structure as the symmetric Jensen-Shannon divergence between two different score distributions. The first one is the content analysis score distribution $S = \{s_i\}$, where s_i is the content analysis score of the document d_i from the set of retrieved documents D. In order to reduce the computational overhead, we consider only the set D^k of the top k ranked documents, according to the distribution $\{s_i\}$. We define the second distribution $U = \{u_i\}$, so as to favour the relevant documents that point to other relevant documents in D:

$$u_i = s_i + \sum_{d_i \to d_j} s_j, \quad d_i \in D^k, d_j \in D$$

where $d_i \to d_j$ denotes that there is a hyperlink from document d_i to document d_j. We normalise both distributions S and U, so that $\sum_{d_i \in D^k} s_i = \sum_{d_i \in D^k} u_i = 1$ and obtain the distributions $S_n = \{sn_i\}$ and $U_n = \{un_i\}$, respectively. The usefulness of the hyperlink structure is defined as the symmetric Jensen-Shannon divergence $L(S_n, U_n)$ between S_n and U_n, as follows:

$$L(S_n, U_n) = \sum_{d_i \in D^k} un_i \log_2 \frac{un_i}{\frac{un_i}{2} + \frac{sn_i}{2}} + \sum_{d_i \in D^k} sn_i \log_2 \frac{sn_i}{\frac{un_i}{2} + \frac{sn_i}{2}} \tag{7}$$

The second characteristic of the set of retrieved documents is a novel estimate of the number of *potential homepages* with all the query terms in the anchor text of their incoming hyperlinks. We assume that if the user submits a query, where all the terms appear in the anchor text of hyperlinks pointing to a homepage of a web site, then it is more useful to favour the homepage as the entry point for the site.

The set of potential homepages H corresponds to the documents with root, subroot, or path URL types, as defined by Westerveld et al. [23]. If we denote the anchor text terms of a document d_i by a_i, and the set of query terms by q, then the number ph_{anchor} of potential homepages with all the query terms in anchor text is defined as follows:

$$ph_{anchor} = |\{d_i | d_i \in (D \cap H) \wedge q \subseteq a_i\}| \tag{8}$$

	$ph_{anchor} \leq t_{ph}$	$ph_{anchor} > t_{ph}$
$L(S_n, U_n) \leq t_L$	case I (do not favour entry points)	case III (low confidence)
$L(S_n, U_n) > t_L$	case II (low confidence)	case IV (favour entry points)

Table 3: The decision mechanism that selects an appropriate retrieval approach for each query.

Our decision mechanism employs $L(S_n, U_n)$ and ph_{anchor}, as shown in Table 3. More specifically, if both $L(S_n, U_n)$ and ph_{anchor} are lower or equal to the thresholds t_L and t_{ph} respectively (case I), then we assume that the query is specific and we do not favour the entry points or homepages of web sites. On the other hand, if both $L(S_n, U_n)$ and ph_{anchor} are higher than the thresholds t_L and t_{ph} respectively (case IV), then we assume that it is more useful to favour the entry points or homepages of web sites, from the set of retrieved documents. For the two other cases, we cannot say with confidence whether we should favour the entry points from the set of retrieved documents. In these cases, we will decide about which approach to use, as described in Section 3.2.

3.2 Description of experiments and results

We have submitted five official runs for the mixed query task. For all submitted runs, we have indexed the .GOV test collection by removing standard stop-words and applying Porter's stemming algorithm. For the content analysis, we have used the weighting model PL2, as described in Section 2 and Table 1. The term frequency normalisation parameter was automatically set equal to c=1.28, using the approach described in Section 2.

We have used two different retrieval approaches. For the first one (CA), we extend the documents by adding the anchor text of their incoming hyperlinks, and perform content analysis with PL2. For the second approach (CAU150), we re-rank the top 150 documents retrieved with CA, using the score:

$$score_i = s_i \times \frac{1}{\log_2(urlpath\_len_i + 1)} \tag{9}$$

where s_i is the score assigned to document d_i by the approach CA, and $urlpath\_len_i$ is the length in characters of the URL path of d_i.

For both retrieval approaches CA and CAU150, the content analysis scores of documents are increased by a given percentage if the query terms appear either in the anchor text, or in the title of the documents. The percentage of the increase was set experimentally, using training data from the TREC2003 topic distillation and named page finding topics [8]. More specifically, if we apply CA and a query term t appears in the anchor text or in the title of a document, then we increase the term's weight in the document's score by 8% or 7%, respectively. If we apply CAU150 and a query term t appears in the anchor text of a document, then we increase the weight of t in the document's score by 20%.

The evaluation results of our official submitted runs for all topics, as well as for each type of topics, are shown in Table 4. The evaluation measures are the mean average precision (MAP), success at 1 retrieved document (Suc@1), success at 5 retrieved documents (Suc@5) and success at 10 retrieved documents (Suc@10). For the named page finding and homepage finding topics, average precision is equivalent to the reciprocal rank of the first relevant retrieved document, provided that there is one relevant document for the topic. The bold entries in Table 4 correspond to the run which resulted in the highest value of the respective evaluation measure.

The first two runs, uogWebCA and uogWebCAU150, correspond to our baselines, where we apply CA or CAU150 for all queries, respectively. With respect to MAP from Table 4, CA is more effective for named page finding queries, while CAU150 is more effective for topic distillation queries. Their performance is similar for homepage finding queries, while CA is more effective than CAU150 over all queries.

Run	MAP	Suc@1	Suc@5	Suc@10	MAP	Suc@1	Suc@5	Suc@10
	All topics				Topic distillation topics			
uogWebCA	0.4325	0.3733	0.6889	0.7689	0.1280	0.1733	0.5200	0.6667
uogWebCAU150	0.3478	0.3378	0.7111	**0.8444**	**0.1791**	**0.5067**	**0.7733**	**0.8933**
uogWebSelAn	**0.4576**	**0.4444**	**0.7600**	0.8178	0.1655	0.3600	0.6800	0.7733
uogWebSelL	0.3895	0.3467	0.7289	0.8089	0.1625	0.3733	0.6933	0.7867
uogWebSelAnL	0.4569	0.4267	0.7422	0.8000	0.1521	0.2933	0.6267	0.7200
	Named page finding topics				Homepage finding topics			
uogWebCA	**0.6082**	**0.4933**	**0.7867**	**0.8400**	0.5613	0.4533	0.7600	0.8000
uogWebCAU150	0.3324	0.1333	0.6133	0.8133	0.5318	0.3733	0.7467	0.8267
uogWebSelAn	0.6042	**0.4933**	**0.7867**	**0.8400**	0.6031	0.4800	**0.8133**	**0.8400**
uogWebSelL	0.4279	0.2400	0.6933	0.8000	0.5780	0.4267	0.8000	**0.8400**
uogWebSelAnL	**0.6082**	**0.4933**	**0.7867**	**0.8400**	**0.6104**	**0.4933**	**0.8133**	**0.8400**

Table 4: Evaluation of the official submitted runs to the mixed query task of the Web track.

$ph_{anchor} \leq 1$	$ph_{anchor} > 1$
apply CA	apply CAU150

Table 5: The decision mechanism used in run uogWebSelAn.

For the next three runs, we use the decision mechanism, where the thresholds are set after training with the TREC2003 topic distillation and known item topics. More specifically, in the third run, uogWebSelAn, we use only ph_{anchor}, as shown in Table 5, and apply CAU150 when there are more than $t_{ph} = 1$ potential homepages with all the query terms in the anchor text, otherwise we apply CA. From Table 4, we can see that this run results in the highest MAP, and success at 1 and 5 retrieved documents, over all queries. Moreover, it performs similarly to the baselines for the topic distillation and named page finding queries, while it outperforms both CA and CAU150 for the homepage finding queries.

The fourth run, uogWebSelL, is based on a decision mechanism that employs the usefulness of the hyperlink structure $L(S_n, U_n)$, computed from the top $k = 150$ retrieved documents (Table 6). If $L(S_n, U_n)$ is higher than the threshold $t_L = 0.26$, then we apply CAU150, otherwise we apply CA. Considering MAP from Table 4, we can see that this approach works well for the topic distillation and the homepage finding topics, but it is not equally effective for the named page finding topics. If we consider all queries, the run uogWebSelL performs similarly to the baseline uogWebCAU150.

For the fifth run, uogWebSelAnL, we select an appropriate retrieval approach based on both ph_{anchor} and $L(S_n, U_n)$, as shown in Table 7. More specifically, we apply CAU150 if $L(S_n, U_n) > 0.26$ and $ph_{anchor} > 1$, otherwise we apply CA. This run performs as well as the best one, uogWebSelAn, with respect to MAP from Table 4. In addition, it is the most effective for the homepage finding topics and equally effective as applying CA uniformly for named page finding topics.

Overall, we can see from the results in Table 4 that the selective application of different retrieval approaches is more effective than the uniform application of one retrieval approach for all queries. The decision mechanism

$L(S_n, U_n) \leq 0.26$	$L(S_n, U_n) > 0.26$
apply CA	apply CAU150

Table 6: The decision mechanism used in run uogWebSelL.

	$ph_{anchor} \leq 1$	$ph_{anchor} > 1$
$L(S_n, U_n) \leq 0.26$	apply CA	apply CA
$L(S_n, U_n) > 0.26$	apply CA	apply CAU150

Table 7: The decision mechanism used in run uogWebSelAnL.

that employs ph_{anchor} is the most effective over all queries. In addition, the decision mechanism that employs both ph_{anchor} and $L(S_n, U_n)$ performs similarly well. Moreover, it is the most effective approach for both named page and homepage finding queries. In both cases, the textual information from the anchor text is an important source of evidence for selecting an appropriate retrieval approach on a per-query basis.

4 Robust Track

In our participation in the Robust Track, we aim to test a series of techniques, including two novel pre-retrieval query performance predictors, a refined weighting function recommender (WFR) mechanism and an enhanced term frequency normalisation parameter tuning method. In the remainder of this section, we introduce these techniques in Sections 4.1, 4.2 and 4.3, respectively. We also provide the experimental setting in Section 4.4 and describe our runs in Section 4.5.

4.1 Pre-retrieval Query Performance Predictors

For the query performance prediction, we follow a pre-retrieval approach, where the prediction does not involve the use of relevance scores. We applied two newly proposed predictors, namely the average inverse collection term frequency (AvICTF) and the standard deviation of idf (σ_{idf}). Unlike the state-of-the-art predictors, such as clarity score [9] and query difficulty [3], the computation of the proposed pre-retrieval predictors does not involve the use of relevance scores. As a consequence, the cost of computing these predictors is marginal. The two applied predictors are the following:

- **Average inverse collection term frequency (AvICTF)**. Intuitively, the performance of a query can be reflected by the average quality of its composing terms. To represent the quality of a query term, instead of idf, we apply Kwok's inverse collection term frequency (ICTF). In [15], Kwok suggested that ICTF can be a good replacement for idf which indicates the quality of a query term t. In our work, we use the average of the ICTF values of the composing query terms to infer the overall quality/performance of a query:

$$AvICTF = \frac{\log_2 \prod_{t \in Q} ICTF}{ql} = \frac{\log_2 \prod_{t \in Q} \frac{token_c}{F}}{ql} \tag{10}$$

  In the above formula, F is the number of occurrences of a query term in the whole collection and $token_c$ is the number of tokens in the whole collection. ql is the number of tokens in a given query Q.

- **Standard deviation of idf (σ_{idf})**. This predictor is defined as the standard deviation of the idf of the composing query terms, where idf is given by the INQUERY's idf formula [1]:

$$idf = \frac{\log_2 (N + 0.5)/N_t}{\log_2 (N + 1)} \tag{11}$$

where N_t is the number of documents in which the query term t appears and N is the number of documents in the whole collection.

The assumption behind this predictor is that the composing terms of a poorly-performing query tend to have similar idf values. This indicates that idf fails to differentiate the informative query terms from the non-informative ones, resulting in poor performance.

According to our work in [13], σ_{idf} has significant linear and Spearman's correlations with average precision on the collection used in this track.

4.2 Weighting Function Recommender Mechanism

The weighting function recommender (WFR) mechanism refines our last year's model selection mechanism [11]. The idea of WFR is to cope with the poorly-performing queries by recommending the optimal weighting functions, including document weighting and term weighting (query expansion) functions, from a set of candidate weighting functions on a per-query basis. The mechanism follows the steps listed below:

1. Using a specific clustering algorithm, cluster a set of training queries into k clusters. The clustering process is based on the above two proposed query performance predictors, i.e. AvICTF and σ_{idf}.

2. Associate the optimal document weighting and term weighting functions to each cluster of training queries by relevance assessment (in this track, we use all the 11 document weighting functions and the 4 term weighting functions, listed in Tables 1 and 2, as the candidate weighting functions).

3. For a given new query, allocate the closest cluster to the query, and apply the associated optimal weighting functions of the allocated cluster.

For the query clustering, we adopt the CURE algorithm [10]. In the CURE algorithm, initially, each vector is an independent cluster. The similarity between two clusters is measured by the cosine similarity of the two closest vectors (having the highest cosine similarity), where the two vectors come from each cluster respectively. If we have n vectors to be processed, we start with n clusters. Then, we merge the closest pair of clusters (according to the cosine similarity measure) as a single cluster. The merging process is repeated until it results in k clusters. Here the number k of clusters is the halting criterion of the algorithm. For the scaling of the elements in a vector, we simply divide each element by the maximum value of the dimension to which the element belongs:

$$E_s = \frac{E}{E_{max}} \tag{12}$$

where E_s is the scaled value for a given element E. E_{max} is the maximum value of all the elements in the dimension that E belongs to.

4.3 Term Frequency Normalisation Parameter Tuning

As mentioned in Section 2, the term frequency normalisation parameter tuning method proposed in [12] uses a set of real queries as training queries. In our participation in this year's TREC, these training queries were obtained using a novel query simulation method that follows the steps listed below:

1. Randomly choose a seed-term from the vocabulary.

2. Rank the documents containing the seed-term using a specific document weighting function.

3. Extract the $exp\_term - 1$ most informative terms from the exp_doc top-ranked documents using a specific term weighting/query expansion function. exp_term is the required number of composing terms of the generated query. exp_doc is a parameter of the applied query expansion methodology, as described in Section 2.

4. To avoid selecting a junk term as the seed-term, we consider the most informative one of the extracted terms in step 3 as the new seed-term. Note that the original seed-term is discarded at this stage.

5. Repeat steps 2 and 3 to extract the $exp\_term - 1$ most informative terms from the exp_doc top-ranked documents, which are ranked according to the new seed-term.

6. The sampled query consists of the new seed-term and the $exp\_term - 1$ terms extracted in Step 5.

Adopting the above query simulation method, our tuning method does not involve the use of real queries.

4.4 Experimental Setting

In this track, there are 249 test topics in total. More specifically, there are 200 old topics used in last year's Robust Track and 49 new topics. Also, from the 200 old topics, 50 poorly-performing topics are chosen as the hard topics.

In our submitted runs, we experimented with three types of queries with respect to the use of different topic fields. The three types of queries are:

- **Short queries**: Only the title field is used.
- **Normal queries**: Only the description field is used.
- **Long queries**: All the three fields (title, description and narrative) are used.

All the applied document weighting and term weighting (query expansion) functions were chosen from the DFR models introduced in Section 2.

For the weighting function recommender (WFR) mechanism, all the 11 DFR document weighting functions and the 4 DFR term weighting functions, listed in Tables 1 and 2, are used as the candidate weighting functions.

For the query simulation of our term frequency normalisation parameter tuning method described in Section 4.3, we applied PL2 and Bo1 weighting functions. We simulated 200 queries to sample the document length distribution of the collection. Using our automatic tuning method, the obtained parameter settings are $c = 5.90$ for short queries, $c = 1.61$ for normal queries and $c = 1.73$ for long queries.

In all our experiments, automatic stop-word removal and Porter's stemming algorithm were applied.

Query expansion was applied in all our experiments. Using a given term weighting model, we extract the 40 most informative terms from the 10 top-ranked documents.

4.5 Description of Experiments

In the Robust Track, we submitted 10 runs. For each type of queries, the baseline corresponds to the DFR document weighting and term weighting functions from Tables 1 and 2, respectively, that resulted in the highest mean average precision for the 200 old queries. Among the submitted runs (see Table 8 for run ids and more details):

- We submitted three runs for short queries. AvICTF is applied in all these runs for query performance prediction. uogRobSBase is the baseline for short queries runs. The applied document weighting and term weighting functions are PL2 and Bo1, respectively. Compared to this baseline, uogRobSWR5 and uogRobSWR10 aim to test the weighting function recommender (WFR) mechanism. The threshold setting of WFR, i.e. the number of clusters, is set to 5 for uogRobSWR5 and 10 for uogRobSWR10.

- Our experiments for normal queries are similar. uogRobDBase is the baseline, and WFR is applied in uogRobDWR5 and uogRobDWR10 with the use of different threshold settings (i.e. 5 and 10 respectively). However, I(n)L2 and CS are chosen as the baseline weighting models. AvICTF and σ_{idf} are applied in uogRobDWR10 and the other two, respectively.

- For long queries, besides of WFR, our term frequency normalisation parameter tuning method is also tested. According to our study in [12], this method outperforms the default setting for normal and long queries, and provides comparable performance with the default setting for short queries. We compare the tuning method to the use of a default setting that is applied in uogRobLBase. Note that the tuning method is applied in all the runs except this baseline. uogRobLBase uses PL2 and Bo1, respectively. The use of the tuning method differs uogRobLT from uogRobLBase. The other two runs, uogRobLWR5 and uogRobLWR10, are again proposed to evaluate WFR.

Furthermore, in order to better evaluate our predictors, besides of the Kendall's tau, we measure the Spearman's correlation of the predictors with average precision. Moreover, since we apply query expansion in all our submitted runs, we also measure the above two correlation measures without query expansion, in order to check how query expansion affects the effectiveness of our predictors.

Run id	docW function	termW function	c	Predictor
Short Queries				
uogRobSBase	PL2	Bo1	$c = 5.90$	AvICTF
uogRobSWR5	WFR	WFR	$c = 5.90$	AvICTF
uogRobSWR10	WFR	WFR	$c = 5.90$	AvICTF
Normal Queries				
uogRobDBase	I(n)L2	CS	$c = 1.61$	σ_{idf}
uogRobDWR5	WFR	WFR	$c = 1.61$	σ_{idf}
uogRobDWR10	WFR	WFR	$c = 1.61$	AvICTF
Long Queries				
uogRobLBase	PL2	Bo1	$c = 1$	AvICTF
uogRobLT	PL2	Bo1	$c = 1.73$	σ_{idf}
uogRobLWR5	WFR	WFR	$c = 1.73$	σ_{idf}
uogRobLWR10	WFR	WFR	$c = 1.73$	AvICTF

Table 8: The submitted runs to the Robust track. Query expansion is applied for all the runs. docW function and termW function stand for the applied document weighting function and term weighting function, respectively. The applied setting of parameter c for run uogRobLBase, i.e. $c = 1$, is the default setting. *WFR* stands for the weighting function recommender mechanism.

Tables 9, 10 and 11 summarise the experiment results for short, normal and long queries, respectively. Also, Tables 12 and 13 provide the obtained Kendall's tau and Spearman's correlation of our predictors with average precision, with and without query expansion, respectively. From the results, we have the following observations:

Run id	pre@10	MAP	MAP(X)	#norel
Old queries				
uogRobSBase	.4400	.2826	.0087	32
uogRobSWR5	.4455	.2911	.0072	35
uogRobSWR10	.4605	.2961	.0097	32
New queries				
uogRobSBase	.4816	.3482	.0265	7
uogRobSWR5	.4571	.3272	.0176	8
uogRobSWR10	.4531	.3216	.0215	6
Hard queries				
uogRobSBase	.2640	.1237	.0030	14
uogRobSWR5	.2780	.1305	.0013	15
uogRobSWR10	.3160	.1360	.0025	13
All queries				
uogRobSBase	.4482	.2955	.0098	39
uogRobSWR5	.4478	.2982	.0075	43
uogRobSWR10	.4590	.3011	.0106	38

Table 9: Results of the official runs for short queries.

Run id	pre@10	MAP	MAP(X)	#norel
Old queries				
uogRobDBase	.4305	.2732	.0062	38
uogRobDWR5	.4460	.2822	.0070	31
uogRobDWR10	.4535	.2861	.0072	32
New queries				
uogRobDBase	.5510	.3888	.0259	6
uogRobDWR5	.5408	.3834	.0234	6
uogRobDWR10	.5286	.3736	.0227	6
Hard queries				
uogRobDBase	.3000	.1230	.0033	15
uogRobDWR5	.3040	.1328	.0032	10
uogRobDWR10	.2960	.1308	.0019	14
All queries				
uogRobDBase	.4542	.2959	.0070	44
uogRobDWR5	.4647	.3021	.0079	37
uogRobDWR10	.4683	.3033	.0083	38

Table 10: Results of the official runs for normal queries.

- In general, WFR achieves higher mean average precision (MAP) than the baselines for the old queries, including the hard queries, but not for the new queries. We suggest that this is due to the used scaling formula (see Equation (12)), which simply divides a given element by the maximum value in the dimension that the element belongs to. This formula implies that 0 is meaningful and does not take the actual distribution of the values (i.e. AvICTF and σ_{idf} values). Therefore, we use the following alternate formula for the data scaling:

Run id	pre@10	MAP	MAP(X)	#norel
Old queries				
uogRobLBase	.4715	.2927	.0130	31
uogRobLT	.4705	.2970	.0136	31
uogRobLWR5	.4800	.3028	.0134	26
uogRobLWR10	.4815	.3084	.0133	25
New queries				
uogRobLBase	.4939	.3586	.0325	3
uogRobLT	.5000	.3776	.0390	2
uogRobLWR5	.5122	.3703	.0388	2
uogRobLWR10	.5143	.3679	.0295	3
Hard queries				
uogRobLBase	.3100	.1609	.0150	34
uogRobLT	.3240	.1552	.0161	33
uogRobLWR5	.3180	.1608	.0158	28
uogRobLWR10	.3120	.1571	.0148	28
All queries				
uogRobLBase	.4759	.3056	.0150	34
uogRobLT	.4763	.3128	.0161	33
uogRobLWR5	.4863	.3161	.0158	28
uogRobLWR10	.4880	.3201	.0148	28

Table 11: Results of the official runs for long queries.

Run id	Predictor	τ	ρ
Short queries			
uogRobSBase	AvICTF	0.259	0.358
uogRobSWR5	AvICTF	0.257	0.365
uogRobSWR10	AvICTF	0.270	0.356
Normal queries			
uogRobDBase	σ_{idf}	0.258	0.317
uogRobDWR5	σ_{idf}	0.259	0.300
uogRobDWR10	AvICTF	0.240	0.298
Long queries			
uogRobLBase	AvICTF	0.163	0.213
uogRobLT	σ_{idf}	0.166	0.194
uogRobWR5	σ_{idf}	0.172	0.200
uogRobWR10	AvICTF	0.176	0.219

Table 12: The Kendall's tau (τ) and the Spearman's correlation (ρ) of the applied predictors with average precision for the official runs in the Robust track. Query expansion is applied in all these runs.

$$E_s = \frac{E - E_{min}}{E_{max} - E_{min}} \tag{13}$$

where E_s is the scaled value for a given element E. E_{max} and E_{min} are the maximum and minimum of all the elements in the dimension that E belongs to, respectively.

Run id	Predictor	τ	ρ
Short queries			
uogRobSBase	AvICTF	0.254	0.367
uogRobSWR5	AvICTF	0.265	0.387
uogRobSWR10	AvICTF	0.259	0.376
Normal queries			
uogRobDBase	σ_{idf}	0.233	0.338
uogRobDWR5	σ_{idf}	0.215	0.309
uogRobDWR10	AvICTF	0.240	0.319
Long queries			
uogRobLBase	AvICTF	0.193	0.282
uogRobLT	σ_{idf}	0.163	0.235
uogRobWR5	σ_{idf}	0.158	0.229
uogRobWR10	AvICTF	0.176	0.280

Table 13: The Kendall's tau (τ) and the Spearman's correlation (ρ) of the applied predictors with average precision for the official runs in the Robust Track but *without the use of query expansion*.

We refine our WFR mechanism by using the above formula for the data scaling and run additional experiments. Table 14 compares the obtained results with results obtained by the original WFR. The applied threshold setting for both the original and refined WFR mechanisms is $k = 5$. Since the original WFR with $k = 5$ results in consistently higher mean average precision than the original WFR with $k = 10$, we only apply the refined WFR for $k = 5$. As we can see from the table, the refined WFR clearly outperforms the original WFR and achieves comparable performance with the baselines. Indeed, by applying Equation (13), we improved the WFR mechanism. We have also tried using the Ward algorithm [22] and the Support Vector Regression [6] instead of the CURE algorithm. Results show that these two algorithms do not improve the WFR mechanism in terms of average precision.

- For the new queries, it is interesting to see that WFR leads to higher pre@10, but lower MAP than the baselines, when we use normal and long queries.
- Our term frequency normalisation parameter tuning method outperforms the baseline in the experiments for long queries. Compared with the baseline, i.e. uogRobLBase, uogRobLT achieves 5.30% of improvement for the new queries, and 2.36% of improvement for all the 249 queries (see Table 11).
- According to the results in Tables 12 and 13, query expansion has no significant effect on the performance prediction, except when Spearman's correlation measure is used for long queries. Our predictors work better for title and normal queries than for long queries. Moreover, in general, Spearman's correlation measures are significantly higher than Kendall's tau measures. We suggest that it is due to the fact that the Kendall's tau compares only the ranking of a list of given values, while the Spearman's correlation takes the actual values into consideration.

To summarise, for the performance prediction, the computation of our pre-retrieval predictors does not involve the use of relevance scores, and the correlations of the predictors with average precision is higher for the short and normal queries than for long queries. Moreover, we have improved the WFR mechanism by changing the data scaling formula, as shown in the additional experiments. Finally, our automatic term frequency normalisation tuning method outperforms the empirical setting and optimises the retrieval performance.

Query type	MAP_b	MAP_WFR5	MAP_r5
Short	0.3482	0.3272	0.3527
Normal	0.3888	0.3834	0.3862
Long	0.3776	0.3703	0.3712

Table 14: The mean average precision for the 49 new queries obtained by the baselines (MAP_b), by the original WFR mechanism (MAP_WFR5), and by the refined WFR mechanism MAP_r5. The applied threshold setting is $k = 5$ for both original and refined WFR mechanisms. The MAP_b and MAP_WFR5 values are taken from Tables 9, 10 and 11.

5 Terabyte Track

In the Terabyte track, we use Terrier in a distributed setting, inspired by our simulation study in [4]. We test the effectiveness of techniques such as the use of anchor text, query expansion, and the automatic parameter tuning of term frequency normalisation, for an ad-hoc retrieval task and the .GOV2 test collection. Moreover, we use a selection mechanism, which allocates the optimal document ranking and query expansion models on a per-query basis. In the remainder of this section, we describe the indexing process and our retrieval experiments.

5.1 Indexing

In order to index the .GOV2 test collection, we employ a local inverted file approach [18]. We split the collection in a number of disjoint sets of documents and index them separately. While indexing, we remove standard stop-words and apply the first step of Porter's stemming algorithm. For each disjoint set of documents, we create the following data structures:

- a *direct file* that contains all the terms of each document. The direct file is used for the query expansion models, given in Table 2.
- an *inverted file* that contains all the document identifiers, in which a term appears.
- a *lexicon* that contains the vocabulary of the indexed documents.
- a *document index* that contains information about the indexed documents.

The direct and inverted files are compressed using *gamma* encoding for the differences of term and document identifiers respectively, and unary encoding for the within-document and within-collection frequencies. The sizes of the data structures on disk are shown in Table 15. Although we index the full text of all documents, the use of compression results in great savings of disk space. More specifically, when we index the content of documents only, the total size of the data structures on disk is 17.48GB, which corresponds to 4.10% of the collection size. In the same index, the total size of the inverted files is 7.77GB, or 1.82% of the collection size. In order to apply query expansion efficiently, we also build a global lexicon for the whole collection, the size of which is 0.60GB.

Using the same indexing approach, we index the collection a second time, after adding to the documents the anchor text of the incoming hyperlinks. We have added the anchor text from 361,379,741 hyperlinks, without using the information about duplicate documents, or redirects between documents. From Table 15, we can see that the total size of the data structures on disk is 18.29GB, or 4.29% of the collection size, while the total size of the inverted files only is 8.47GB (1.99% of the collection size).

For indexing the collection, we used one AMD Athlon 1600 processor, running at 1.4GHz and one Intel Xeon processor, running at 2.8GHz. The total cumulative CPU time required for indexing the content of documents, and the content and anchor text of documents, was 12,037 minutes and 30,104 minutes, respectively. The indexing

	without anchor text	with anchor text
Total size	17.48GB	18.29GB
Inverted file size	7.77GB	8.47GB
Direct file size	7.00GB	7.70GB
Lexicon size	1.84GB	1.25GB
Document index size	0.87GB	0.87GB

Table 15: The total size on disk of the data structures (inverted files, direct files, lexicons and document indexes), with or without anchor text.

Run	Description	Query Type	MAP	pre@10	bpref	Time to retrieve 20 documents
uogTBBaseS	PL2 content retrieval	T	0.2709	0.5306	0.3026	4 sec
uogTBBaseL	PL2 content retrieval	**TDN**	0.3054	**0.6327**	0.3356	28 sec
uogTBQEL	Query expansion	**TDN**	**0.3075**	0.6163	**0.3359**	46 sec
uogTBAnchS	PL2 content and anchor text retrieval	T	0.2690	0.5245	0.3025	3 sec
uogTBPoolQEL	Weighting model selection	TDN	0.2311	0.5592	0.2589	46 sec
median			0.1427	0.4102	0.2015	

Table 16: Description and evaluation results of our official submitted runs to the Terabyte track. For running the experiments, we used 4 machines with 8 processors and 6GB of RAM in total.

time corresponds to the time required to build both the direct and the inverted files, even though the inverted file is sufficient for performing retrieval and the direct file is only used for applying query expansion. Note that the indexing time can be improved by using more processors.

5.2 Description of Experiments

For our experiments in the ad-hoc retrieval task of the Terabyte track, we have used a distributed version of Terrier. In this system, a central broker receives the queries and submits them to several independent query servers. The query servers assign scores to documents and send the partial lists of results back to the broker. The broker collects all the partial lists of results and merges them in order to create a final ranked list of retrieved documents. The scores of documents are computed using global statistics, collected by the broker from the query servers. Therefore, the results of our distributed retrieval system are equivalent to the results we would obtain if we used Terrier in a centralised setting.

We have tested both short and long queries. The short queries were created from the title field of the topics (T), while the long queries were created from all fields of the topics (TDN).

In Table 16, we present an overview and the evaluation results of our official submitted runs. We report the values of mean average precision (MAP), precision at 10 retrieved documents (pre@10) and bpref. For all five runs, the only parameter of the system, related to the term frequency normalisation, was automatically set to $c = 15.34$ for short queries and $c = 2.16$ for long queries, using the approach described in Section 2, with the sampling of queries described in Section 4.3.

Our first run, uogTBBaseS is a content-only baseline, where we employ short queries and assign scores to documents using the weighting model PL2 from the DFR framework, as described in Section 2 and Table 1. For the second run, uogTBBaseL, we use the weighting model PL2 with long queries. In the third run, uogTBQEL, we employ query expansion. More specifically, we expand the original query by adding the 20 most informative terms from the 5 top-ranked documents, using the term weighting model Bo1 from Table 2. Both runs uogTBBaseL and uogTBQEL, where we employ long queries, significantly improve the retrieval effectiveness over the baseline uogTBBaseS, where we employ short queries. The run with the highest MAP and bpref is uogTBQEL. In addition, the run uogTBBaseL, where we use long queries, achieves the highest pre@10.

In the fourth run, uogTBAnchS, we extend documents by adding the anchor text of their incoming hyperlinks, and use short queries for retrieval with PL2. Although anchor text has been shown to be very effective for topic distillation and known-item finding tasks (see Section 3), it does not affect the retrieval effectiveness for this ad-hoc task. For the last run, uogTBPoolQEL, we used a simple pooling technique to select the appropriate weighting models on a per-query basis. We consider 8 document weighting models from Table 1 (i.e. all the weighting models apart from BB2, PB2 and I(F)B2), and the 4 term weighting models from Table 2, in order to create the pool. Thus, we have $8 \times 4 = 32$ pairs of document weighting and term weighting models. For a given query, we create a pool, which contains documents retrieved among the top 15 ranks by at least 28 pairs of models. Then, we apply the weighting models that retrieve most of the documents in the pool. Compared to the baseline uogTBBaseL, the run uogTBPoolQEL does not improve retrieval effectiveness, and further investigation is needed in order to refine it.

Regarding the hardware setting, for all official submitted runs, we used 4 machines, with 8 processors and 6GB of memory in total. The configuration of the machines is the following:

- one machine with 2GB of memory and 4 Intel Xeon processors at 2.8GHz.
- one machine with 2GB of memory and 2 AMD Athlon processors at 1.4GHz.
- two machines with 1GB of memory and one Intel Pentium 4 at 2.4GHz.

All the data structures were saved on a 1.6TB RAID disk, mounted on the first machine. The time to retrieve the top 20 documents for each of the five runs is shown in Table 16. It should be stressed that a better throughput could be achieved by using more query servers, as suggested in [4].

Overall, we have found that the retrieval approaches that generally work effectively for ad-hoc retrieval from smaller collections, are still very effective for ad-hoc retrieval with the .GOV2 collection. All our official submitted runs are significantly more effective than the median. In addition, applying query expansion with long queries achieves the highest mean average precision, while our baseline that uses long queries performs similarly well.

5.3 Additional Experiments

In addition to the official submitted runs, we have performed several unofficial experiments, in order to evaluate the effectiveness of our tuning method for the term frequency normalisation parameter. We also investigate different settings for applying query expansion.

In Table 17, we present the results for different values of the parameter c for the DFR weighting model PL2. We can see that the tuning method performs very well for long queries. The estimated value $c = 2.16$ is very close to the manually best-obtained setting $c = 2$. For short queries, the tuning method estimates the value $c = 15.34$, while the manually best-obtained setting is $c = 5$. This results in a difference of 4% in mean average precision.

We have also conducted experiments using Okapi's BM25 [19] with different settings for its term frequency normalisation parameter b, including the setting obtained using our tuning approach described in Section 2. The other parameters, k_1 and k_3, are set by default to 1.2 and 1000, respectively [19]. As we can see from Table 18, the performance obtained using our tuning method is very close to the performance of the manually best-obtained setting.

Short queries (T)								
c	0.1	0.5	1	2	5	10	15.34	16
MAP	.0978	.1968	.2374	.2659	**.2822**	.2772	<u>.2709</u>	.2703
Long queries (TDN)								
c	0.1	0.5	0.8	1	2	2.16	3	4
MAP	.1390	.2530	.2812	.2903	**.3058**	<u>.3054</u>	.3017	.2951

Table 17: Results for short and long queries, using PL2 and different values of the parameter c. The value in bold is the highest mean average precision (MAP) and the underlined value is the MAP obtained using our tuning method.

b	0.2	0.34	0.40	0.50	0.65	0.75	0.80	0.90
MAP	.2660	<u>.2771</u>	**.2785**	.2764	.2626	.2478	.2362	.2098

Table 18: Results for short queries, using BM25 and different parameter settings. The value in bold is the highest mean average precision (MAP) and the underlined value is the MAP obtained using our tuning method.

Instead of Equation (6), we can employ alternate mechanisms to determine the $qtfn$ for an expanded query term. The first one is inspired by Rocchio's relevance feedback [20] and introduces the Rocchio's β to adjust the $qtfn$ in Equation (6) [2]:

$$qtfn = \beta \cdot \frac{w(t)}{w_{max}(t)} \tag{14}$$

where $w(t)$ is the weight of term t, $w_{max}(t)$ is the maximum $w(t)$ of the expanded query terms and β is a parameter. All our official runs with query expansion can be seen as having $\beta = 1.0$. We can also employ a second parameter-free mechanism of Terrier, where the $qtfn$ for the expanded query terms is set automatically according to the distribution of the expanded query terms in the top ranked documents.

Short queries (T)						
exp_{doc}	exp_{term}	$\beta = 0.1$	$\beta = 0.2$	$\beta = 0.5$	$\beta = 1.0$	parameter-free
3	10	.2807	.2826	.2919	.2532	.2772
5	20	.2842	.2822	.2991	.2901	.2660
10	40	.2857	.2826	.2935	.2808	.2680
Long queries (TDN)						
exp_{doc}	exp_{term}	$\beta = 0.1$	$\beta = 0.2$	$\beta = 0.5$	$\beta = 1.0$	parameter-free
5	20	.3145	.3180	.3168	.3075	.3175
10	40	.3206	.3257	.3195	.3024	.3210

Table 19: Results using PL2 with Bo1 query expansion model for different settings of Rocchio's β and the parameter-free query expansion of Terrier.

In Table 19, we present results using different settings for the number of expanded query terms (exp_{term}) from the exp_{doc} top returned documents, with the query expansion parameter β (see Equation (14)), and with the parameter-free mechanism of Terrier. The applied setting for parameter c is the same as that in our official runs, i.e. $c = 15.34$ for short queries and $c = 2.16$ for long queries. As shown by the results, compared with the results obtained without query expansion (see Table 17), the query expansion does improve retrieval performance, if an appropriate setting is applied.

From our additional experiments, we have seen that the automatic setting of the term frequency normalisation parameter c, without employing relevance assessments, has been particularly effective for long queries. In addition, query expansion is beneficial for both short and long queries, provided that an appropriate setting is applied.

6 Conclusions

We have participated in the Web, the Robust and the Terabyte tracks of TREC2004, using our retrieval system, Terrier, in both a centralised and a distributed setting.

In our experiments for the Web track, we have used a decision mechanism that identifies the queries for which to favour the entry points of relevant web sites and applies an appropriate retrieval approach. From our results, we can see that using the decision mechanism results in important improvements over the uniform application of one retrieval approach for all queries.

For the Robust track, we have proposed two novel pre-retrieval performance predictors. We employ these predictors in a weighting function recommender mechanism that selects the optimal weighting function for the poorly-performing queries in an effective way. Furthermore, we have employed a refined approach for automatically setting the value of the term frequency normalisation parameters, without the need of real user queries in the tuning process.

With our participation in the Terabyte track, we have evaluated the scalability of a distributed version of Terrier in handling very large test collections, such as the .GOV2. We have seen that even with very limited resources, we can use Terrier to index and experiment with .GOV2. Our results show that the retrieval methods that generally work well for ad-hoc retrieval from smaller collections, are still very effective for the larger collection .GOV2. In both our official and additional experiments, we show that using long queries and query expansion is very effective. Moreover, the automatic term frequency normalisation tuning performs very well for retrieval with long queries, without the need of relevance assessments.

Overall, we have seen that Terrier is a scalable and modular framework, which provides parameter-free baselines and it can be used effectively in a variety of different retrieval settings.

Acknowledgements

This work is funded by a UK Engineering and Physical Sciences Research Council (EPSRC) project grant, number GR/R90543/01. The project funds the development of the Terrier Information Retrieval framework (url: http://ir.dcs.gla.ac.uk/terrier).

References

[1] J. Allan, L. Ballesteros, J. Callan, and W. B. Croft. Recent experiments with INQUERY. In *NIST Special Publication 500-236: The Fourth Text REtrieval Conference (TREC-4)*, pages 49–63, Gaithersburg, MD, 1995.

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

[3] G. Amati, C. Carpineto and G. Romano. Query Difficulty, Robustness, and Selective Application of Query Expansion. In *Lecture Notes in Computing Science, Proceedings of the 26th European Conference on Information Retrieval (ECIR'04)*, pages 127–137, Sunderland, UK, 2004.

[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*, article in press, 2004.

[5] J. Callan and M. Connell. Query-based sampling of text databases. In *ACM Transactions on Information Systems (TOIS)*, Volume 19(2), pages 97 – 130, April, 2001.

[6] C. Chang and C. Lin. Training nu-support vector regression: theory and algorithms. *Neural Computation*, 14(2002), pages 1959–1977, 2002.

[7] A. Chowdhury, M. C. McCabe, D. Grossman, and O. Frieder. Document normalization revisited. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 381–382. ACM Press, 2002.

[8] N. Craswell, D. Hawking, R. Wilkinson, and M. Wu. Overview of the TREC 2003 Web Track. In *NIST Special Publication 500-255: The Twelfth Text REtrieval Conference (TREC 2003)*, pages 78–92, 2003.

[9] S. Cronen-Townsend, Y. Zhou, and W. B. Croft. Predicting query performance. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 299–306, Tampere, Finland, 2002.

[10] S. Guha, R. Rastogi, and K. Shim. CURE: an efficient clustering algorithm for large databases. In *Proceedings of the ACM SIGMOD Conference, Seatltle, WA*, pages 73–84, 1998.

[11] B. He and I. Ounis. University of Glasgow at the robust track – a query-based model selection approach for the poorly-performing topics. In *NIST Special Publication 500-255: The Twelfth Text REtrieval Conference (TREC 2003)*, pages 636–645, Gaithersburg, MD, 2003.

[12] B. He and I. Ounis. A study of parameter tuning for term frequency normalization. In *Proceedings of the 12th International Conference on Information and Knowledge Management (CIKM)*, pages 10–16. ACM Press, 2003.

[13] 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)*, pages 43–54. Springer Verlag, 2004.

[14] B. He and I. Ounis. Term Frequency Normalisation for BM25 and DFR Model. In *Lecture Notes in Computing Science, Proceedings of the 27th European Conference on Information Retrieval (ECIR'05).* Santiago de Compostela, Spain. March, 2005.

[15] K. L. Kwok. A new method of weighting query terms for ad-hoc retrieval. In *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 187–195. ACM Press, 1996.

[16] V. Plachouras and I. Ounis. Usefulness of hyperlink structure for query-biased topic distillation. In *Proceedings of the 27th Annual International SIGIR Conference on Research and Developement in Information Retrieval*, pages 448–455. ACM Press, 2004.

[17] J. M. Ponte and W. B. Croft. A language modeling approach to information retrieval. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and Development in Information Retrieval*, pages 275–281. ACM Press, 2004.

[18] B. A. Ribeiro-Neto and R. A. Barbosa. Query performance for tightly coupled distributed digital libraries. In *Proceedings of the third ACM conference on Digital libraries*, pages 182–190. ACM Press, 1998.

[19] S. Robertson, S. Walker, and M. Beaulieu. Okapi at TREC-7: Automatic Ad hoc, Filtering, VLC and Interactive. In *NIST Special Publication 500-242: The Seventh Text REtrieval Confereence (TREC-7)*, pages 253–264, Gaithersburg, MD, 1998.

[20] J. Rocchio. Relevance Feedback in Information Retrieval. In *The SMART Retrieval System–Experiments in Automatic Document Processing*, pages 313–323. Englewood Cliffs, NJ, Prentice Hall, Inc., 1971.

[21] A. Singhal, C. Buckley, and M. Mitra. Pivoted document length normalization. In *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 21–29. ACM Press, 1996.

[22] J. Ward. Hierarchical Grouping to optimize an objective function. *Journal of American Statistical Associationm*, 58(301), pages 236–244, 1963.

[23] T. Westerveld, D. Hiemstra, and W. Kraaij. Retrieving Web Pages Using Content, Links, URLs and Anchors. In *NIST Special Publication 500-250: The Tenth Text REtrieval Conference (TREC 2001)*, pages 663–672, Gaithersburg, MD, 2001.