Query Efficiency Prediction for Dynamic Pruning

Nicola Tonellotto
Information Science and Technologies Institute
National Research Council
Via G. Moruzzi 1, 56124 Pisa, Italy
nicola.tonellotto@isti.cnr.it

Craig Macdonald, Iadh Ounis
School of Computing Science
University of Glasgow
Glasgow, G12 8QQ, UK
firstname.lastname@glasgow.ac.uk

ABSTRACT

Dynamic pruning strategies are effective yet permit efficient retrieval by pruning - i.e. not fully scoring all postings of all documents matching a given query. However, the amount of pruning possible for a query can vary, resulting in queries with similar properties (query length, total numbers of postings) taking different amounts of time to retrieve search results. In this work, we investigate the causes for inefficient queries, identifying reasons such as the balance between informativeness of query terms, and the distribution of retrieval scores within the posting lists. Moreover, we note the advantages in being able to predict the efficiency of a query, and propose various query efficiency predictors. Using 10,000 queries and the TREC ClueWeb09 category B corpus for evaluation, we find that combining predictors using regression can accurately predict query response time.

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

General Terms: Performance, Experimentation

Keywords: Efficiency, Dynamic Pruning, Predictors

1. INTRODUCTION

Not all queries submitted to a search engine take the same time to complete. Ignoring transient effects such as query or operating system caching, the number of postings scored has a significant impact on the total retrieval time taken. *Dynamic pruning strategies* aim to improve efficiency by short-cutting or omitting the scoring of the postings of documents that will not be retrieved in the top k documents.

As a search engine may be processing multiple queries concurrently, having a prediction on the completion time of a query may have advantages in the scheduling of resources. For instance, if there are multiple query servers serving the same index shard, then a query can be routed to the query server that will soon be idle. Similarly, if a query is predicted to be expensive, then the search engine may reduce its effectiveness tolerance, thereby allowing increased efficiency.

Previous works in information retrieval have devised various *query performance predictors* [8, 11, 12, 14], whereby the expected *effectiveness* performance of the retrieval system when measured using relevance assessments and an evaluation measure (e.g. Mean Average Precision) are aimed to be predicted before evaluation has occurred. Various predictors have been proposed, some based on the statistics of the query terms alone (*pre-retrieval* predictors), while other *post-retrieval* predictors are calculated after the documents have been ranked [12].

However, no existing work has aimed to predict the *efficiency* performance of a search engine for a query. Hence, inspired by the previous work on query performance prediction, in this paper, we propose various novel *query efficiency predictors*, which can predict the response time of a query. The proposed predictors are inexpensive and pre-retrieval, in that they are based on various statistics of the query terms that can be conveniently calculated at indexing time.

The contributions of this paper are three-fold: we study how the efficiency of various queries can differ, and identify several reasons for inefficient queries; we propose the notion of query efficiency prediction, while also proposing several accurate predictors. Indeed, to the best of our knowledge, this is the first work on query efficiency prediction.

The remainder of this paper is structured as follows: the next section reviews the related work in terms of dynamic pruning strategies, as well as existing work in query performance predictors; Section 3 studies the problem of efficiency; Section 4 proposes and evaluates various query efficiency predictors; concluding remarks follow in Section 5. Details of the common experimental setup used throughout this paper are provided in Appendix A.

2. RELATED WORKS

In the following, we define both the efficient retrieval context of this work (Section 2.1), as well as background material on prediction within information retrieval (Section 2.2).

2.1 Efficient Retrieval

To rank documents for a query by traversing posting lists, there are two basic techniques, namely Document-at-a-time (DAAT) and Term-at-a-time (TAAT) [7]. TAAT strategies are more commonly used in traditional information retrieval systems, and perform well for small corpora. For large corpora, DAAT strategies have two advantages: they require less memory at query time, and they exploit I/O parallelism more effectively [4]. In this work, due to the size of the Web corpus that we consider, we focus exclusively on DAAT.

However, the *exhaustive* scoring of every posting of every query term can be improved upon for efficient retrieval. Indeed, to attain the typical sub-second response times of Web search engines, several strategies to enhance retrieval

LSDS-IR'11, October 28, 2011, Glasgow, Scotland, UK.

efficiency have been proposed (e.g. [2, 4, 16, 18, 19]). In particular, *dynamic pruning* strategies aim to eliminate the scoring of documents that will not make it into the final top k ranking, thereby benefiting efficiency. A *safe-up-to-rank* k (safe) strategy identifies the same top k documents and their relative ordering as an exhaustive scoring strategy.

All state-of-the-art safe dynamic pruning strategies [2, 4, 18, 19][1] aim to avoid scoring parts of the posting lists, to save disk access, decompression and score computation costs. This mechanism is implemented by maintaining additional information during retrieval, namely: a *threshold*, which is the minimum score that documents must achieve to have a chance to be present in the final top k results; and for each query term, a *term upper bound*, which is the maximal contribution of that particular term to any document score.

Buckley and Lewit [5] and Turtle and Flood [19] proposed safe dynamic pruning strategies for TAAT and DAAT scoring respectively, dealing with document-sorted indexes. The main logic behind these strategies is that the scoring of a document can be terminated early once the *pruning condition* holds, i.e. even if the document contained all yet unscored query terms, it would still not exceed the threshold and make it into the retrieved set of documents. For instance, in DAAT, since all postings for the same document are evaluated at the same time, it is possible to omit the remaining score computations once the pruning condition holds. This is the essence of the DAAT MaxScore strategy [19]: if the condition holds, all posting lists containing the current *docid* are moved on to the next docid, without the further processing of the remaining postings relative to the pruned docid. However, as at least one posting for each document must be scored, for a single term query, no pruning is possible.

In the DAAT Wand strategy [4], *skips lists* [16] are also exploited. For each document, Wand calculates a query upper bound, summing up the upper bounds for the terms occurring in the document. If the pruning condition is not satisfied, the document is fully scored. Otherwise, the next document is processed. The selection of the next document to score is optimised to facilitate skipping [4]. Wand represents the current state-of-the-art safe DAAT dynamic pruning techniques. For this reason, we only use Wand in this work.

2.2 Query Performance Prediction

The notion of predicting query difficulty refers to techniques that infer the (effectiveness) performance of a given query, without knowing the relevance assessment information. There has been some previous work regarding the query performance prediction issue. In [8], Cronen-Townsend et al. proposed inferring query performance by using a clarity score, which is the divergence of a term's query language model from its collection language model. Amati et al. [1] proposed the query difficulty notion that measures how a query term's distribution in a pseudo relevance document set diverges from randomness. In [12], He et al. proposed and studied a set of pre-retrieval query performance predictors (in the sense that their computation does not involve the use of the relevance scores or content of retrieved documents, unlike clarity or query difficulty). An example of a pre-retrieval predictor is Average inverse collection term frequency (AvICTF) [12]:

$$AvICTF = \frac{\log_2 \prod_Q \frac{token_c}{F}}{ql} \quad (1)$$

where 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 query length, which is the number of unique non-stop words in the query. The idea of ICTF is similar to the inverse document frequency (IDF): the frequency of a query term in the collection reflects its contribution in retrieval. AvICTF infers query performance by the average quality of the composing query terms of a query Q. Moreover, as all of the statistics (F, ql and $token_c$) are known before retrieval commences, there is no need to query the inverted index to compute this pre-retrieval predictor. Indeed, pre-retrieval predictors are advantageous as they permit changes in the retrieval strategy before retrieval commences [12]. In contrast, most post-retrieval predictors require the scores or contents of retrieved documents (e.g. to estimate a language model) or passes over the posting lists to estimate the number of documents matching a query, or the conditional probability of pairs of query terms.

Later, machine learning has been applied to query performance prediction [14, 15]. The recent work of Hauff [11] studies query performance prediction in detail. However, all work on query performance prediction thus far has focused upon the prediction of effectiveness, ignoring the efficiency that a query may achieve. In the following, we investigate the causes of inefficient queries, as well as proposing and evaluating several query efficiency predictors.

3. EFFICIENCY OF DYNAMIC PRUNING

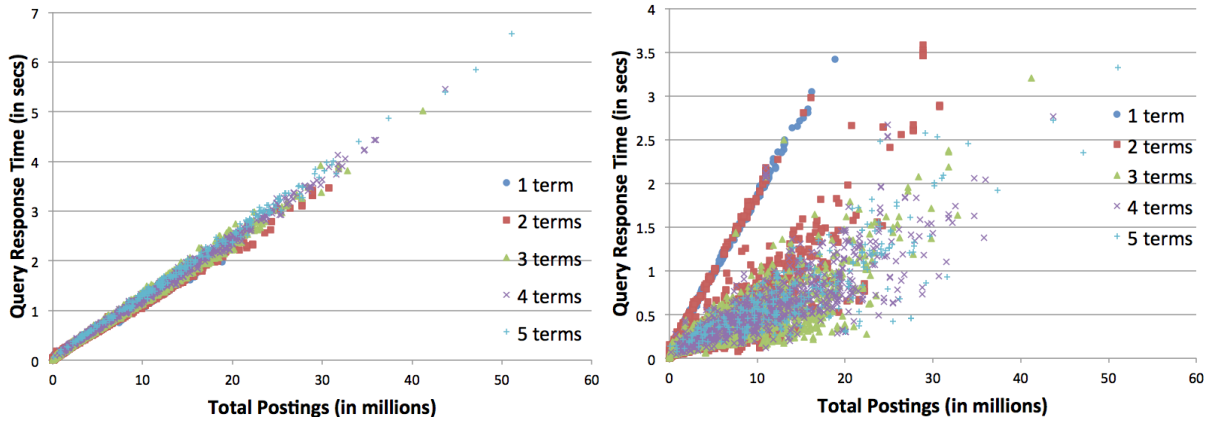
The time taken to retrieve a set of documents for a query can be broken down as follows:
1. Pre-process the query (e.g. tokenisation, stemming).
2. Lookup the statistics of each query term in the lexicon.
3. Process the postings for each query term, accumulating scores for each document to identify a final retrieved set.
4. Output the retrieved set with metadata (e.g. URLs).

For a given query, the scoring of postings (step 3) is the largest variable of the response time of a query (step 4 is a constant for k retrieved documents). Intuitively, the time taken for processing the postings for each query term is proportional to the length of the postings lists to be processed and scored, but, as we will show, this component of the response time largely depends on how much pruning occurs.

3.1 Posting Scoring

To illustrate the relationship between the total number of postings and query response time, we measure both for a stream of 10,000 queries from a real query log (see Appendix A for experimental setup). Figure 1 shows scatter plots for total postings and response time for two strategies, exhaustive DAAT and Wand. From Figure 1 (a), it is clear that the correlation between query response times and total postings is almost perfect for exhaustive DAAT, independently from the number of terms in the query. This correlation is explained by the fact that the strategy scores every posting for each query term, without any pruning. Overall, as the number of scored postings is equal to the total number of postings for each query term, Figure 1 (a) shows that the response time for exhaustive DAAT can be easily predicted before retrieval time.

[1]We omit the database-focused algorithms of Fagin et al. [10], which assume random access on the list.



(a) Exhaustive DAAT strategy. (b) WAND DAAT dynamic pruning strategy.

Figure 1: Total postings vs. Response times for two retrieval strategies.

However, in Figure 1 (b), the same correlation is not observed for the WAND dynamic pruning strategy. In particular, single term queries still exhibit a strong correlation between number of scored postings and response times. Yet for many queries with more than one term, response time can be markedly lower than other queries with the same number of postings to score. These results are in line with expectations, as clearly WAND is able to perform pruning for many queries. However, the level of pruning achieved can vary markedly, even between queries of the same length and number of postings. In the next section, we identify reasons for such variance.

3.2 Pruning

Dynamic pruning has a strong impact on the efficiency of query processing. However, there are several factors that influence how pruning is carried out at runtime. To quantify this impact, we analyse the manner that WAND prunes queries. In particular, from our 10,000 queries, we extracted the 6,351 queries with more than one term (recall that no pruning is possible for single term queries), and compute the *pruning ratio* p, defined as the percentage of posting scored by the WAND strategy and the total number of postings. A value of $p = 35\%$ means that the 35% of the total number of postings of the query were actually scored, while the remaining 65% of the postings were pruned. The pruning ratios of the 6,351 queries have been bucketed in buckets with size 10%, and the results are reported in Figure 2.

More than the 50% of the queries have a pruning ratio of less than 10%, showing the efficiency potential of WAND strategy in terms of postings pruned. We analysed these queries, and found that efficiency is higher for longer queries. Indeed, the variance in pruning behaviour for different queries can be explained by the distribution of scores within the query terms' posting lists. If high scoring documents are encountered at the beginning of the posting lists, then the pruning condition is satisfied early, and a larger number of postings can be pruned. For longer queries, the threshold rises quicker, resulting in more pruning.

However, more than 10% of the queries are not pruned at all (i.e. pruning ratio 100%). Most of these queries (571) consist of two query terms. This pruning "difficulty" has two main causes. Firstly, most of these queries have a very small total number of postings, so it is difficult to process

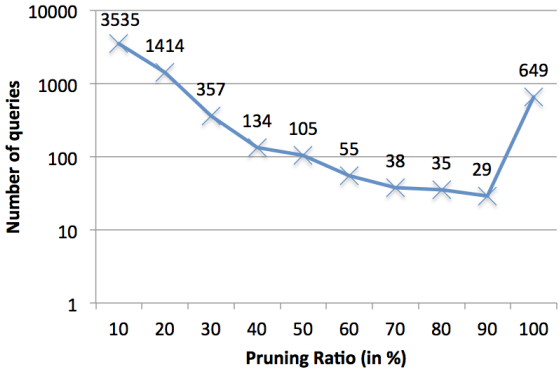


Figure 2: The bucketed pruning ratios for WAND using 6,351 queries with more than one term.

enough of them to achieve a high enough threshold to start pruning. Secondly, when the total number of postings is large, one of the two terms has a very low discriminative power, i.e. a term with a very low IDF and consequently a very low maximum score. In these cases, the strategy is forced to behave like in single term queries, where the single term is the most discriminative (high IDF) one, with the other just adding some background noise.

In summary, WAND performs a very good job at pruning postings. However, the query length and the total number of postings is not enough to predict WAND's query response time (this was also noted earlier by Baeza-Yates et al. [3] in the context of query caching). Instead, the number of documents that are actually scored is the central statistic to be predicted. To some extent, this is similar to query performance prediction, which aim to determine how many documents represent a good match for the query terms, and hence predict the effectiveness of the system for the query. Hence, inspired by this previous work in query performance prediction, we posit that various statistics of the query terms that can be obtained offline can be used to predict the response time of a query. In the following section, we propose and evaluate various statistics for predicting query efficiency.

4. PREDICTING EFFICIENCY

Having an accurate estimate of the efficiency of a query can be useful to a search engine in many ways. For instance,

in a distributed setting, if the predicted availability of query servers can be scheduled by the broker, then a new query could be sent to the query server expected to be idle soonest. Moreover, if a query is estimated to take a long time to complete, advanced ranking functionalities may be turned off to ensure that a target response time is still met. For instance, the requirement of top-k safeness may be diluted (e.g. by reducing the term upper bounds [4]), such that results that are slightly degraded in effectiveness are received in a timely manner.

To be of use, efficiency predictors must be pre-retrieval - i.e. available before retrieval has commenced, and easily obtained at retrieval time. Indeed, a post-retrieval efficiency predictor is of no use, as the response time of the query is known after retrieval has occurred! Hence, our efficiency predictors are based on the available statistics of the query terms (frequency, number of postings, IDF etc). In particular, in this work, we assume that this precludes statistics of pairs of query terms (e.g. phrases, joint probabilities), even though some search engines may record additional posting lists for some phrases [22].

In the remainder of this section, we propose many efficiency predictors (Section 4.1), which we then evaluate compared to the response times of 6,351 queries (Section 4.2), both in isolation (Section 4.3) and when combined using a machine learned regression model (Section 4.4).

4.1 Query Efficiency Predictors

All of our proposed query efficiency predictors are based on various statistics of the constituent terms of each query. Each statistic is then aggregated in different manners (e.g. sum, mean) to form different predictors. However, we rely on more than frequency counts and IDF. In particular, dynamic pruning strategies require that an upper bound be obtained on the weighting model score. This is normally identified at indexing time, by scoring the postings lists of all terms [7]. Assuming this is the case, we then have an opportunity to calculate other term-based statistics, such as the maximum, mean and standard deviation of the scores observed for each posting list, as well the number of documents that would be inserted into the top k, if there was only a single query term in that query.

For each term-based statistic, we create several query efficiency predictors by aggregating using six different statistical functions: sum, max, mean, range, variance and standard deviation. The top part of Table 1 provides the full list of term statistics that we calculate. From this table, we highlight some representative term statistics:

 # Postings: The number of postings in a term's posting list. The sum of this statistic is the number of postings scored for an exhaustive DAAT, and the upper bound on the number of postings that would be scored by WAND.

 # Maxima: A term which has fewer maxima in the score distribution may be easier to prune. This statistic can easily be obtained for each term at indexing time while the term upper bounds are being computed.

Promotions into k: If this query term was the only query term, how many documents containing this term would make it into the top k retrieved. A term with a low number of promotions probably has its highest value documents towards the start of the posting list.

Moreover, as discussed in Section 3.2, some query performance (effectiveness) predictors may also indicate efficiency, as they attempt to measure how well covered the query is in the corpus. Hence, we also consider five pre-retrieval performance predictors, as listed in second part of Table 1.

Predictor	**#**
Predictors from Term-Based Statistics	
Arithmetic, geometric, harmonic means of score	18
Max score	6
Approximation of max score	6
Variance of score	6
# Postings	6
# Maxima	6
# Maxima > avg score	6
# Postings with max score	6
# Postings with 5% of max score	6
# Postings with score within 5% of final k threshold	6
Promotions into k	6
IDF	6
Query Performance (Effectiveness) Predictors	
AvICTF [13]	1
AvIDF [13]	1
Gamma [13]	2
Similarity Collection Query [21]	1
TOTAL	89

Table 1: All tested query effectiveness predictors. Term-based statistics are aggregated into efficiency predictors using 6 different functions: sum, max, mean, range, variance and standard deviation.

While not all term-based statistics and their resulting predictors in Table 1 are likely to match query response time, it is possible that a regression technique could learn an appropriate combination. In the remainder of this section, we define the experimental setup, before evaluating the predictors in isolation, and when a combination of predictors is learned using regression.

4.2 Experimental Setup: Predicting

We address two research questions in the following experiments: (i) How accurate are our proposed predictors at predicting efficiency (Section 4.3); and (ii) Can an accurate regression model for query efficiency prediction be obtained by combining multiple predictors (Section 4.4)?

To address these research questions, we use the 6,351 queries with more one query term (we discard the few queries with more than 5 terms, while as as per Section 3.1, the prediction of the efficiency of single term queries is trivial as no pruning takes place). We split this query set in half chronologically, to form training and testing sets. The training set will be used to train the regression models in Section 4.4, while all results are reported on the testing set.

Accuracy is measured by Pearson's correlation r ($-1 \leq r \leq 1$) between the prediction and actual response times for all queries (1 is a perfect correlation). In addition, when regression is applied, we also report the root mean square errors (RMSEs) when on the test queries for regression trained on single and all predictors respectively. The RMSE measure – which should be minimised for more accurate predictions – is more suited to measuring the accuracy of predicting response times, as we are interested in obtaining a prediction of response time, rather than just the relative ranking of queries by their predicted efficiency.

4.3 Results: Single Predictors

The most effective six efficiency predictors, as well as the five query performance predictors, and their respective correlations for different query lengths are shown in Table 2. From the table, we note that the most accurate efficiency

predictors are based on the number of postings (e.g. sum # postings). Predictors based on # maxima are also accurate, mainly because they are highly correlated with those based on # postings. This is expected, as a docid sorted index typically has no trend on score distribution, so # maxima $\rightarrow \frac{1}{2}$ # entries. Query performance predictors exhibit some strong correlations, with some (e.g. $r = -0.85$) almost as strong as the proposed efficiency predictors - this illustrates that there is indeed a connection between query effectiveness prediction and efficiency prediction. Overall, while the total number of postings appears to be the most accurate, we know from Section 3 that it is not effective for all queries, hence in the next section, we determine if the prediction can be improved by combining predictors using regression.

4.4 Results: Combined Predictors

Given that a single predictor is not able to fully capture the query response time behaviour when dynamic pruning is used, we now investigate our second research question. In particular, we use a multiple linear regression approach [20] to make statistical predictions about the query response time by combining all 89 predictors from Table 1. We estimate the coefficients for the multi-linear regression of the real response times on the predictors in the train set, and we compute the predicted response times for the test set queries using these coefficients.

The correlations for the trained model on the test queries are reported in Table 3, while a detailed comparison of our statistical pre-retrieval prediction w.r.t. response time is shown in Figure 3. Additionally, Tables 2 & 3 also report the RMSE of single predictors when trained by regression.

On analysing Table 3, we observe that, for all query lengths, correlations increase and RMSEs decrease when the multi-linear regression is applied to learn a combination of predictors. Indeed, near perfect correlations as high as 0.98 are observed. This is mirrored in Figure 3, where query response time is seen to be accurately predicted across most queries. Outliers tend to occur for queries of 3 terms or longer, where the actual response time was up to 0.25 seconds longer than predicted. Overall, we conclude query efficiency can be successfully tackled as a regression problem, leading to improved pre-retrieval knowledge of the likely response time of a query.

5. CONCLUSIONS

This work performs a first investigation into predicting how long a search engine may take to process a query when dynamic pruning strategies are used. After analysing the efficiency of a state-of-the-art pruning strategy for different queries, we proposed various novel query efficiency predictors, which for a given query, can predict the amount of time taken to process the query. All of these predictors are inexpensive, in that they are based on various statistics of the query terms that can be conveniently calculated at indexing time and hence do not require retrieval to take place. We proposed a multiple linear regression approach to combine multiple predictors and obtain a more accurate query efficiency prediction in terms of response time. After training, for all query lengths, our model was able to increase correlations and reduce RMSEs between actual and predicted response time. In particular, near perfect correlations ($r = 0.98$) are obtained for two terms queries.

In the future, we will examine if the efficiency of other dynamic pruning strategies can be predicted (e.g. TAAT and DAAT MaxScore), as well as contrasting and comparing efficiency prediction between disk and memory based indices. We foresee a great potential for query efficiency prediction. For instance, we intend to run simulations to determine how distributed query traffic could be better balanced among replicated query servers if the expected termination time of the query was known.

	Correlations		RMSEs	
$\|q\|$	Best single	Regression	Best single	Regression
2	0.83	**0.98**	0.20	**0.07**
3	0.80	**0.89**	0.17	**0.13**
4	0.86	**0.93**	0.19	**0.14**
5	0.87	**0.91**	0.26	**0.22**

Table 3: Correlation and RMSEs for different query lengths on the 50% of the 6,351 non-single term queries for the best predictor in Table 2 and the regression combined predictors.

6. REFERENCES

[1] G. Amati, C. Carpineto, and G. Romano. Query difficulty, robustness, and selective application of query expansion. In *Proceedings of ECIR 2004*.

[2] V. N. Anh and A. Moffat. Impact transformation: effective and efficient web retrieval. In *Proceedings of SIGIR 2002*.

[3] R. Baeza-Yates, A. Gionis, F. P. Junqueira, V. Murdock, V. Plachouras, and F. Silvestri. Design Trade-Offs for Search Engine Caching. *Transactions on the Web*, 2(4):20–28, 2008.

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

[5] C. Buckley and A. F. Lewit. Optimization of inverted vector searches. In *Proceedings of SIGIR 1985*.

[6] N. Craswell, R. Jones, G. Dupret, and E.Viegas, editors. *Proceedings of the Web Search Click Data Workshop at WSDM 2009*.

[7] W. B. Croft, D. Metzler, and T. Strohman. *Search Engines – Information Retrieval in Practice*. Addison-Wesley, 2009.

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

[9] P. Elias. Universal codeword sets and representations of the integers. *Transactions on Information Theory*, 21(2):194 – 203, 1975.

[10] R. Fagin, A. Lotem, and M. Naor. Optimal aggregation algorithms for middleware. *Journal of Computer and System Sciences*, 66(4):614–656, 2003.

[11] C. Hauff. *Predicting the Effectiveness of Queries and Retrieval Systems*. PhD thesis, Univ. of Twente, 2010.

[12] B. He and I. Ounis. Inferring query performance using pre-retrieval predictors. In *Proceedings of the SPIRE 2004*.

[13] B. He and I. Ounis. Query performance prediction. *Information Systems*, 31(7):585–594, 2006.

[14] K. Kwok, L. Grunfeld, H. Sun, P. Deng, and N. Dinstl. TREC 2004 robust track experiments using PIRCS. In *Proceedings of TREC 2004*.

[15] S. Luo and J. Callan. Using sampled data and regression to merge search engine results. In *Proceedings of SIGIR 2002*.

[16] A. Moffat and J. Zobel. Self-indexing inverted files for fast text retrieval. *Transactions on Information Systems*, 14(4):349–379, 1996.

[17] I. Ounis, G. Amati, V. Plachouras, B. He, C. Macdonald, and C. Lioma. Terrier: A high performance and scalable information retrieval platform. In *Proceedings of the OSIR Workshop 2006*.

[18] M. Persin. Document filtering for fast ranking. In *Proceedings of SIGIR 1994*.

[19] H. Turtle and J. Flood. Query evaluation: strategies and optimizations. *Information Processing and Management*, 31(6):831–850, 1995.

Predictor	2 terms		3 terms		4 terms		5 terms	
	r	RMSE	r	RMSE	r	RMSE	r	RMSE
mean # maxima	0.82	0.20	0.80	0.17	0.86	0.19	0.86	0.27
mean # maxima > avg score	0.83	0.20	0.80	0.17	0.86	0.19	0.87	0.26
mean # postings	0.82	0.20	0.80	0.17	0.86	0.19	0.85	0.28
sum # maxima	0.83	0.20	0.80	0.17	0.86	0.20	0.86	0.27
sum # maxima > avg score	0.82	0.20	0.80	0.17	0.86	0.19	0.87	0.26
sum # postings	0.82	0.20	0.80	0.17	0.86	0.19	0.85	0.28
AvICTF	-0.61	0.36	-0.54	0.30	-0.66	0.42	-0.60	0.59
AvIDF	-0.61	0.38	-0.56	0.34	-0.67	0.48	-0.61	0.68
Gamma1	-0.16	0.40	-0.12	0.37	-0.26	0.53	-0.05	0.72
Gamma2	0.02	0.36	0.09	0.32	-0.04	0.46	0.16	0.65
SCQ	-0.72	0.37	-0.80	0.32	-0.85	0.44	-0.85	0.63

Table 2: The top 6 most accurate efficiency predictors, and all previous query performance predictors. accuracy is shown for different lengths of queries in terms of Pearson r correlation and RMSE.

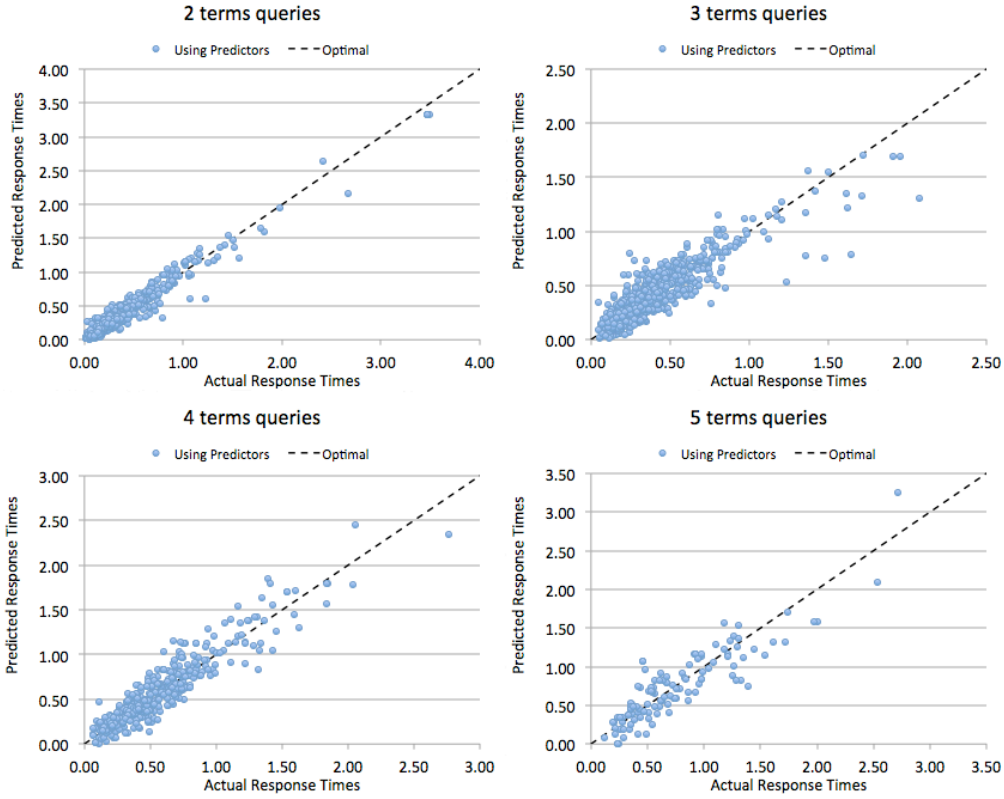


Figure 3: Predicted vs. actual response times for different query lengths.

[20] S. Weisberg. *Applied Linear Regression*. Wiley, 3rd ed., 2005.

[21] Y. Zhao, F. Scholer, and Y. Tsegay. Effective pre-retrieval query performance prediction using similarity and variability evidence. In *Proceedings of ECIR 2008*.

[22] M. Zhu, S. Shi, N. Yu, and J.-R. Wen. Can phrase indexing help to process non-phrase queries? In *Proceedings of CIKM 2008*.

APPENDIX

A. EXPERIMENTAL SETUP

All experiments are conducted using a 50 million document subset TREC ClueWeb09 corpus (known as category B). We index this corpus using the Terrier information retrieval platform [17][2], applying Porter's English stemmer, and removing standard stopwords. In the posting lists of the inverted index, docids are encoded using Elias Gamma-encoded deltas [9] and term frequencies using Elias Unary [9]. Each posting list also includes skip points [16], one every 1,000 postings. The resulting inverted index size is 22GB.

For testing retrieval efficiency, we use a port of the Terrier's retrieval infrastructure written in C++, and extract a stream of user queries from a real search engine log. In particular, we select the first 10,000 queries of the MSN 2006 query log [6], applying Porter's English stemmer and removing standard stopwords (empty queries are removed). Documents are ranked for each query using BM25, with the default parameter settings, while the number of documents retrieved is set to $k = 1,000$. All experiments are made using a dual quad-core Intel Xeon 2.6GHz, with 8GB RAM and a 2TB SATA2 disk containing the index, measuring the query response time as well as the number of postings scored.

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