Index Pruning and Result Reranking: Effects on Ad-Hoc Retrieval and Named Page Finding

(Wumpus at TREC 2006)

Stefan Büttcher Charles L. A. Clarke Peter C. K. Yeung

School of Computer Science
University of Waterloo, Canada

ABSTRACT

We describe experiments conducted for the TREC 2006 Terabyte track. Our experiments are centered around two concepts: Static index pruning (for increased retrieval efficiency) and result reranking (for improved precision).

We investigate their effect on retrieval efficiency and effectiveness, paying special attention to the difference between ad-hoc retrieval and named page finding. We show that index pruning and reranking based on relevance models can be beneficial in an ad-hoc retrieval setting, but have a disastrous repercussion on the effectiveness of named page finding. Result reranking based on anchor text, on the other hand, is very useful for named page finding, but should not be used for ad-hoc retrieval.

This dichotomy poses a problem for search engines, as there is no easy way for a search engine to decide whether a given query represents an ad-hoc retrieval task, with the purpose to satisfy an abstract information need, or a named page finding task, targeting a specific document.

1. INTRODUCTION

In the general context of text-based document retrieval, we can distinguish between at least three different types of search tasks:

- In an **ad-hoc retrieval** task, the user is interested in all documents that satisfy an abstract information need, e.g., that contain the answer to a question the user might have. All documents satisfying this information need are considered relevant, while documents that do not satisfy it are considered irrelevant.
- In a **topic distillation** task, the user is looking for a page that is representative of a certain topic, possibly a page that contains links to other pages important in the chosen context, and that gives her a good overview of what information is available on this topic. A topic distillation task can be thought of as a second-order ad-hoc retrieval task, because the user is looking for documents that give her easy access (for instance by providing hyperlinks) to documents that would be considered relevant in an ad-hoc retrieval setting.
- In a **named page finding** task, the user is interested in one specific document, or page. Only this particular page (plus possible duplicates) is relevant to the user's need, while all other documents are irrelevant. Often the simulated scenario is the situation where a user visited a web page some time ago and is trying to find back to it several days or weeks later, only remembering a few keywords. The home page finding task, a web search task in which the user is interested in a particular web home page, can be considered a special case of named page finding.

These three task types are different in nature and thus require different techniques if the search engine is to return high-quality search results to the user. This was first noticed when researchers – unsuccessfully – tried to apply link-based retrieval methods, like PageRank [12], to ad-hoc retrieval tasks [10] [9] [2] [8].

In order to apply the appropriate ranking functions to the different tasks, a search engine would need to be able to decide whether a given search query, submitted by the user, represents an ad-hoc retrieval, a topic distillation, or a named page finding task. Unfortunately, inferring the type of the search task, given a keyword query, is not always easy. For example, consider the following three queries:

- A. blue grass music festival history
- B. arizona retirement system history
- C. kalamazoo public library history

Query A (TREC 796) represents an ad-hoc topic ("Describe the history of bluegrass music and give location of bluegrass festivals."), while queries B (NP849) and C (NP1060) represent named page finding tasks, targeting specific documents. By just looking at each query, it is rather difficult to determine its type – even for a human, and much more so for a computer.

In this paper, we do not try to solve the problem of inferring the type of a search task from the given query. Instead, we look at different techniques that can be applied to the retrieval method used for any document retrieval task and analyze their effects on the different types of tasks. In particular, we study a static index pruning method and two different result reranking techniques and look at how they affect the quality of the search results in ad-hoc retrieval and in named page finding tasks.

TREC 2006, November 14–17, 2006, Gaithersburg, Maryland, USA.
This paper is available on-line at:
http://stefan.buettcher.org/papers/buettcher_trec2006.pdf

2. EVALUATION METHODOLOGY

To be able to study the effect of different retrieval methods on ad-hoc and named page finding tasks at the same time, it is convenient to have a single evaluation measure that can be applied to the search results for both task types. Unfortunately, the quality measures that are usually employed do not meet this criterion:

- In named page finding tasks, retrieval effectiveness is usually measured by the *mean reciprocal rank* (MRR) of the relevant documents in the rankings produced by the search engine. This evaluation measure gives high weight to documents ranking near the top of the search results.

- In ad-hoc retrieval tasks, on the other hand, the most prominent measure is *mean average precision* (MAP), which puts more emphasis on recall than does MRR.

While MRR and MAP are compatible in the sense that using MAP to evaluate the retrieval effectiveness in a named page finding task, with only a single relevant document, leads to the same result as MRR, doing so would be misleading. Especially when the effectiveness of result reranking techniques that mainly affect the order of the top search results is to be evaluated, changes in MAP will be dramatically different between ad-hoc retrieval and named page finding, simply because MAP does not give enough weight to top-ranking documents if the number of relevant documents for the given topic is large.

MRR, on the other hand, gives too much weight to top-ranking documents, reducing the score of a ranking from 1 to 0.5 if the named page is moved from position 1 to position 2 in the ranking produced by the search engine. Of course, this penalty is not reflected by the usefulness of the ranking to the user. In most cases, the quality difference between two such rankings will be negligible.

In our choice of the quality measure to be applied to both ad-hoc retrieval and named page finding, we are guided by view that what really matters to a user is the content of the first page of search results. For an ad-hoc retrieval task, the quality of the first page depends on the number of relevant documents it contains; thus, we choose *precision at 10 documents* (P@10) as our primary measure. For a named page finding task, the quality of the first page depends on whether it contains the named page or not, and we choose *success at 10 documents* (S@10) as our primary measure. We combine these two measures into one single measure Goodness@10 (G@10), our primary effectiveness measure:

$$G@10 = \begin{cases} \text{S@10} & \text{for named page finding} \\ \text{P@10} & \text{for ad-hoc retrieval} \end{cases} \quad (1)$$

When evaluating the two task types independently, we also look at generalizations of this measure (G@k), as well as more traditional measures, such as MAP, bpref, and MRR.

3. RETRIEVAL BASELINE

Our retrieval baseline is based on a standard document-ordered frequency index, without any positional information. Documents are ranked by Okapi BM25 [15], using Porter's algorithm [13] for term stemming. The retrieval effectiveness of this method, for both ad-hoc retrieval and named page finding, varying BM25's document length normalization parameter b, is shown in Table 1. It is interesting that for both task types the default value ($b = 0.75$) does not produce the best results. For the ad-hoc retrieval tasks, search results are best if $b \approx 0.5$ and degrade as b is increased. For named page finding, on the other hand, the best results are achieved with $b \approx 0.8$ and get worse as b gets smaller. Under other measures than our primary measure G@10, the difference for the named page finding topics is even larger: For the 2005 named page finding topics, Success@1 decreases from 0.3175 to 0.2659 when changing b from 0.9 to 0.6 (MRR decreases from 0.3862 to 0.3524).

Topics	$b = 0.60$	$b = 0.75$	$b = 0.90$
701-750 (ad-hoc '04)	0.5204	0.5041	**0.4612**
751-800 (ad-hoc '05)	**0.6280**	0.5920	**0.5620**
801-850 (ad-hoc '06)	0.5240	0.5200	**0.4840**
601-872 (NP '05)	0.5040	0.5159	0.5119
901-1081 (NP '06)	**0.4641**	0.5028	0.5249

Table 1: Varying the document length normalization parameter in Okapi BM25. Precision is measured by G@10. Bold numbers indicate statistical significance (paired t-test, $p < 0.05$) compared to the default value $b = 0.75$.

What this means is that on average a document that is relevant to one of the ad-hoc topics tends to be substantially larger than a typical named page. In fact, when analyzing the qrels file for the topics from 2005, it turns out that for the ad-hoc retrieval topics a relevant document contains 6537 tokens on average; for named page finding a relevant document only contains 2350 tokens on average. It is not clear whether this finding has any deeper meaning or whether it is just an artifact of the topic creation process.

Since the optimal values of the BM25 parameter b are so different between the two task types, and since it is not clear which task the retrieval function should be optimized for, we use the default value $b = 0.75$ throughout this paper.

Document Structure

Robertson et al. [14] proposed a method to integrate document structure by computing within-document term frequency values according to predefined weights of different fields within each document (e.g., title, anchor text). This is different from earlier methods, where information from different fields was usually fused by computing a linear combination of the individual scores.

We adjust term frequency values according to the following rules:

- `<title>`: +3
- `<h1>`, `<h2>`, `<h3>`, `<b>`, `<strong>`: +2
- `<i>`, `<em>`, `<u>`, `<dochdr>`: +1

That is, every time a query term appears in the document title, it is counted as 4 occurrences (1+3). When it appears italicized (`<i>`) and underlined (`<u>`), it is counted as 3 occurrences (1+1+1).

In contrast to Robertson et al. [14], we do not adjust the value of BM25's k_1 parameter to take the now increased average TF values into account. This is because our retrieval

Topics	G@3	G@10	MAP	MRR
701-750 (ad-hoc '04)	0.5442	0.4980	0.2373	0.7398
751-800 (ad-hoc '05)	0.6667	0.5900	0.3065	0.7895
801-850 (ad-hoc '06)	0.5200	0.4880	0.2564	0.3303
601-872 (NP '05)	0.4683	0.5794	n/a	0.4236
901-1081 (NP '06)	0.3923	0.5193	n/a	0.3528

Table 2: Effectiveness figures for the baseline retrieval method (Okapi BM25 + document structure). Parameter setting: $k_1 = 1.2$, $b = 0.75$.

framework did not give us easy access to the necessary information.

The retrieval effectiveness of this structure-aware version of BM25 on the given topic sets (ad-hoc and named page finding) is shown in Table 2. By comparing the column for G@10 in Table 2 with the numbers shown in Table 1, it can be seen that that retrieval effectiveness for the ad-hoc retrieval tasks is almost unaffected by giving additional weight to terms appearing in special fields of an HTML document; the slight decrease is probably due to our not adjusting the value of k_1 when increasing the effective TF values. Effectiveness for named page finding, however, as measured by G@10, improves significantly, from 0.5159 to 0.5794 for topics NP601-872. We therefore decided to use this variant of BM25 as the baseline for all experiments discussed in the remainder of this paper.

It should be noted that increasing TF values based on document structure gives unfair advantage to HTML documents over unstructured text, like plain text and PDF. This might be one reason why it performs so well in the named page finding tasks, because named pages tend to be HTML documents (again, this might be an artifact of the topic creation process).

Implementation Details and Efficiency Baseline

We implemented the method described above in the Wumpus[1] information retrieval system. Document scores are computed from posting lists stored in a frequency index, containing for each term a list of postings of the form:

(document ID, term frequency)

The index does not contain any positional information, as our baseline retrieval method does not make any use of proximity information anyway. Postings are encoded as integers, with the 5 least-significant bits representing the term frequency and the remaining bits representing the document ID. Term frequency values are encoded in the following way:

$$\text{enc}(tf) = \begin{cases} tf: & tf \leq 15 \\ 16 + \left\lfloor \log_{1.15}(\frac{tf}{16}) \right\rfloor: & 15 < tf < 1218 \\ 31: & tf \geq 1218 \end{cases} \quad (2)$$

For the majority of all postings, this method leads to an exact encoding of their term frequency values, while for most other postings (those with *tf* values greater than 15), a small error – less than 8% – will be introduced.

While this small inaccuracy for terms with high with-document frequency results in a minor decrease of precision (MAP, for instance, decreased by about 0.005 in our experiments), it has the advantage that queries can be processed more efficiently. By using a similar bucketing technique for document length values, the restriction to 31 possible *tf* values allows us to construct a table with precomputed score impacts and to obtain the score impact of a given (*doclen*, *tf*) pair through a simple table lookup — instead of having to perform the full BM25 score computation for every posting.

The encoded postings are compressed using a byte-aligned index compression technique [16]. This procedure results in an inverted file with a total size of 13.6 GB for GOV2. Queries are processed sequentially at a speed of 290 ms per query on average (returning the document IDs of the top 20 documents for each of the 100,000 efficiency queries).

This number, as well as all other performance figures reported in this paper, was obtained under Linux 2.6.13.3, running a 64-bit version of Wumpus on a single-core Athlon64 3500+ (2.2 GHz) with 2 GB of RAM and a 7,200-rpm SATA hard drive, accessing the index through an `ext3` file system.

4. STATIC INDEX PRUNING

The notion of *static index pruning* was officially introduced by Carmel et al. [6]. In their paper, they propose a method to limit the postings in each inverted list to those that have the greatest impact on a document's score when encountered during query processing. By applying their technique to an inverted file, it is possible to dramatically decrease its size, which will then lead to faster query processing at the cost of decrease retrieval effectiveness. By pruning more or less aggressively, the right point in this *efficiency vs. effectiveness* trade-off can be chosen.

Carmel's pruning technique is *static* because it is applied during index construction, without any knowledge about the queries that are to be processed. It is *term-centric* because each posting list is pruned independently of all the other posting lists in the inverted file.

For last year's TREC Terabyte efficiency task, we conducted experiments with a variation of Carmel's method, where a pruned index for the n most frequent terms is held in memory, while an unpruned index for all remaining terms is kept on disk [3]. By increasing the value of n (we performed experiments for $500 \leq n \leq 20000$), the number of queries for which the unpruned on-disk needs to be accessed can be decreased, which then results in lower query processing latency – again, at the cost of lower-quality search results.

More recently, we have conducted experiments with *document-centric* static index pruning [4]. In document-centric pruning, the index construction process does not select the top postings from each term (as in [6] and [3]), but the top postings from each document.

This type of pruning was first motivated by experiments with pseudo-relevance feedback, trying to reproduce the results obtained by Carpineto et al. [7]. Carpineto's method is based on the Kullback-Leibler divergence between the unigram language model defined by an individual document and the language model of the entire text collection. It uses each term's contribution to the document's KL divergence to assign feedback scores to potential expansion terms.

Given two discrete probability distributions (here: unigram language models) P and Q, their KL divergence is:

$$\text{KLD}(P, Q) = \sum_{T \in \mathcal{T}} P(T) \cdot \log \left(\frac{P(T)}{Q(T)} \right), \quad (3)$$

[1] http://www.wumpus-search.org/

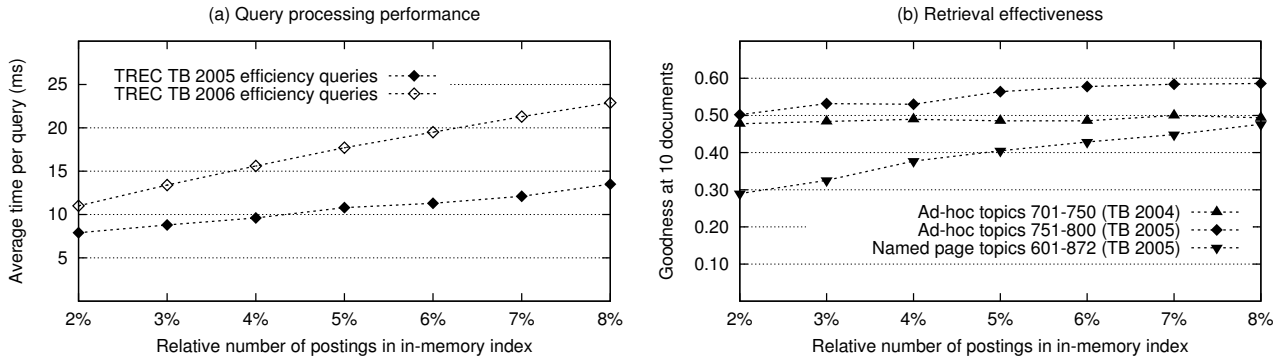

Figure 1: Impact of index pruning on query latency and retrieval effectiveness. Latency is measured for the 2005 and 2006 efficiency queries. Effectiveness is measured by mean Goodness@10 for the ad-hoc and named page finding topics from 2004, 2005, and 2006.

where $\mathcal{T}$ is the set of all terms in the vocabulary, and $P(T)$ and $Q(T)$ denote T's probability of occurrence under the distribution P and Q, respectively. Note that KLD is not a metric and in particular is not symmetric, i.e., in general we have $\mathrm{KLD}(P,Q) \neq \mathrm{KLD}(Q,P)$. However, it is non-negative, and it is zero if and only if $P = Q$. Thus, it can be understood as a measure for how far apart two term distributions are from each other.

In their feedback mechanism, Carpineto et al. select a set $\mathcal{R}$ of pseudo-relevant documents, build a language model $\mathcal{M}_R$ for each document $R \in \mathcal{R}$, and compute the feedback score of each term T appearing in $\mathcal{R}$ according to the rule

$$\mathrm{Score}_{\mathrm{FB}}(T) = \sum_{R \in \mathcal{R}} \mathcal{M}_R(T) \cdot \log\left(\frac{\mathcal{M}_R(T)}{\mathcal{M}^*(T)}\right), \qquad (4)$$

where $\mathcal{M}^*$ is the global language model of the entire text collection. That is, each term's score is the sum of its contributions to the documents' KL divergence from the global language model. The idea behind this definition of the feedback score is that a good expansion term is a term that has a high contribution to the difference between relevant documents and the rest of the collection.

When conducting some initial experiments with this feedback method, we noticed that for almost every documents considered for feedback at least one of the query terms what among the top expansion terms from the document (a more detailed analysis is given in [4]). Thus, by performing pseudo-relevance feedback on individual documents, without taking any query into account, it is possible to predict the terms for which each document would be assigned a top rank if these terms appear in a search query.

This led us to a first definition of our document-centric index pruning method. For each document D in the text collection, a unigram language model $\mathcal{M}_D$ is; each term T in the document D is assigned a score:

$$\mathrm{Score}_{\mathrm{DCP}}(T) = \mathcal{M}_D(T) \cdot \log\left(\frac{\mathcal{M}_D(T)}{\mathcal{M}^*(T)}\right), \qquad (5)$$

where $\mathcal{M}^*$ is the background language model of the text collection. All terms within a document are ranked according to their score, and the top p% are kept for each document; all other terms are discarded. Limiting the number of postings per document in this way results in a pruned index that is much smaller than the original, unpruned index. In fact, the index can be made so small that it can be completely loaded into main memory, leading to even greater gains.

Topics	Latency	G@3	G@10	MRR
701-750 (ad-hoc '04)	27.2 ms	0.5510	0.4857	0.720
751-800 (ad-hoc '05)	22.2 ms	0.6400	0.5640	0.770
801-850 (ad-hoc '06)	24.1 ms	0.4933	0.5000	0.608
601-872 (NP '05)	33.9 ms	0.2976	0.4048	0.282
901-1081 (NP '06)	28.0 ms	0.3260	0.4088	0.290

Table 3: Impact of document-centric static index pruning on average query latency and search result quality, for pruning level $p = 5$ (top 5% terms from each document). Note that latency is given for the actual ad-hoc and named page topics, not for the efficiency queries.

We performed experiments with this pruning function and found that the pruned lists for very common terms, such as "the", tend to be rather long, longer than p% of their original size, which would be suggested by the pruning criterion. We just made a applied a minor modification to the pruning function, giving less weight to the raw term frequency of a term and more weight to the difference between the term's frequency in the given document and its relative frequency in the whole collection:

$$\mathrm{Score}_{\mathrm{DCP}}(T) = \mathcal{M}_D(T)^{1-\delta} \cdot \log\left(\frac{\mathcal{M}_D(T)}{\mathcal{M}^*(T)}\right), \qquad (6)$$

For the experiments reported here, we chose $\delta = 0.15$, but our experimental results indicate that anything between 0 and 0.2 results in both decreased query latency and improved retrieval effectiveness, compared to the initial definition with $\delta = 0$.

Using the pruning criterion defined by equation 6, we built pruned indices for various pruning levels p. The size of the pruned index depends on the actual value of p. For $p = 2$, for instance (top 2% from each document), we obtain a pruned index of size 439 MB. For $p = 8$ (top 8% from each document), the size of the pruned index is 1509 MB. Note that these numbers are different from those in [4]. This is partly due to a slightly different index structure, taking document structure into account, but mainly due to a bug

Topics	P@10	P@20	MAP	bpref
701-750 (ad-hoc '04)	0.4980/0.5367/0.5531	0.4745/0.5214/0.5173	0.2373/0.2592/0.2467	0.3176/0.3359/0.3379
751-800 (ad-hoc '05)	0.5900/0.6320/0.6320	0.5440/0.5900/0.6270	0.3065/0.3361/0.3291	0.3632/0.3898/0.4133
801-850 (ad-hoc '06)	0.4880/0.5240/0.5500	0.4310/0.5030/0.5220	0.2564/0.2948/0.2791	0.3303/0.3561/0.3734

Table 4: Reranking based on relevance models and its effect on ad-hoc retrieval effectiveness. Precision values for $\rho = 0$ (left), $\rho = 1$ (middle), and $\rho = |\mathcal{Q}|$ (right).

found in the original implementation of our pruning method.

For query processing, we follow the same strategy already employed in [4]. The pruned index, loaded into memory, and the original, unpruned index, stored on disk, are used in parallel. Whenever a query term cannot not be found in the pruned in-memory index, the unpruned index has to be consulted. Of course, this leads to an increased query latency. On the other hand, it seems like the only way to guarantee that no terms are lost.

The impact of static index pruning on search efficiency (average query latency) and search result quality (G@10) is shown in Figure 4. For the ad-hoc topics, G@10 is almost unaffected by pruning the primary index. Only for $p < 6$, the decrease becomes noticeable. This is not true for named page finding. The loss of precision is huge; for $p = 5$ and the 2005 NP topics, for instance, Success@10 decreases by 30% (from 0.5794 to 0.4048), compared to our baseline. Table 4 proves that this difference is not just caused by our using different measures (S@10 vs. P@10) for named page finding and ad-hoc retrieval. Even when using the same measure (MRR), the differences are enormous. They are caused by the two task types being so fundamentally different. In ad-hoc retrieval, if, by pruning the index, a relevant document disappears from the top documents, it can simply be replaced by another relevant document. For named page finding, the document is lost, and – assuming there are no duplicate documents – cannot be replaced by another relevant document.

5. RERANKING FOR AD-HOC RETRIEVAL: RELEVANCE MODELS

Lavrenko and Croft [11] presented a retrieval method based on *relevance models* – language models derived from the top documents in an initial retrieval stage. Their method is similar to traditional query expansion by means of pseudo-relevance feedback [7], but does not require a predefined limit for the number of expansion terms, because it replaces the original query by an entire language model, without any bounds on the number of terms in it.

We present a method similar to theirs, but for performance reasons do not use the language model constructed from the top documents retrieved to perform a new ranking, but only to rerank the documents retrieved by our baseline retrieval function. The reranking is performed by computing the Kullback-Leibler divergence between the language model of each document in the initial ranking and a language model built from the top 10 documents of the initial retrieval stage.

Given an initial ranking

$$\mathcal{R} = (D_j, s_{D_j})_{1 \le j \le K}$$

($K = 1000$ in our experiments), a language model $\mathcal{M}_k^*$ is built from the top k documents (here: $k = 10$) by concatenating them and treating them as a continuous stream of terms. The language model is a mapping from each term to its probability of occurrence:

$$\mathcal{M}_k^* : \mathcal{T} \to [0, 1] \; ; \quad T \mapsto \mathcal{M}_k^*(T) = p(T),$$

where $\mathcal{T}$ is the set of all terms in the text collection. We use Porter's algorithm to group all stem-equivalent terms into equivalence classes and employ the maximum likelihood estimate (MLE) to generate $\mathcal{M}_k^*$.

Topics	$\rho = 0$	$\rho = 1$	$\rho = \|\mathcal{Q}\|$
701-750 (ad-hoc '04)	0.4980	**0.5367**	**0.5531**
751-800 (ad-hoc '05)	0.5900	**0.6320**	0.6320
801-850 (ad-hoc '06)	0.4800	**0.5240**	0.5500
601-872 (NP '05)	0.5794	0.5754	**0.3929**
901-1081 (NP '06)	0.5193	0.5193	**0.3315**

Table 5: The effect of reranking based on relevance models. Precision is measured by G@10. Bold numbers indicate statistical significance (paired t-test, $p < 0.05$) compared to $\rho = 0$.

After the language model $\mathcal{M}_k^*$ has been built from the text found in the top k documents, each document D in the initial ranking $\mathcal{R}$ is scored according to its similarity to $\mathcal{M}_k^*$. This is done by analyzing the text found inside D, building a language model $\mathcal{M}_D$ representing the document (using MLE), and computing the Kullback-Leibler divergence between the language model $\mathcal{M}_D$ and the background model $\mathcal{M}_k^*$. The definition of KL divergence has not changed since the previous section; the KLD between these two language models still is:

$$\mathrm{KLD}(\mathcal{M}_D, \mathcal{M}_k^*) = \sum_{T \in \mathcal{T}_D} \mathcal{M}_D(T) \cdot \log \left(\frac{\mathcal{M}_D(T)}{\mathcal{M}_k^*(T)} \right),$$

where $\mathcal{T}_D$ is the set of all terms within the document D; $\mathcal{M}_D(T)$ and $\mathcal{M}_k^*(T)$ are the probabilites of occurrence for the term T according to the respective language model. A smaller KL divergence means that the language model defined by document D is closer to the language model defined by the top k documents. A larger KL divergence means that it is farther away. In most cases, the KLD between the two language models is fairly small (between 0.5 and 3.0).

When all KL divergence scores have been computed, the final score for each document D in $\mathcal{R}$ is calculated according to the rule:

$$s_D^{(\mathrm{new})} := s_D - \rho \cdot \mathrm{KLD}(\mathcal{M}_D, \mathcal{M}_k^*).$$

ρ is a tuning parameter that can be used to increase or decrease the impact of the KLD score on the final ranking.

We tested two parameter configurations: $\rho = 1$ and $\rho = |\mathcal{Q}|$, where $\mathcal{Q}$ is the set of query terms that was used to

obtain the initial ranking (after stopword removal). The rationale behind $\rho = |\mathcal{Q}|$ is that the initial document scores (according to BM25) are usually higher when there are more query terms. We tried to compensate for this by increasing the weight of the KLD component in an equal fashion.

The effect of this reranking method on search quality is documented by Tables 4 and 5. Table 4 shows that the method improves precision in ad-hoc retrieval tasks substantially according to all four measures, consistently across all three ad-hoc topic sets examined, and regardless of the exact value of the reranking parameter ρ. Most of these improvements are statistically significant according to a paired t-test ($p < 0.05$). For named page finding tasks, on the other hand, reranking based on language models decreases search quality. For $\rho = |\mathcal{Q}|$, for instance, S@10 drops from 0.5794 to 0.3929 on the NP '05 topic set. For $\rho = 1$, the quality also deteriorates, but the decrease is not caught by our primary measure G@10, as the reranking weight is not big enough yes to push the named page out of the top 10 documents.

Apart from its not working for named page finding tasks, the drawback of this method is that, in our current implementation, it requires access to the full text of the K documents that are to be reranked. As all documents are stored on disk, this is very slow; it requires at least K disk seeks. Faster implementations, using document vectors in a forward index, are possible, but share the same general shortcoming.

6. RERANKING FOR NAMED PAGE FINDING: ANCHOR TEXT

For the named page finding task, the relatively simple Okapi BM25 baseline, with weighted fields to take document structure into account, already works very well for named page finding, as shown in section 3. Not surprisingly, though, it is still underperforming compared to link- or anchor-text-based retrieval functions. The best named page finding run in the TREC 2005 Terabyte track, for instance, which made use of the anchor text found in incoming links of a document, achieved an MRR of 0.463 (MRR of our baseline: 0.4236).

We addressed this issue by integrating a result reranking technique, similar to the one discussed in the previous section, that, however, is based on the anchor text of incoming hyperlinks instead of the text found in the document itself. Every link from a document D_j to document D_k whose anchor text includes some of the query terms is considered additional evidence that D_k is relevant. The strength of this evidence depends on D_j's original score and the number of query terms found in the anchor text.

More precisely, given an initial ranking

$$\mathcal{R} = (D_j, s_{D_j})_{1 \le j \le K},$$

for each document D_j among the top K search results ($K = 1000$ in our experiments) we compute its anchor score as

$$a_D = \frac{1}{\sum_{Q \in \mathcal{Q}} w_Q} \cdot \sum_{Q \in \mathcal{Q}} a_{D,Q} \cdot w_Q, \tag{7}$$

where w_Q is the IDF weight of the query term Q, and $a_{D,Q}$ is the anchor score of the term Q for the document D. The set $\mathcal{Q}$, however, is not simply the set of query terms, but is augmented by a pseudo-term Q^*. This pseudo-term, which by definition appears in the anchor text assigned with any hyperlink between two documents, is assigned the IDF weight

$$w_{Q^*} := \frac{1}{4} \sum_{Q \in \mathcal{Q}_{\text{orig}}} w_Q, \tag{8}$$

where $\mathcal{Q}_{\text{orig}}$ is the original set of query terms, without the pseudo-term. The effect of adding the pseudo-term Q^* to the query is that a link between two documents carries some weight (20% of its maximum weight), even if it does not contain any query terms. The anchor score $a_{D,Q}$ for a document D and a query term Q is then computed as follows:

$$a_{D,Q} = \frac{v_{D,Q}}{k_1 + v_{D,Q}}, \tag{9}$$

$$v_{D,Q} = \sum_{j=1}^{K} [D_j \rightarrow^Q D] \cdot \frac{s_{D_j}}{s_{D_1}} \cdot d^{j-1}, \tag{10}$$

where

$$[D_j \rightarrow^Q D] = \begin{cases} 1: & D_j \text{ uses } Q \text{ in the anchor text of a link to } D \\ 0: & \text{otherwise} \end{cases}$$

The formulation of equation 9 was motivated by BM25; for the free parameter k_1, we chose its BM25 default value: $k_1 = 1.2$. The parameter d in equation 10 is a standard damping factor that limits the impact of the "long tail" of documents on the ranking of highly ranked documents. In our experiments, we set $d := 0.99$.

The technique can be thought of as letting the documents in the initial ranking vote for each other (hence the notation $v_{D,Q}$). If a document D_j links to the document D and uses the query term Q in the anchor text that goes with this link, then we count this as a vote for D's relevance. The weight of the vote is a combination of D_j's rank and score in the initial ranking. Votes from the top-ranking document carry

Topics	$\rho = 0$	$\rho = .1$	$\rho = .2$	$\rho = .3$
701-750 (ad-hoc '04)	0.544	0.503	0.476	**0.463**
751-800 (ad-hoc '05)	0.667	**0.607**	**0.600**	**0.593**
801-850 (ad-hoc '06)	0.520	0.520	0.513	0.507
601-872 (NP '05)	0.468	**0.496**	**0.508**	**0.516**
901-1081 (NP '06)	0.392	**0.470**	**0.508**	**0.492**

Table 6: The effect of reranking based on query terms found in the anchor text of incoming links. Precision is measured by G@3. Bold numbers indicate statistical significance (paired t-test, $p < 0.05$) compared to $\rho = 0$.

Topics	MRR	S@3	S@10
NP601-872	0.424/0.459	0.468/0.508	0.579/0.611
NP901-1081	0.353/0.419	0.392/0.508	0.519/0.569

Table 7: Reranking based on anchor text and its effect on named page finding effectiveness. Precision values for $\rho = 0$ (left) and $\rho = 0.2$ (right). The improvement is statistically significant for all three measures on both topic sets.

Topics	P@10	P@20	MAP	bpref
uwmtFadTPRR	0.6100	0.5540	0.3524	0.4091
uwmtFadTPFB	0.6020	0.5570	0.3392	0.4251
uwmtFadDS	0.5180	0.4770	0.2877	0.3582
uwmtFmanual	0.7900	0.7030	0.4246	0.4785

Table 8: Official submissions for the ad-hoc retrieval task. Retrieval effectiveness for ad-hoc topics 801-850 (TREC Terabyte 2006).

Topics	S@1	S@3	S@10	S@20	MRR
uwmtFnpstr1	0.249	0.387	0.514	0.597	0.3446
uwmtFnpstr2	0.249	0.392	0.530	0.597	0.3474
uwmtFnpRR1	0.282	0.453	0.547	0.641	0.3856

Table 9: Official submissions for the named page finding task. Retrieval effectiveness for NP topics 901-1081 (TREC Terabyte 2006).

weight 1; votes from documents further down the list have a weight that is somewhat smaller.

The votes for each pair (D, Q) are scored in an Okapi-like fashion (equation 9), and these scores are combined as defined by equation 7. Once the anchor score a_D for each document D has been computed, it is combined with the original score s_D according to the following formula:

$$s_D^{(\text{new})} = s_D^{(\text{old})} * (1 + \rho \cdot a_D). \tag{11}$$

This gives us a new ranking $\mathcal{R}_{\text{new}}$. The parameter ρ in equation 11 is a tuning parameter and was set to $\rho = 0.15$ in our official runs.

Table 6 shows the results we obtained for various values of the reranking parameter ρ, starting from the anchor-text-unaware retrieval function defined in section 3 (BM25 plus document structure). For named page finding, the improvements caused by the anchor-based reranking, compared to the baseline, technique are statistically significant for all measures we looked at. With $\rho = 0.3$, our method improves Success@3 by 10% on the 2005 NP topics and by 25% on the 2006 NP topics.

For the ad-hoc retrieval topics, however, the effect is completely different. Precision decreases, and it does so quite a bit. Reranking with $\rho = 0.3$ decreases Precision@3 by 15%, 11%, and 3%, respectively, for the three different ad-hoc topic sets. Thus, we have the same situation that we had for the relevance-model-based reranking technique discussed in the previous section, only that this time an improvement is achieved for named page finding, while the search result quality for ad-hoc retrieval suffers.

Like for the reranking method based on relevance models, discussed in section 5, our current implementation of the reranking technique described above requires access to the full text of all documents involved in the reranking step.

7. OFFICIAL RUNS

We submitted runs for all three tasks of this year's Terabyte track. The runs we submitted are slightly different from the experiments described in this paper, because the system used a slightly different set of tokenization rules and

Topics	Latency	P@5	P@10	P@20
uwmtFnoprune	246.0 ms	0.5520	0.5180	0.4770
uwmtFdcp12	32.0 ms	0.5720	0.5300	0.4790
uwmtFdcp06	17.5 ms	0.5280	0.5220	0.4610
uwmtFdcp03	12.5 ms	0.5160	0.4820	0.4110

Table 10: Official submissions for the efficiency task. Retrieval effectiveness for ad-hoc topics 801-850 (TREC Terabyte 2006). Query latency for the efficiency query stream composed of 100,000 queries.

because we optimized the parameters of the retrieval function for the respective task.

Ad-hoc Retrieval

For the ad-hoc retrieval task, we submitted four runs – three automatic runs and one manual run. All three automatic runs were title-only.

uwmtFadTPFB – This is a pseudo-relevance feedback run, using the technique described by Billerbeck and Zobel [1], with 15 pseudo-relevant documents and 15 expansion terms. The initial ranking is produced by BM25TP [5] ($k_1 = 1.2$, $b = 0.5$).

uwmtFadTPRR – For this run we used the reranking technique based on relevance models described in section 5. Like for uwmtFadTPFB, the initial ranking was produced by BM25TP ($k_1 = 1.2$, $b = 0.5$).

uwmtFadDS – This run is very similar to the baseline retrieval method described in section 3, except that we set $b := 0.5$ instead of 0.75 in BM25.

uwmtFmanual – A manual run involving the assessment of 1,800 documents by a bored PhD student.

Named Page Finding

For the named page finding task, we submitted three runs. All three runs used a frequency index in which the term frequency values were updated according to the procedure discussed in section 3. The BM25 document length normalization parameter was $b = 0.75$ for all runs.

uwmtFnpstr1 – Simply the baseline method from section 3.

uwmtFnpstr2 – This run is like uwmtFnpstr1, except that, as a postprocessing step, all documents for which there is a duplicate document with higher rank are removed from the ranking. Since in named page finding, as opposed to ad-hoc retrieval, duplicate documents only count once, this procedure can be expected to result in a slight improvement over uwmtFnpstr1.

uwmtFnpsRR1 – The anchor-text-based reranking method described in section 6, with the reranking parameter set to $\rho = 0.15$.

Efficiency

For the efficiency task, we submitted four runs. All four runs were conducted on a single PC with an AMD Athlon64 3500+ CPU. Three of the four runs make use of the static index pruning method discussed in section 4. For none of the four runs, document structure was taken into account.

For all of them, BM25's document length normalization parameter was set to $b := 0.5$. In contrast to the experiments reported on in section 4, our official runs exclusively used the pruned in-memory index to produce search results and did not access the unpruned on-disk index at all. This resulted in a slightly decreased query latency (between 1 and 2 ms per query for the efficiency query streams), at the risk of missing query terms not present in the pruned index.

uwmtFnoprune – This run is very similar to the baseline method from section 3, with the exception that document structure was not taken into account and that the BM25 document length normalization parameter was set to $b := 0.5$.

uwmtFdcp03 – Document-centric pruning, as described in section 4. The top 3% of all terms in a document were taken into the pruned index.

uwmtFdcp06 – The same as uwmtFdcp03, with 6% of the terms within a document taken into the pruned index.

uwmtFdcp12 – Similar to uwmtFdcp06, with 12% of the terms within a document taken into the pruned index. The only difference is that, because the original pruned index, compressed using a byte-aligned encoding method, was too large to fit into main memory, we recompressed it using a Huffman code that treated the gaps between two consecutive document IDs and the term frequency values independently, building two separate Huffman trees. The resulting pruned index was almost 30% smaller than the byte-aligned index, but required some extra computational effort at query time, due to the more complicated compression method.

8. CONCLUSION

We have studied the effects of static index pruning and two different result reranking techniques, one based on language models, the other one based on anchor text in links between the top-ranking documents, on ad-hoc retrieval and named page finding effectiveness.

Index pruning can decrease the average query latency greatly and – if not applied too aggressively – almost does not affect search quality in ad-hoc retrieval tasks. For named page finding, however, index pruning has disastrous effects on search quality. Similarly, search result reranking based on language models created from the top documents and reranking based on the anchor text found in inter-document hyperlinks only work for one task type, while they decrease the quality of the search results when used for the other type. This makes it difficult to apply these techniques in a general-purpose search engine that is regularly confronted with both types of search tasks.

We leave the problem of designing a unified ranking function, optimizing the quality of search results in both search contexts, for future work.

9. REFERENCES

[1] Billerbeck, B., and Zobel, J. Questioning Query Expansion: An Examination of Behaviour and Parameters. In *Proceedings of the 15th Conference on Australasian Database* (2004), pp. 69–76.

[2] Buckley, C., and Walz, J. SabIR Research at TREC-9. In *Proceedings of the 9th Text REtrieval Conference* (Gaithersburg, USA, 2000).

[3] Büttcher, S., and Clarke, C. L. A. Efficiency vs. Effectiveness in Terabyte-Scale Information Retrieval. In *Proceedings of the 14th Text REtrieval Conference* (Gaithersburg, USA, November 2005).

[4] Büttcher, S., and Clarke, C. L. A. A Document-Centric Approach to Static Index Pruning in Text Retrieval Systems. In *Proceedings of the 15th ACM Conference on Information and Knowledge Management* (Arlington, USA, November 2006).

[5] Büttcher, S., Clarke, C. L. A., and Lushman, B. Term Proximity Scoring for Ad-Hoc Retrieval on Very Large Text Collections. In *Proceedings of the 29th ACM SIGIR Conference on Research and Development in Information Retrieval* (Seattle, USA, August 2006).

[6] Carmel, D., Cohen, D., Fagin, R., Farchi, E., Herscovici, M., Maarek, Y., and Soffer, A. Static Index Pruning for Information Retrieval Systems. In *Proceedings of the 24th ACM SIGIR Conference on Research and Development in Information Retrieval* (2001), pp. 43–50.

[7] Carpineto, C., de Mori, R., Romano, G., and Bigi, B. An Information-Theoretic Approach to Automatic Query Expansion. *ACM Transactions on Information Systems 19*, 1 (2001), 1–27.

[8] Gevrey, J., and Rüger, S. M. Link-Based Approaches for Text Retrieval. In *Proceedings of the 10th Text REtrieval Conference* (Gaithersburg, USA, 2001).

[9] Hawking, D. Overview of the TREC-9 Web Track. In *Proceedings of the 9th Text REtrieval Conference* (Gaithersburg, USA, September 2001).

[10] Hawking, D., Voorhees, E., Craswell, N., and Bailey, P. Overview of the TREC-8 Web Track. In *Proceedings of the 8th Text REtrieval Conference* (Gaithersburg, USA, February 2000).

[11] Lavrenko, V., and Croft, W. B. Relevance-Based Language Models. In *Proceedings of the 24th ACM SIGIR Conference on Research and Development in Information Retrieval* (New Orleans, USA, August 2001), pp. 120–127.

[12] Page, L., Brin, S., Motwani, R., and Winograd, T. The PageRank Citation Ranking: Bringing Order to the Web. Tech. rep., Stanford Digital Library Technologies Project, 1998.

[13] Porter, M. F. An Algorithm for Suffix Stripping. *Readings in Information Retrieval 14*, 3 (1980), 130–137.

[14] Robertson, S., Zaragoza, H., and Taylor, M. Simple BM25 Extension to Multiple Weighted Fields. In *Proceedings of the Thirteenth ACM Conference on Information and Knowledge Management* (New York, USA, 2004), pp. 42–49.

[15] Robertson, S. E., Walker, S., Jones, S., Hancock-Beaulieu, M., and Gatford, M. Okapi at TREC-3. In *Proceedings of the Third Text REtrieval Conference* (Gaithersburg, USA, November 1994).

[16] Scholer, F., Williams, H. E., Yiannis, J., and Zobel, J. Compression of Inverted Indexes for Fast Query Evaluation. In *Proceedings of the 25th ACM SIGIR Conference on Research and Development in Information Retrieval* (Tampere, Finland, 2002).