Lessons Learned from Indexing Close Word Pairs

Leonid Boytsov and Anna Belova
leo@boytsov.info, anna@belova.org

TREC-19, November 2010, Gaithersburg, Maryland, USA

We describe experiments with proximity-aware ranking functions that use indexing of word pairs. Our goal is to evaluate a method of "mild" pruning of proximity information, which would be appropriate for a moderately loaded retrieval system, e.g., an enterprise search engine. We create an index that includes occurrences of close word pairs, where one of the words is frequent. This allows one to efficiently restore relative positional information for all non-stop words within a certain distance. It is also possible to answer phrase queries promptly. We use two functions to evaluate relevance: a modification of a classic proximity-aware function and a logistic function that includes a linear combination of relevance features. Additionally, we use the spam scores provided by the University of Waterloo.

1. INTRODUCTION

Incorporating word proximity into relevance ranking is a well-known method to improve the retrieval effectiveness. A classic approach to computation of proximity scores involves intersection of word positional postings. Lengths of positional postings of common (non-stop) words are orders of magnitude larger than a number of documents in the collection. Because common words often appear in user queries, evaluation of proximity scores is computationally intensive.

A straightforward solution to this problem consists in indexing word pairs that occur in the documents. Yet, due to the ginormous index size it is not feasible for all but very small document collections. It is possible to build an index of manageable size by including only adjacent words [Williams et al. 1999], but it would not adequately capture proximity information. A more practical approach consists in pruning word pairs that have little potential to influence result rankings. Even though static (index time) pruning adversely affects retrieval quality, it is often justified by shorter retrieval time.

One well-known implementation of this idea consists in indexing of all pairs where the distance between words does not exceed a threshold L. Such pairs are often referred to as *close pairs*. It is commonly believed that words located far from each other are less likely to represent a relevant document. Thus, in most suggested proximity scoring functions, a contribution of a word pair in an overall score significantly decreases as the distance between words increases (typically, inversely proportional to the distance to the power of $\alpha \gtrsim 1$ [Clarke and Cormack 2000; Rasolofo and Savoy 2003; Büttcher and Clarke 2005]). Consequently, exclusion of distant pairs does not significantly affect relevance scores and associated document rankings.

Even with a limit on the distance between indexed pairs, a corresponding index would be too big. It is estimated to be more than 20 times larger than a non-positional classic inverted index with precomputed BM25 scores [Schenkel et al. 2007] (See [Robertson 2004] for a description of BM25). Schenkel et al. [2007] propose to further prune word pair postings by keeping only approximately 1000 entries with highest scores.

Such aggressive pruning is justifiable in case of a highly loaded Web search engine, but it may be excessive in case of a smaller and less loaded system, e.g., in the corporate environment. We suggest a "milder" pruning strategy that produces an index capable of restoring relative positional information for all (non-stop) words within the short distance L: in our experiments

$L = 10$. Our indexing approach is similar to the approach of Bahle et al. [2002], who proposed to index only those adjacent words where one of the words is frequent. To evaluate the effectiveness and efficiency of our approach, we participated in ad hoc web-track 2010.

2. EXPERIMENTAL SETUP

We index a smaller subset of ClueWeb09 collection that contains 50 million web-pages, mostly English. A search index is created on a 2Ghz dual-core Dell laptop running Linux. The computer has 3Gb of RAM and uses a pair of USB-connected external hard drives with bulk transfer speed of approximately 30 MByte/sec.

The indexing consists of four stages:

(1) Conversion of WARC files into text;

(2) Collecting information on word occurrences and compiling a dictionary;

(3) Creating a set of temporary inverted files;

(4) Merging inverted files.

Completion of each stage requires about one week. Indexing can be restarted from any stage.

We use a hybrid index that includes positional postings of infrequent words, non-positional postings of frequent words, as well as positional postings of real and surrogate word pairs (see Section 4). During indexing we exclude stop words (though it is not essential to the method). The remaining words are normalized: different grammatical forms of the same word are indexed as a single word. HTML structure is taken into account by repeating title words 4 times as described by Büttcher and Clarke [2005].

Because we are unaware of any software that can efficiently index close pairs on a commodity hardware, we have created a custom implementation in C++. The implemented application has been cross validated against another custom-written sequential search application.

3. RANKING FUNCTIONS

For the following discussion, we let $d = (w_1, w_2, \ldots, w_n)$ be a document containing terms w_i and $q = (q_1, q_2, \ldots, q_m)$ be a query containing terms q_i (stop words are excluded).

All our ranking functions use the following relevance features:

- The contribution of the word q_i in the overall BM25 score of the document d, which is denoted by $\text{score}_{BM25}(d, q_i)$ (see Section 3.1);
- The spam score of the document d is a percent of documents that are considered less "spammy" in comparison to d. It is denoted by SpamScore(d) and is equal to 99−CormackRank, where CormackRank is the score provided by Cormack et al. [2010].

3.1 Okapi BM25

Implemented ranking algorithms employ the classic Okapi BM25 ranking function [Robertson 2004], which is a generalization of BM11 and BM15 [Robertson and Walker 1994]. This function is also used as a comparison baseline. Given a query q, the BM25 score of the document d is computed as

$$\text{score}_{BM25}(d, q) = \sum_{1 \le i \le m} \text{score}_{BM25}(d, q_i) = \sum_{1 \le i \le m} \text{IDF}(q_i) \frac{(k_1 + 1)}{1 + K(\text{TF}(d, q_i))^{-1}}, \qquad (1)$$

$$K = k_1 \left((1-b) + b \frac{|d|}{\text{avgdl}} \right),$$

where $|d|$ is the length of the document d (in words) and avgdl is the average length of documents in a collection; b and k_1 are parameters. $\text{TF}(d, q_i)$ is a frequency of the term q_i in the document d (i.e., the number of occurrences). $\text{IDF}(q_i)$ is the inverse document frequency of the word q_i, which may be computed in several ways. We use the following non-skewed formula for IDF:

$$\text{IDF}(q_i) = \log N/n_i, \tag{2}$$

where N is a number of documents in the collection and n_i is the number of documents that contain q_i.

3.2 Pairwise Inverse Squared Distance Function

A pairwise inverse squared distance function is based on the formula proposed by Rasolofo and Savoy [2003] and modified by Büttcher and Clarke [2005], Schenkel et al. [2007]. However, we calculate a score of each pair independently.

The scoring function additionally incorporates the Okapi BM25 and is calculated as follows:

$$\text{score}(d, q) = \text{score}_{BM25}(d, q) +$$

$$+\gamma \times \sum_{1 \le i < j \le m} (k_1 + 1) \left(\frac{\min(1, \text{IDF}(w_i))}{1 + K(\text{acc}_{i,j} \text{IDF}(w_j))^{-1}} + \frac{\min(1, \text{IDF}(w_j))}{1 + K(\text{acc}_{i,j} \text{IDF}(w_i))^{-1}} \right), \tag{3}$$

where the value of $\text{acc}_{i,j}$ is equal to:

$$\sum_{|\rho - \tau| \le L, w_\rho = q_i, w_\tau = q_j} \frac{1}{(\rho - \tau)^2}.$$

The summation above is carried out through all occurrences of close word pairs (w_i, w_j) in the document d. Parameters $k_1 = 1.2$ and $b = 0.3$ have been tuned using TREC 2009 relevance assessments. The parameter γ has been set to 1.

In addition, we embed spam scores into Equation (3) through multiplying the relevance score by:

$$1 - \alpha \times \text{SpamScore}(d), \tag{4}$$

where α is an empirically determined constant equal to 0.00025.

3.3 Logistic Function

To construct a logistic relevance function we assume that there is a probabilistic relationship between the document relevance, which is a binary variable, and several independent relevance features. This relationship is often modeled using the logistic regression.

The proximity between query words q_i and q_j in the document d is represented by the relevance feature $\text{P75}(d, q_i, q_j|L)$. This feature is equal to a 75th percentile of the distribution of distances between words q_i and q_j in the document d. In that, only distances less than or equal to L (i.e., only distances between words in close pairs) are considered.

In Section 3.3.1, which can be skipped in a first read, we describe this approach in more detail. The ranking function itself is presented in Section 3.3.2.

3.3.1 *Statistical Rationale.* The logistic regression models the relationship between a binary outcome variable and independent explanatory variables x_1, x_2, …, x_m through the logistic function:

$$p(x_1, x_2, \ldots, x_m) = \frac{e^z}{e^z + 1},\ z = \beta_0 + \sum_{i=1}^{n} \beta_i x_i, \tag{5}$$

where β_i are regression coefficients. The value of the logistic function is interpreted as a probability of success in a series of Bernoulli trials. The regression coefficients are determined using the maximum likelihood method, see, e.g., a book by Maddala [1999]. The likelihood function is given by the expression:

$$\prod_k p_k^{rel_k}(1 - p_k)^{1-rel_k},$$

where rel_k is a binary relevance indicator (equal to one if the k-th trial produces a relevant document) and p_k is a probability of relevance. To compute p_k one has to "plug" values of relevance features $(x_1, x_2, \ldots, x_m)$ observed in the k-th trial into Equation (5).

For each query q used in the TREC 2009 web-track, we obtain the set of documents $D(q) = \{d_k\}$ that is an intersection of the following sets:

- the set of documents that contain all query words;
- the set of documents whose relevance to the query q was judged by assessors.

Each query is associated with a series of trials. A trial outcome associated with a document $d \in D(q)$ is successful if and only if d was judged as relevant. Depending on the properties of the query and documents in $D(q)$, these trials are modeled using different sets of explanatory variables.

Single-word queries are modeled using two explanatory variables: $\text{score}_{BM25}(d, q_1)$ and $\text{SpamScore}(d)$. Each document from $D(q)$ represents one trial. In the case of multi-word queries we essentially stack $m(m-1)/2$ series of trials each of which corresponds to a pair of query words (q_i, q_j), i.e., one document may correspond to several trials. Although trials that correspond to different pairs occurring in the same document have the same outcome (because the document is judged only once), for simplicity, we assume that these trials are independent.

Two cases are possible:

- The set of retrieved documents $D(q)$ does not contain documents where (q_i, q_j) is a close pair;
- $D(q)$ contains a document where (q_i, q_j) is a close pair.

In the first case, all documents from $D(q)$ participate in modeling. The explanatory variables include: $\text{score}_{BM25}(d, q_i)$, $\text{score}_{BM25}(d, q_j)$, and $\text{SpamScore}(d)$.

In the second case, the modeling is based on a subset of documents from $D(q)$ that contain at least one close pair (q_i, q_j). The explanatory variables include: $\text{score}_{BM25}(d, q_i)$, $\text{score}_{BM25}(d, q_j)$, $\text{SpamScore}(d)$, and an additional variable $\text{P75}(d, q_i, q_j|L)$.

3.3.2 *Description of Logistic Ranking Function.* An argument of the logistic function is a linear combination of relevance features. For one-word queries, the only features are the BM25 score and the spam score. For queries containing two or more words, the values of the logistic function are summed up across all query pairs:

$$\text{score}(d, q) = \sum_{1 \le i < j \le m} [(q_i, q_j) \text{is a close pair in } d] + \frac{e^{f(d,q_i,q_j)}}{1 + e^{f(d,q_i,q_j)}} \tag{6}$$

The first term under the summation is equal to 1 if and only if (q_i, q_j) is a close pair in the document d. Otherwise, this term is zero.

If (q_i, q_j) is a close pair, $f(d, q_i, q_j)$ is a linear combination of the document spam score, the sum of BM25 scores for q_i and q_j in d, as well as of the 75th percentile of distances between q_i and q_j in d (when those distances do not exceed L):

$$f(d, q_i, q_j) = \beta_0 + \beta_1 \left(\text{score}_{BM25}(d, q_i) + \text{score}_{BM25}(d, q_j)\right) +$$

$$+\beta_2 \text{P75}(d, q_i, q_j | L) + \beta_3 \text{SpamScore}(d)$$

If (q_i, q_j) is not a close pair, then the function is represented by a linear combination with different values for parameters β_i. Because the document d does not contain (q_i, q_j) as a close pair, this linear combination omits the 75th percentile.

Initially, we attempted to estimate parameters β_i through the logistic regression (see Section 3.3.1), using TREC 2009 relevance rankings as a training dataset. To this end, we chose parameters that optimized a likelihood function for each query separately. Then we used the Monte-Carlo method to combine query-specific estimates. The results are depicted in Figure 1.

Although this approach has not produced an effective retrieval method, we have learned that existence of a close pair in a document was a strong and statistically significant predictor of relevance. In particular, whenever a set of documents retrieved for a specific multi-word query contains documents with and without close pairs (q_i, q_j), the documents without close pairs are almost never relevant. This is the rationale for including an indicator of a close pair in Equation (6). In addition, we have found the following associations (see Figure 1):

- a positive correlation between the document relevance and BM25 scores of query words;
- a negative correlation between the document relevance and the distances between query words in the document;
- a negative correlation between the document relevance and the document spam score.

Therefore, one may expect that all three relevance features (BM25 scores of query words, document spam scores, and distances between query words) should be predictors of the document relevance. However, the experiments have shown that word proximity information has only little effect on the retrieval effectiveness.

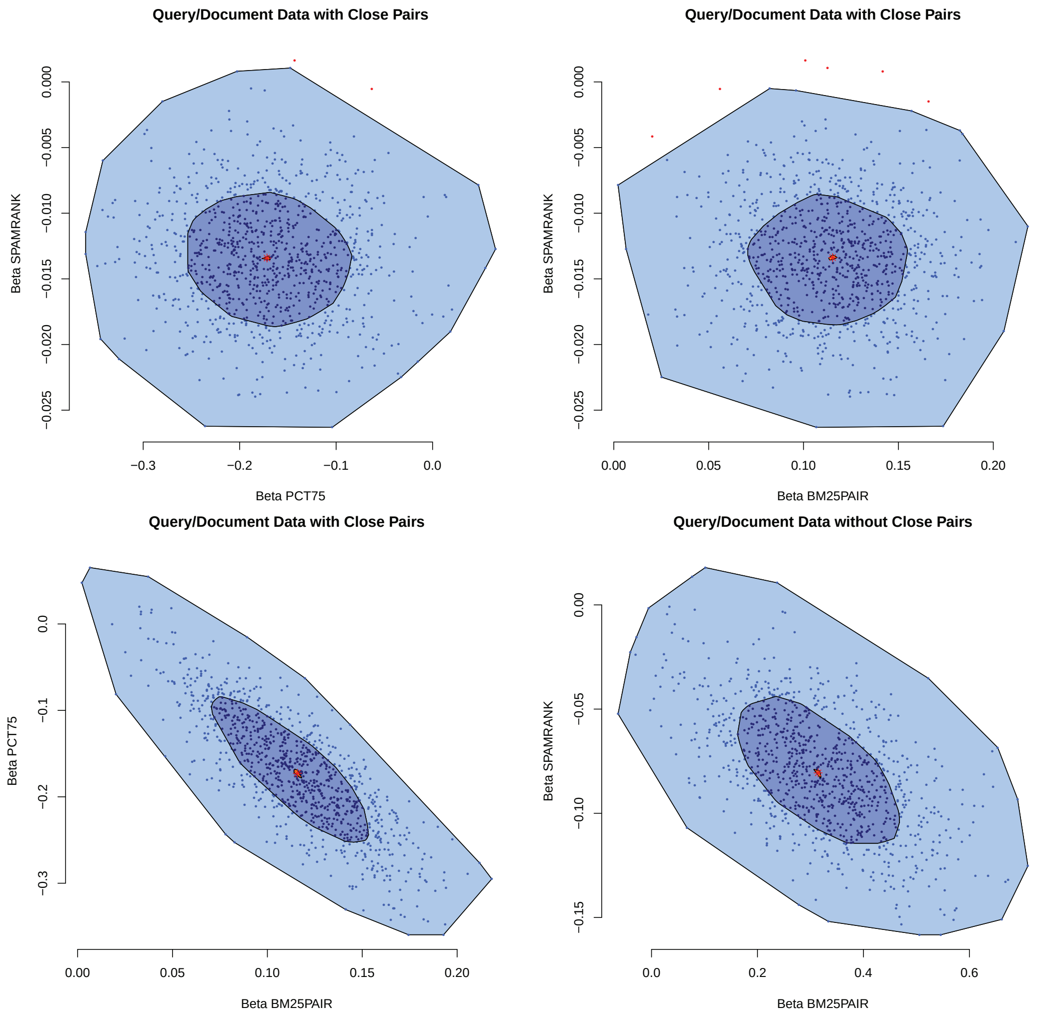
In order to obtain a competitive relevance function, we employ direct optimization using the method by Nelder and Mead [1965]. An objective function is the mean average precision, which is computed using the standard TREC utility trec_eval.

4. INDEXING APPROACH

This simple folklore approach should have been described in the literature, but we are unaware of exact references. It is similar to the method by Bahle et al. [2002] and is inspired by the following assumptions and observations:

- If we choose only several hundred most frequent words, the number of pairs where both words are frequent would not be excessively large. The set of such word pairs would be small and could be easily kept in RAM;
- All other (i.e., infrequent) words would have relatively short posting lists. Therefore, we can keep full positional information for these words;
- One-word queries do not require positional information.

Fig. 1: Averaging parameters obtained through logistic regression using the Monte-Carlo method.



Notes: Beta BM25Pair represents score$_{BM25}(d, q_i)$+score$_{BM25}(d, q_j)$ for some not necessarily close pair (q_i, q_j).

Therefore, proximity queries can be efficiently satisfied if the following indices are created:

- A non-positional index of frequent words;
- A positional index of the remaining (infrequent) words;
- A positional index of pairs occurring within distance $L = 10$, where both words are frequent;
- A positional index of pairs occurring within distance $L = 10$, where only one word is frequent.

The indices of the first three types are implemented as classic inverted files with a dictionary of unique entries and posting lists that encode entry occurrences. To decrease space requirements and improve retrieval time, postings are encoded using *byte*-aligned codes of variable length [Williams and Zobel 1999].

For the index of the fourth type, i.e., the index of pairs containing an infrequent word w_j, a straightforward inversion of quadruplets in the form $(w_i, w_j, \text{pos}_j, L_{i,j})$, where w_i is a frequent word and $L_{i,j}$ is the distance between w_i and w_j, may be infeasible because:

- It is hard to manage a dictionary containing hundred million (if not billion) word pairs (w_i, w_j). For instance, it cannot be loaded into RAM;
- The index would contain many short postings, which have poor compression ratio.

Therefore, we opt for indexing surrogate pairs. This method consists in dividing all infrequent words into a small number of groups with the help of a hash function $h(w)$. Then, we create an inverted file of the following quadruplets: $(w_i, h(w_j), \text{pos}_j, L_{i,j})$. Note that such quadruplets should always include a position of an *infrequent* word. In our experiments, the number of frequent words is 500 and the number of groups containing infrequent words is 100.

Our current implementation ignores stop words, but it is also possible to index them. Had we decided to index "stop pairs" (i.e., word pairs where at least one word is a stop word) we would have used a specific distance threshold, which is much smaller than L. Perhaps, a good idea is to choose $L = 1$, i.e., to include only those stop pairs where words are adjacent. In addition, we would not create a non-positional index of stop words.

During retrieval time, we compute the BM25 score and adjust it using positional information. Consider for example a pair of query words q_1 and q_2. If, both words are infrequent, we retrieve their (full) positional posting lists and create an intersection. If, both words are frequent, we retrieve a positional posting for the pair (q_1, q_2), which was precomputed at the indexing step.

If only q_1 is frequent, we retrieve a positional posting of the surrogate pair $(q_1, h(q_2))$. Because several real pairs are mapped into a single surrogate pair, this posting may contain spurious entries that do not represent occurrences of the pair (q_1, q_2). To eliminate these excess entries, we need to retrieve the full positional posting of the infrequent word q_2 and intersect it with the posting of the surrogate pair $(q_1, h(q_2))$.

It can be seen that, unlike the method by Schenkel et al. [2007], this method can answer any phrase query (if stop pairs are indexed). It may also answer queries in the form "find documents containing q_1 and q_2 that occur within a user-specified distance", but full recall is not guaranteed.

5. DISCUSSION

We have submitted two official runs: *blv79y00shnk* and *blv79y00prob*. The respective ranking functions are described in Sections 3.2 and 3.3. Additionally, we have evaluated several other methods using TREC 2010 queries and relevance judgements. The goal of this evaluation is to investigate how spam scores and proximity scores influence the retrieval effectiveness.

Performance measures of the retrieval effectiveness are computed using the standard TREC utilities trec_eval and gdeval. These measures include: the mean average precision, the precision at 5 and 15 (P@5 and P@15), the normalized discounted cumulative gain at 20 (NDCG@20) and the expected reciprocal rank at 20 (ERR@20) [Chapelle et al. 2009]. For some of the methods, we measure retrieval time and time to calculate proximity scores (if applicable). The results are presented in Table I, p. 8.

5.1 Retrieval Effectiveness

We have evaluated the following additional methods (which represent unofficial runs):

- *BM25* is a baseline method that evaluates the standard BM25 score with $k_1 = 1.2$, $b = 0.3$, and the non-skewed formula for IDF, see Equation (2).
- *BM25 (+close pairs)* is the extended BM25 method, where scores of documents that contain at least one close pair are increased by some large constant. As a result, all documents without close pairs are ranked lower than documents with close pairs.
- *BM25 (+spamrank)* is the BM25 method extended with spam scores. To this end, each relevance score is multiplied by a factor given by Equation (4).
- *BM25 (+close pairs +spamrank)* is a modification of the spamrank-enhanced BM25, i.e, *BM25 (+spamrank)*, where all documents with close pairs are ranked higher than documents without close pairs.
- *blv79y00shnk* $\gamma = 0.3$ is a variant of *blv79y00shnk* that uses the proximity weight equal to 0.3;
- *blv79y00shnk (-spamrank)* $\gamma = 1$ is a variant of *blv79y00shnk* that does not use spam scores. The proximity weight is equal to 1;
- *blv79y00shnk (-spamrank)* $\gamma = 0.3$ is a variant of *blv79y00shnk* that does not use spam scores. The proximity weight is equal to 0.3;

Note that we evaluate methods *blv79y00shnk* and *blv79y00shnk (-spamank)* using three different proximity scores γ, see Equation (3). In Table I we present results only for $\gamma = 1$ and $\gamma = 0.3$: $\gamma = 1$ was used in official runs and $\gamma = 0.3$ yielded the best results for 2010 queries.

Table I shows that the best official run is *blv79y00shnk*, which outperforms the baseline (*BM25*) by 10-50 percent according to all measures. It can be seen that this improvement is largely due to using spam scores. Had not we used the spam scores, *blv79y00shnk* with $\gamma = 1$ would have been worse than *BM25* with respect to both ERR@20 and the mean average precision. Yet, it would have had a slightly better values of P@5, P@15, and NDCG@20. Furthermore, the unofficial run *BM25 (+spamrank)* outperforms the official run *blv79y00shnk* with respect to all measures.

In addition to the pairwise inversed square distance proximity function, we have evaluated a simpler approach in which proximity is taken into account by sorting documents with close pairs higher than documents without close pairs. It is interesting that the use of close pairs

Table I: Method performance.

Run name	Retrieval time (sec)	ERR@20	NDCG@20	Mean average precision	P@5	P@15
		TREC 2010				
blv79y00shnk **(official run)** $\gamma = 1$	1.72/0.48	0.10935	0.18772	0.1334	0.3500	0.3444
blv79y00shnk (-spamrank) $\gamma = 1$		0.06413	0.12876	0.1079	0.2042	0.2417
blv7900shnk $\gamma = 0.3$		0.11332	0.19829	**0.1371**	0.3583	0.3403
blv7900shnk (-spamrank) $\gamma = 0.3$		0.06969	0.13743	0.1116	0.2083	0.2458
blv79y00prob **(official run)**	2.42/1.17	0.09563	0.16256	0.1224	0.3375	0.3181
BM25	1.25	0.06532	0.12682	0.1100	0.2000	0.2333
BM25 (+close pairs)	1.53/0.28	0.06413	0.12876	0.1077	0.2042	0.2417
BM25 (+spamrank)		0.11229	0.18894	0.1365	0.3625	0.3486
BM25 (+close pairs +spamrank)		**0.12265**	**0.21086**	0.1340	**0.4083**	**0.3639**

has not improved *BM25*. However, the method *BM25 (+close pairs +spamrank)* that combines BM25 scores, spam scores, and close pairs has the best performance overall.

Note that the proximity weight $\gamma = 1$ is not optimal: *blv79y00shnk (-spamrank)* $\gamma = 0.3$ outperforms *BM25* according to all measures. The values of P@5 and P@15 of *BM25* are also somewhat worse than those of *BM25 (close pairs)*. However, none of the proximity-aware ranking functions without spam scores is better than *BM25* by more than 10 percent: in most cases the improvement is much smaller. This result is consistent with evidence reported by other researchers [Büttcher and Clarke 2005; Tao and Zhai 2007; Schenkel et al. 2007].

5.2 Efficiency

Two most important efficiency benchmarks are retrieval time and index size. Because we index occurrences of numerous word pairs, the total size of the index is about 284 percent of an original text collection size after conversion from HTML (see Table II). The index of pairs itself uses space equal to 260 percent of the original text collection size.

Even though the size of the index is 5-10 times the size of a state-of-the-art compressed positional index (without stop words), this is still a tolerable overhead. It is an order of magnitude smaller than the estimate by Schenkel et al. [2007]. Furthermore, because word postings do not contain positional information for frequent words, they require less space and can use the existing memory better.

Table II: Inverted Index Statistics

	Size (GB)
Source text data (HTML is stripped off)	250
Dictionary	2
Infrequent word postings	50
Frequent word postings	9
Pair postings	646

Note that evaluation of proximity scores in our implementation takes less time than computation of BM25 scores (see Table I). Preliminary estimates show that our mild pruning method should be at least twice as fast as a method that uses a complete positional index. Yet, thorough experiments are needed to verify our calculations. It is also noteworthy that a simpler proximity evaluation method: *BM25 (+close pairs)*, is even more efficient. In comparison to *blv79y00shnk*, it requires about half of the time to compute proximity scores.

5.3 Failure Analysis

5.3.1 *Statistical Learning.* The logistic-function based method has a good retrieval effectiveness with appropriate parameters β_i (see Table I, row *blv79y00prob*). However, these parameters have been obtained through direct optimization of the mean average precision and not through statistical analysis. This may have happened because:

- For some queries, the information contained in the relevance judgments is not sufficient to make adequate inference;
- Statistical methods may not work well for indicators, such as the mean average precision, because they optimize a different objective function.

5.3.2 *Queries that Performed Poorly.* A quick review of the results showed that there were several queries that had zero value for ERR@20.

- In case of topics 70 (“`to be or not to be the question is`”), 72 (“`the sun`”) and 92 (“`the wall`”), no relevant documents have been found. The obvious reason: we do not index stop words.

- In case of queries 74 (“`kiwi`”) and 94 (“`titan`”), at least half of all relevant documents have been identified, but not ranked high enough. The exact reasons are yet to be determined.

5.3.3 *Choice of Optimal Parameters.* We have not properly tuned the weight of the proximity score γ. Had we used $\gamma \approx 0.3$, *blv79y00shnk* would have improved by 2-3 percent with respect to all performance measures except P@15.

5.4 Conclusions and Future Work

We have presented results of experiments with proximity-aware ranking functions that use close word pair statistics and spam scores. Both spam scores and proximity-aware ranking improve the retrieval effectiveness, however, use of spam scores is the major contributor to this improvement.

We have also learned that incorporating proximity into a ranking formula does not improve the search quality in all cases. For most queries, taking proximity into account leads only to small improvement of the effectiveness. To achieve this improvement, parameters have to be carefully tuned. In certain experimental setups, the retrieval effectiveness would not be improved at all. Consider, for instance, results reported by Craswell et al. [2010] in the notebook version of TREC 2010 proceedings. Craswell et al. [2010] calculate proximity as an inverse size of the span (the minimum window in a document that contains all query words), which results in a method that is even less effective than a method that does not take proximity into account.

Our statistical analysis has shown that if two query words occur closely in a document (a close pair), this is a strong indicator of relevance. We have conjectured that a simplified proximity ranking function that places documents with close pairs above documents without close pairs, should be a reasonable replacement for a computationally intensive ranking function (based on calculation of distances between words). However, this conjecture has been only partially confirmed by experiments: extending BM25 to account for close pairs improves P@5, P@15, and NDCG@20, in all cases, but not ERR@20 and the mean average precision (compare *BM25* against *BM25 (+close pairs)* and *BM25 (+spamrank)* against *BM25 (+close pairs +spamrank)* in Table I).

We have also shown that it is possible to index all close word pairs that occur in a quarter-terabyte text collection using commodity hardware, namely, our personal laptop. The resulting index of pairs moderately improves retrieval time, in comparison to a full positional index. This improvement comes at the price of longer indexing time and a larger index, which is around 280 percent of the original text size.

In the future we plan to investigate the following:

- How does the choice of L – the maximum value for the distance between words in indexed pairs – affect the retrieval effectiveness? In particular, it would be useful to learn how to obtain word-specific values of L: for less frequent words, one may want to include pairs where words are far from each other, while in the case of stop pairs (at least one word in a pair is a stop word) it may be reasonable to choose $L = 1$.
- Can a simplified relevance ranking that does not take the distance between words in close pairs into account be an effective proxy for more computationally intensive proximity ranking functions (e.g., based on computation of inverse squared distances)?
- How the index on word pairs can help pruning of the non-positional index effectively? Evaluation of BM25 scores takes long time. There are quite a few methods that improve time to retrieve top-k results using impact-sorted indices. We, however, do not know any method

that compensates for pruning of non-positional postings by memorizing occurrences of word pairs. In particular, if the word-pair index is available, it may be possible to store postings of frequent words in the form of bit vectors (i.e., omitting frequency information) without sacrificing retrieval effectiveness. This may be useful if one has to compute accurate number of documents found for each query.

REFERENCES

Bahle, D., Williams, H. E., and Zobel, J. 2002. Efficient phrase querying with an auxiliary index. In *SIGIR '02: Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, New York, NY, USA, 215–221.

Büttcher, S. and Clarke, C. L. A. 2005. Efficiency vs. effectiveness in terabyte-scale information retrieval. In *TREC-14: Proceedings of the Fourteenth Text REtrieval Conference*.

Chapelle, O., Metlzer, D., Zhang, Y., and Grinspan, P. 2009. Expected reciprocal rank for graded relevance. In *Proceeding of the 18th ACM conference on Information and knowledge management*. CIKM '09. ACM, New York, NY, USA, 621–630.

Clarke, C. L. A. and Cormack, G. V. 2000. Shortest-substring retrieval and ranking. *ACM Trans. Inf. Syst. 18*, 44–78.

Cormack, G. V., Smucker, M. D., and Clarke, C. L. A. 2010. Efficient and effective spam filtering and re-ranking for large web datasets. *CoRR abs/1004.5168*.

Craswell, N., Fetterly, D., and Najork, M. 2010. Microsoft research at trec 2010 web track. In *TREC-19: Proceedings of the Nineteenth Text REtrieval Conference*.

Maddala, G. 1999. *Limited-dependent and qualitative variables in econometrics*, Repr. ed. Number 3 in Econometric Society monographs. Cambridge Univ. Press.

Nelder, J. A. and Mead, R. 1965. A simplex method for function minimization. *The Computer Journal 7,* 4 (January), 308–313.

Rasolofo, Y. and Savoy, J. 2003. Term proximity scoring for keyword-based retrieval systems. In *Advances in Information Retrieval*, F. Sebastiani, Ed. Lecture Notes in Computer Science, vol. 2633. Springer Berlin / Heidelberg, 79–79.

Robertson, S. 2004. Understanding inverse document frequency: On theoretical arguments for IDF. *Journal of Documentation 60*, 503–520.

Robertson, S. E. and Walker, S. 1994. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*. SIGIR '94. Springer-Verlag New York, Inc., New York, NY, USA, 232–241.

Schenkel, R., Broschart, A., Hwang, S., Theobald, M., and Weikum, G. 2007. Efficient text proximity search. In *SPIRE'07: Proceedings of the 14th international conference on String processing and information retrieval*. Springer-Verlag, Berlin, Heidelberg, 287–299.

Tao, T. and Zhai, C. 2007. An exploration of proximity measures in information retrieval. In *SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, New York, NY, USA, 295–302.

Williams, H. E. and Zobel, J. 1999. Compressing Integers for Fast File Access. *The Computer Journal 42,* 3, 193–201.

Williams, H. E., Zobel, J., and Anderson, P. 1999. What's next? - index structures for efficient phrase querying. In *Proceedings of the Australasian Database Conference*. 141–152.