MMCI at the TREC 2010 Web Track

Andreas Broschart, Ralf Schenkel
Saarland University, Saarbrücken, Germany
{abrosch,schenkel}@mmci.uni-saarland.de

ABSTRACT

Term proximity scoring models incorporate distance information of query term occurrences and are an established means in information retrieval to improve retrieval quality. The integration of such proximity scoring models into efficient query processing, however, has not been equally well studied. Existing methods make use of precomputed lists of documents where tuples of terms, usually pairs, occur together, usually incurring a huge index size compared to term-only indexes. This paper uses a joint framework for trading off index size and result quality. The framework provides optimization techniques for tuning precomputed indexes towards either maximal result quality or maximal query processing performance under controlled result quality, given an upper bound for the index size.

1. INTRODUCTION

The MMCI group at Saarland University has participated in the 2010 Web Track Adhoc task. Our submission uses a joint framework described in [1] to tune index parameters to be used for later index construction. We impose an index size limit for the tuned index and keep an eye on the result quality at the same time. For our experiments in the TREC Web Track, we used the topics from the AdHoc task 2009 for training and optimizing index parameters. We considered only the 50% least spammy English ClueWeb09 documents from Category A according to the Waterloo Fusion spam scores (approximately 6 TB uncompressed size).

The remainder of the paper is structured as follows: Section 2 describes the score model our approach is based on, the employed index structures, and the query processing approach. Section 3 contains a detailed description of the runs we have submitted to the 2010 Web Track Adhoc task. We comment on some results and conclude in Section 4.

2. SYSTEM

2.1 Score Model

The proximity score model we use is based on [2], the modification was initially described in [4]. For a document d from a collection $\mathcal{C}$ of documents, we denote by the term frequency $tf_d(t)$ the number of times term t occurs in d, and the length $l_d = \sum tf_d(t)$ of document d is the sum of the term frequencies of all terms it contains. We use the established BM25 content score [3], which is computed for a query $q = \{t_1, \ldots, t_n\}$ of terms as

$$score_{\text{BM25}}(d,q) = \sum_{t \in q} \frac{tf_d(t) \cdot (k_1+1)}{tf_d(t) + k_1 \cdot (1 - b + b\frac{l_d}{avgdl})} \cdot idf(t)$$

where k_1 and b are tunable parameters, $avgdl$ is the average length of all documents in the collection, and $idf(t)$ is the inverse document frequency of t in the collection: Denoting by $df(t)$ the number of documents in which t occurs and by N the number of documents in $\mathcal{C}$, $idf(t)$ is defined as

$$idf(t) = \log \frac{N}{df(t)}$$

We denote by $p_i(d)$ the term at position i of d, omitting d when the document is clear from the context. For a term t, $P_d(t)$ denotes the positions in d where t occurs. For a query $q = \{t_1, \ldots, t_n\}$, $P_d(q) := \cup_{t_i \in q} P_d(t_i)$ denotes the positions of those terms in d, and

$$Q_d(q) := \{(i,j) \in P_d(q) \times P_d(q) \mid i < j \wedge p_i \neq p_j\}$$

denotes the position pairs of distinct terms from q in d. The score for a query $q = \{t_1, \ldots, t_n\}$ is then a linear combination of a standard BM25 content score and a BM25-style proximity score where term frequencies are replaced by per-term accumulators acc':

$$\begin{aligned} score_{\text{Büttcher}}(d,q) &= score_{\text{BM25}}(d,q) \\ &\quad + \sum_{t \in q} min\{1, idf(t)\} \frac{acc'_d(t) \cdot (k_1+1)}{acc'_d(t)+1} \end{aligned}$$

Here, the accumulator for term $t_k \in q$ is defined as

$$\begin{aligned} acc'_d(t_k) &= \sum_{(i,j) \in Q_d(q): p_i = t_k} \frac{idf(p_j)}{(i-j)^2} \\ &\quad + \sum_{(i,j) \in Q_d(q): p_j = t_k} \frac{idf(p_i)}{(i-j)^2} \end{aligned}$$

This score shows two major differences from the original score developed by Büttcher et al.: (1) it does not include the document length in the proximity score, and (2) accumulators combine not only adjacent query term occurrences. It has been shown in [4] that these modifications do not have an impact on result quality, but allow for efficient precomputation and indexing. A simple reformulation of the definition of $acc'_d(t_k)$ yields

$$\begin{aligned} acc'_d(t_k) &= \sum_{t \in q} idf(t) \cdot \underbrace{\sum_{\substack{(i,j) \in Q_d(q): \\ (p_i = t_k, p_j = t) \\ \vee (p_i = t, p_j = t_k)}} \frac{1}{(i-j)^2}}_{:= acc_d(t_k,t)} \\ &= \sum_{t \in q} idf(t) \cdot acc_d(t_k,t) \end{aligned}$$

Now $acc'_d(t_k)$ is represented as a monotonous combination of per-pair scores $acc_d(t_k,t)$, which can be precomputed for all possible term pairs and stored in an inverted index.

2.2 Description of the Index Structures

As described in [1] we use two different index structures:

- text index lists (short: TLs): each list stores, for a single stemmed term t and for each document d where this term occurs, an entry of the form $(d.docid, score_{\text{BM25}}(d,t))$ where $d.docid$ is a unique numerical id for document d. The entries are ordered by descending $score_{\text{BM25}}$ (i.e., BM25 score with $b = 0.5$ and $k_1 = 1.2$).
- combined index lists (short: CLs): each list contains, for a single stemmed term pair (t_1, t_2) and for each document where this term pair occurs within a text window of 10 terms in the document, an entry of the form $(d.docid, acc_d(t_1,t_2), score_{\text{BM25}}(d,t_1), score_{\text{BM25}}(d,t_2))$. The resulting proximity contribution of (t_1, t_2) for d is stored in $acc_d(t_1,t_2)$. Each index list is ordered by descending acc contribution.

The tuning process in this submission aims at finding appropriate tuning parameters for the later index construction such that index entries with the least impact in both TLs and CLs are not materialized.

We tune two parameters:

- the minimal score cutoff: We keep only those combined index list entries whose acc-score is not below a certain lower limit m.
- the list length cutoff: We keep at most the l entries from each text and combined index list that have the highest scores ($score_{\text{BM25}}$ for TL, acc for CL).

The index tuning process results in optimal choices for m and l which are named $\bar{m}$ and $\bar{l}$. More details are given in Section 5 of [1]. The resulting tuned indexes lead to efficient processing of queries using an n-ary merge join, still providing at least BM25 result quality and controlling the index space requirements. All indexes are compressed using delta- and v-byte encoding. Details about the different tuning approaches and the compression can be found in [1].

2.3 Query Processing

We use an n-way merge join with pruned index lists to get rid of the overhead costs dynamic pruning algorithms incur (cp. Section 6 in [1]). Our merge-based processing architecture consists of the following components:

1. After pruning index lists to a fixed maximal number of entries (and, possibly, using a minimal score cutoff for combined lists), we resort each list in ascending order of document ids, and optionally compress it.
2. At query time, the n text and combined lists for the query are combined using an n-way merge join that combines entries for the same document and computes its score. If that score is higher than the current k^{th} best score, the document is kept in a heap of candidate results, otherwise it is dropped.
3. Once all index entries have been read, the content of the heap is returned.

Instead of maintaining a heap with the currently best k results, an even simpler implementation could keep all results as result candidates and sort them at the end; however, this would increase the memory footprint of the execution as not k, but all encountered documents and their scores need to be stored.

3. DESCRIPTION OF THE RUNS

For the 2010 Web Track Adhoc task we submitted the following three runs:

- *MMCITLCLl20M:* This run uses TLs and CLs pruned to the first 20 million entries ($l = 20,000,000$ and $m = 0.00$) as input to an NRA (a dynamic pruning algorithm) implementation. The input lists consist of TLs for all stemmed query terms and CLs for all pairs of stemmed query terms. Stopwords are removed. We exclude the 50% spammiest documents according to the Waterloo Fusion spam score. The judging precedence was set to medium. We had to limit the index lists to the first 20 million entries as building the unpruned CLs exceeded our disk capacity.
- *MMCITLl20M:* This run uses TLs pruned to the first 20 million entries ($l = 20,000,000$) as input to an NRA implementation. The input lists consist of TLs for all stemmed query terms. Stopwords are removed. We exclude the 50% spammiest documents according to the Waterloo Fusion spam score. The judging precedence was set to least important. To be able to compare to *MMCITLCLl20M*, we decided to limit the index lists to the first 20 million entries for the TLs as well.
- *MMCIl410m1:* This run aims at providing the retrieval quality of BM25 scores (i.e., similar early precision quality) for $k = 10$ result documents and at the same time efficient query processing, still considering the index size. The indexes are tuned for efficient evaluation but still aim at providing BM25 result quality with a maximum index size of 1 TB for the test topics from the 2010 Web Track Adhoc task. We have used the efficiency-oriented absolute index quality tuning approach as described in [1] to come up with appropriate index tuning parameters. A list length of $\bar{l} = 410$ with a minimum score requirement of $\bar{m} = 1.00$ for the proximity contribution acc turned out to be optimal. To allow for efficient query processing, this run uses pruned TL and CL indexes that are reordered by docid then. The pruned indexes are the input to a merge join implementation. The input lists consist of pruned and reordered TLs for all stemmed query terms as well as pruned and reordered CLs for all pairs of stemmed query terms. Stopwords are removed. We exclude the 50% spammiest documents according to the Waterloo Fusion spam score. Judging precedence was set to most important.

For the tuning process that produced the index parameters for *MMCIl410m1*, we used the 50 Web Track 2009 topics as training topics and compared the respective precision values for the top-10 results at many pruning levels (i.e., many (l, m) combinations) with the precision for the top-10 results of the *MMCITLl20M* evaluation. Intuitively, the tuning process selects the smallest list length that can still achieve at least BM25 quality for the training topics within the index space constraint of 1 TB. If there is more than one solution for that list length, we select the one with the highest m to minimize the required index space.

As *MMCIl410m1* aims at providing the retrieval quality of BM25 and *MMCITLl20M* uses (almost) complete (i.e., non-pruned) BM25 score lists, *MMCITLl20M* is the baseline run for *MMCIl410m1*'s retrieval quality. *MMCITLCLl20M* is a baseline run for the effectiveness-oriented index tuning approaches. Due to the limitation of submittable runs, we can only provide runtimes for some of the runs.

Opt. goal	k	$(\bar{l}, \bar{m})$	size[GB]		prec@k		$\varnothing$reads$\cdot 10^{-5}$		$\varnothing$bytes$\cdot 10^{-5}$		$\varnothing t_{warm}[ms]$		$\varnothing t_{cold}[ms]$	
			est.	real	train	test	train	test	train	test	train	test	train	test
efficiency-oriented	10	(410,1.00)	503.6	499.8	0.190	0.2292	0.01	0.01	0.07	0.06	0.77	0.47	79.61	71.15
index quality	100	(400,1.00)	464.6	462.8	0.1170	-	0.01	0.01	0.07	0.06	0.63	0.44	69.05	55.60
effectiveness-oriented	10	(1410,1.00)	640.2	636.2	0.200	-	0.04	0.04	0.24	0.18	2.32	1.25	67.35	56.38
index quality	100	(4600,0.30)	979.0	977.9	0.1344	-	0.14	0.11	0.75	0.56	5.35	3.33	83.69	61.44

Table 1: Index tuning for absolute index quality

4. RESULTS AND CONCLUSION

Table 1 depicts our absolute index quality tuning results with an index size limit of 1 TB. [1] As described earlier, for our experiments in the Web Track Adhoc Task 2010, we used the topics from the Web Track AdHoc task 2009 as training topics and their relevance assessments to optimize the index parameters. We considered only the 50% least spammy English ClueWeb09 documents from Category A according to the Waterloo Fusion spam scores (approximately 6 TB uncompressed size).

We provide processing times of a single-threaded, Java-based implementation running on a single cluster node. These measurements were taken by running the complete batch of queries five times and taking the average. When we measure cold cache runtimes, we empty the filesystem cache before the evaluation of each query (not every query batch) which is a very conservative setting.

MMCITLl20M, the baseline run for indexes produced by efficiency-oriented tuning approaches, generates a precision@10 of 0.180, and a precision@100 of 0.1110 for the training topics. For the test topics, it produces a precision@10 of 0.2250 and a precision@100 of 0.1294.

With index parameters tuned for efficiency and top-10 document retrieval, the most efficient resulting index $(\bar{l}, \bar{m}) = (410, 1.00)$ (used in run *MMCIl410m1*) needs less than 500 GB of disk space. For the training topics, at a precision@10 of 0.190, it provides a result quality comparable to indexes that use pure BM25 scores (which generate a precision@10 of 0.180). Query processing with the index pruned at $(l, m) = (410, 1.00)$ requires on average less than 1 ms for warm caches and about 80 ms for cold caches to process one training topic.

Due to shorter index lists for the topics from the Adhoc task 2010 which serve as test topics, query processing is even a bit faster than for the training topics, showing more performant warm cache times and cold cache times of about 70 ms. This is reflected in the average number of accessed entries per query: while for the training topics we need on average 1,302 reads per query, we only need 1,045 reads for the test topics. The precision@10 of 0.2292 for the tuned index *MMCIl410m1* on the test topics slightly exceeds the precision@10 of 0.2250 for *MMCITLl20M*. The most efficient resulting index for top-100 document retrieval, $(\bar{l}, \bar{m}) = (400, 1.00)$ at a precision@100 of 0.1170 also meets the precision requirements for the training topics (*MMCITLl20M* has a precision@100 of 0.1110).

MMCITLCLl20M, the baseline run for indexes produced by effectiveness-oriented tuning approaches, generates a precision@10 of 0.198, and a precision@100 of 0.1324 for the training topics. For the test topics, it produces a precision@10 of 0.2250 and a precision@100 of 0.1358.

Our effectiveness-oriented index for top-10 document retrieval with $(\bar{l}, \bar{m}) = (1410, 1.00)$ aims at providing precision values which are comparable to *MMCITLCLl20M*. It requires 640 GB disk space and provides a precision@10 value of 0.200. This precision is comparable to the precision for the (almost) non-pruned proximity score run *MMCITLCLl20M* that has a precision@10 value of 0.198. Query execution takes about 2 ms for the training topics and slightly more than 1 ms for the test topics for warm caches.

The precision@10 value of 0.2250 for *MMCITLCLl20M* is the same as for *MMCITLl20M*; we would have expected higher precision values for *MMCITLCLl20M* like for our previous experiments on the GOV2 collection with the TREC 2004 to 2006 Terabyte Track (detailed results are provided in [1]). This may be partially due to the sparser relevance assessments for the Web Track topics. As to the precision@100 values *MMCITLCLl20M* performs better (at 0.1358) than *MMCITLl20M* (at 0.1294) which meets our expectations.

Our effectiveness-oriented index for top-100 document retrieval with $(\bar{l}, \bar{m}) = (4600, 0.30)$ has longer index lists than for k=10 which leads to a slightly slower query processing; despite the longer lists, it still fits the 1 TB space limit.

In [1] we also describe experiments with the GOV2 collection: Although the indexed part of ClueWeb09 is one order of magnitude larger than GOV2 (6 TB vs. 426 GB uncompressed size), the required index space does not grow as fast as the collection size. For the efficiency-oriented index quality setting, the parameter tuning for k=10 on ClueWeb09 results in $(\bar{l}, \bar{m}) = (410, 1.00)$ whereas for GOV2 it results in $(\bar{l}, \bar{m}) = (310, 0.05)$. The final index size is 500 GB for ClueWeb09 and 95 GB for GOV2. Indexes tuned for the ClueWeb09 collection often have shorter list lengths which provides even faster query processing.

5. REFERENCES

[1] A. Broschart and R. Schenkel. Real-time text queries with tunable term pair indexes. *MPI Technical Report MPI-I-2010-5-006*, 2010. Available at `http://www.mpi-inf.mpg.de/reports`.

[2] S. Büttcher, C. L. A. Clarke, and B. Lushman. Term proximity scoring for ad-hoc retrieval on very large text collections. In *SIGIR*, pages 621–622, 2006.

[3] S. E. Robertson and H. Zaragoza. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4):333–389, 2009.

[4] R. Schenkel, A. Broschart, S. won Hwang, M. Theobald, and G. Weikum. Efficient text proximity search. In *SPIRE*, pages 287–299, 2007.

[1] Please note that the retrieval quality measured in this paper is based on relevance assessments for 48 topics while the evaluation in [1] was based on the older relevance assessments that only contained 36 topics, hence the different precision values.