# A Practitioner's Guide for Static Index Pruning

Ismail Sengor Altingovde, Rifat Ozcan and Özgür Ulusoy

Department of Computer Engineering, Bilkent University, Ankara, Turkey
{ismaila, rozcan, oulusoy}@cs.bilkent.edu.tr

**Abstract.** We compare the term- and document-centric static index pruning approaches as described in the literature and investigate their sensitivity to the scoring functions employed during the pruning and actual retrieval stages.

## 1 Static Inverted Index Pruning

Static index pruning permanently removes some information from the index, for the purposes of utilizing the disk space and improving query processing efficiency. In the literature, several approaches are investigated for the static index pruning techniques.

Among those methods, the term-centric pruning (referred to as *TCP* hereafter) proposed in [3] is shown to be very successful at keeping the top-$k$ ($k \leq 30$) answers almost unchanged for the queries while significantly reducing the index size. In a nutshell, TCP scores (using the Smart's TFIDF function) and sorts the postings of each term in the collection and removes the tail of the list according to some decision criteria. In [1], instead of the TFIDF function, BM25 is employed during the pruning and retrieval stages. In that study, it's shown that by tuning the pruning algorithm according to the score function, it is possible to further boost the performance.

On the other hand, the document-centric pruning (referred to as *DCP* hereafter) introduced in [2] is also shown to give high performance gains. In DCP approach, only those terms that can most probably be queried are left in a document, and others are discarded. The importance of a term for a document is determined by its contribution to the document's Kullback-Leibler divergence (KLD) from the entire collection. However, the experimental setup in this latter work is significantly different than that of [3]. That is, only the most frequent terms of the collection are pruned and the resulting (relatively small) index is kept in the memory, whereas the remaining unpruned body of index resides on the disk. During retrieval, if the query term is not found in the pruned index in memory, the unpruned index is consulted. Thus, it is hard to infer how these two approaches, namely, TCP and DCP, compare to each other. Furthermore, given the evidence of recent work on how tuning the scoring function boosts the performance [1], it is important to investigate the robustness of these methods for different scoring functions that are employed during the pruning and retrieval, i.e., query execution.

In this paper, we provide a performance comparison of TCP and DCP approaches in terms of the retrieval effectiveness for certain pruning levels. Furthermore, for TCP, we investigate how using the Kullback-Leibler divergence scores, instead of TFIDF or BM25, during the pruning affects the performance. This may allow applying the TCP method independent of the retrieval function and thus providing

more flexibility for the retrieval system. For the sake of completeness, we also employ other scoring functions instead of KLD while selecting the most promising terms of a document in the DCP approach, and question whether KLD is the best scoring mechanism for this case. Our study puts light on the sensitivity of TCP and DCP approaches to the scoring functions and provides an in-depth comparison of these strategies under the same conditions.

## 2 Experimental Set-up and Results

**Pruning strategies.** For both TCP and DCP, we attempt to adapt the best set-up as reported in corresponding works [1, 2, 3]. In particular, for both approaches it's shown that using an adaptive strategy while deciding what to prune is better than a uniform strategy. Below, we outline these strategies as employed in our study.

- *TCP(I, k, $\varepsilon$)*: For each term *t* in the index *I*, first the postings in its posting list are sorted by a function (referred to as *PruneScore* function hereafter). Next, the $k^{th}$ highest score, $z_t$, is determined and all postings that have scores less than $z_t$ * ε are removed. Note that, as we are not considering theoretical guarantees in this paper, we use the $z_t$ scores as is, without the shifting operation as proposed in [2]. A similar approach has also been taken in [1]. In this algorithm, *k* is the number of results to be retrieved and $\varepsilon$ is the parameter to set the pruning level. In [3], Smart's TFIDF is used as both the PruneScore and retrieval function.
- *DCP(D, $\lambda$)*: For each document *d* in the collection *D*, its terms are sorted by the PruneScore function. Next, the top $|d|*\lambda$ terms are kept in the document and the rest are discarded. The inverted index is created over these pruned documents. In [2], KLD is employed as the PruneScore function and BM25 is used as the retrieval function.

In this paper, we consider the impact of the following parameters for these strategies.

- *PruneScore & retrieval functions:* First, for TCP and DCP, we employ each of the scoring functions (TFIDF, BM25 and KLD, as described in [1, 2, 3]) during the pruning stage. For the cases of using TFIDF and BM25 during the pruning, it's intuitive to use the same function in the retrieval stage, too. When KLD is used for pruning, we experiment with each of the TFIDF and BM25 functions, since KLD itself cannot be employed as a query-document similarity function.

  Note that, KLD is essentially proposed to be used with DCP [2], and our study is the first to use it for a term-centric approach. By this way, we attempt to score the postings of a list and prune them in a manner independent from the retrieval function. That is, we prune a posting by considering not only how "good" (or, "divergent") that particular term is for that document, but also how that posting's KLD score is ranked among the other postings of this term. Clearly, this is different from the DCP approach. The preliminary results are, although not conclusive, encouraging.

  For the sake of completeness, we also employ TFIDF and BM25 functions instead of KLD in the term scoring for DCP. It turns out that, KLD may not always be the best choice for DCP approach, and incorporating the actual retrieval function to the pruning stage can perform better.

- *Document lengths:* In [1], it is demonstrated that updating the document lengths after the index pruning further improves the performance. For all experiments, we try both of the approaches, namely *fix_dl* where document lengths are not updated after the pruning, and *upd_dl*, where lengths are updated.

  Note that, for TCP case where BM25 is employed during pruning, in addition to updating the lengths, we employed the other tunings reported to perform best in [1]. In particular, we prune all terms with *document frequency* > $|D|/2$ (where $|D|$ is the number of documents in the collection), and don't update the *average document length*. This case is denoted as "Bm25*" in Figure 1.

**Dataset and evaluation.** We use the AQUAINT corpus of English News Text along with the corresponding query topics and relevance judgments. The dataset includes 1,033,461 documents. During the indexing, stopwords are removed but no stemming is applied, yielding an index of 776,820 terms and 170,004,786 postings. We formed a set of 50 queries from the topic titles, including 2.5 terms on the average. Experiments with other query sets are not reported here due to lack of space. For each query, top-1000 results are retrieved. The results are evaluated using the trec_eval v8.1 software and reported in terms of MAP and P@10. For DCP and TCP, pruned index files are obtained at the pruning levels of 10%, 20%, etc. by tuning the $\lambda$ and $\varepsilon$ parameters, respectively. For TCP, top-$k$ parameter is set to 10 during the pruning.

**Experimental results: TCP.** In the plots, each case is denoted as a triple, namely, document length tuning (fix or updated), PruneScore function and retrieval function. We start with the performance of TCP approach. In Figure 1a, we provide MAP figures for the cases where the PruneScore function is TFIDF or KLD, and retrieval function is TFIDF. The curves reveal that KLD-TFIDF combination is as successful as TFIDF-TFIDF till 40% pruning level, whereas the performance is not sensitive to whether document lengths are updated or not. For P@10 (Figure 1b), TFIDF-TFIDF case with fixed document lengths is superior to the other combinations, which exhibit a mixed behavior. Similar experiments are provided in Figures 1c and 1d, where the PruneScore function is either BM25 or KLD, and retrieval function is BM25. In terms of MAP, employing KLD in the pruning stage achieves comparably good results up to 40% pruning level, but BM25 pruning is better thereafter. Again, the issue of the document length update is not very determinative, although the updated cases perform slightly better. For P@10 measure, document updates improve performance, as reported in [1]. Also, pruning with BM25 or KLD seems superior to others up to 60%. After 70%, KLD pruned cases are inferior to BM25 pruning.

Another important observation from Figure 1 is that the cases employing BM25 as the retrieval function achieve higher MAP and P@10 figures than the TFIDF cases. The drops in MAP and P@10 curves are sharper for TFIDF retrieval than the BM25 case. Indeed, for BM25 retrieval, the KLD and BM25 pruned index files yield precision results better than the baseline until 70%. This also conforms to [1].

As a summary, for TCP, BM25 retrieval function serves better and can be coupled with either BM25 or KLD based pruning. The effect of the document length updates is more emphasized for P@10 measure.

**Experimental results: DCP.** For DCP, we present the MAP and P@10 figures for the cases of KLD (or TFIDF) pruning and TFIDF retrieval function in Figures 2a and

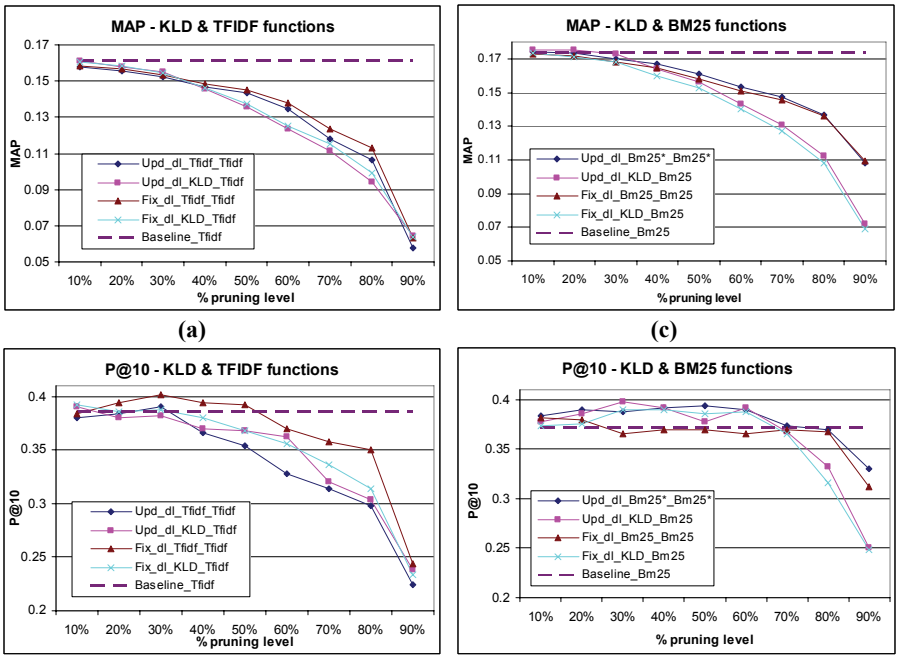


**Fig. 1.** Effectiveness results for TCP. Baseline is shown as a dotted line.

2b, respectively. Surprisingly, KLD based pruning performs worse than TFIDF based pruning. A similar trend is also observed for retrieval with BM25 in Figures 2c and 2d. This is different than the results presented in [2]. We attribute this difference to the fact that, as mentioned before, DCP has previously been applied to only the most frequent terms, for which the underlying model is expected to be more accurate [2]. In that work, the effect of applying DCP for all terms is not elaborated in detail (see section 5 in [2] for a brief discussion) and may require further investigation. Of course, another reason may be the differences in the datasets and experimental setup, e.g., we remove stopwords, in contrast to Büttcher et al. Figure 2 further reveals that for all cases, updating document lengths does not provide any improvements for DCP.

**Experimental results: TCP vs. DCP.** In Figure 3, we provide the comparison of the best cases for TCP and DCP. The MAP and P@10 plots reveal that TCP is better than DCP in terms of retrieval effectiveness, and the gaps are getting larger for higher pruning percentages.

## 3 Conclusion

In this study, we compare the performance of TCP and DCP algorithms on the static pruning of the entire index files, and show that the former performs better for our dataset and experimental set-up. We also propose to use KLD as a scoring function in the pruning stage of TCP, and show that it provides comparable results with BM25

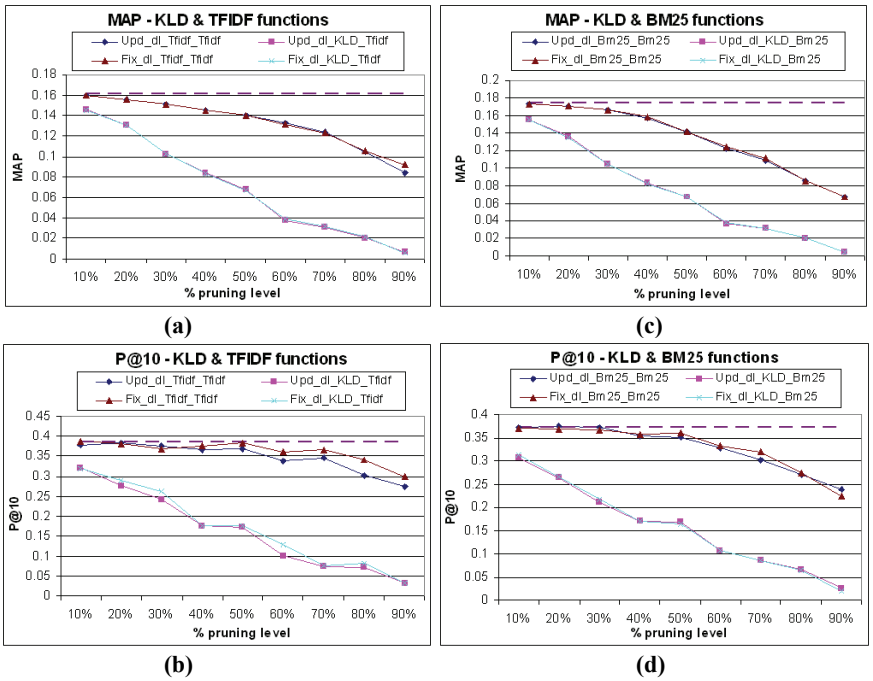


**Fig. 2.** Effectiveness results for DCP. Baseline is shown as a dotted line.

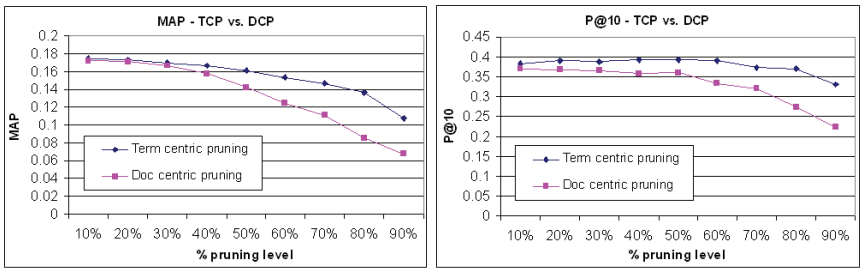


**Fig. 3.** Effectiveness comparison of TCP and DCP (best results).

and TFIDF, especially at the relatively lower pruning levels. In contrast, we find that for DCP, the actual retrieval function instead of KLD can be incorporated to the pruning stage for higher effectiveness.

**Acknowledgments.** This work is supported by TÜBİTAK by the grant# 108E008.

## References

1. Blanco, R., Barreiro, A. Boosting static pruning of inverted files. In: SIGIR'07, The Netherlands. (2007) 777-778
2. Büttcher, S., Clarke, C. L. A document-centric approach to static index pruning in text retrieval systems. In: Proc. of CIKM'06, USA. (2006) 182-189
3. Carmel, D., Cohen, D., Fagin, R., Farchi, E., Herscovici, M., Maarek, Y. S., Soffer, A. Static index pruning for information retrieval systems. In: SIGIR'01, USA. (2001) 43-50