Exploiting Query Views for Static Index Pruning in Web Search Engines

Ismail Sengor Altingovde, Rifat Ozcan, Özgür Ulusoy
Department of Computer Engineering, Bilkent University, 06800, Ankara, Turkey
{ismaila, rozcan, oulusoy}@cs.bilkent.edu.tr

ABSTRACT
We propose incorporating query views in a number of static pruning strategies, namely term-centric, document-centric and access-based approaches. These query-view based strategies considerably outperform their counterparts for both disjunctive and conjunctive query processing in Web search engines.

Categories and Subject Descriptors
H.3.1 [**Information Storage and Retrieval**]: Content Analysis and Indexing *–indexing methods* H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval *–search process.*

General Terms
Algorithms, Experimentation, Performance.

1. INTRODUCTION
Static index pruning techniques permanently remove a presumably redundant part of an inverted file, to reduce the file size and query processing time. The sole purpose of a static pruning strategy is staying loyal to the original ranking of the underlying search system especially for the top-ranked results, while reducing the index size, to the greatest extent possible.

In this paper, we propose a new pruning approach that exploits query views. In the literature, the idea of using query terms to represent a document is known as query view (e.g., [8]). In the scope of our work, for a given document, all queries that rank this particular document among their top-ranked results constitute the query view of that document. Our contributions are as follows:

- First, we fully explore the potential of a previous strategy, namely access-based pruning, that makes use of the query logs in the static index pruning context. To this end, we provide an adaptive version of the term-centric pruning algorithm provided in [6]. We also introduce a new document-centric version of the access-based algorithm, and show that the latter outperforms its term-centric counterpart.
- Second, we provide an effectiveness comparison of these access-based approaches to the term-centric approach [4] and document-centric approach [3], for their best performing setups reported in the literature. Our experimental findings reveal that, although the access based methods are inferior to the latter strategies for disjunctive query processing (as shown in the literature [6]), they turn out to be the most effective strategies when the queries are processed in the conjunctive mode. This is a new result that has not been reported before. Furthermore, the document-centric version of the access-based strategy as described here is found to be superior to all other strategies for conjunctive query processing, which has vital importance for WSEs.
- Finally, the main contribution of this paper is exploiting query views to tailor more effective static index pruning strategies for both disjunctive and conjunctive query processing; i.e., the most common query processing modes in WSEs. More specifically, the terms of a document that appear in the query view of this particular document are considered to be privileged and preserved in the index to the greatest possible extent during the static pruning. The query view heuristic is coupled with all three pruning approaches in the literature (term- and document-centric approaches as proposed in [4, 3], and the access-based term-centric method adapted from [6]) as well as the document-centric version of the access-based method that is introduced here.

  Our findings reveal that for both disjunctive and conjunctive query processing, the query view based pruning strategies reveal an excellent performance in terms of the similarity of the top-ranked results to the original results (i.e., obtained by using the original index) and significantly outperform their counterparts without query views. The gains are especially emphasized at the higher levels of pruning.

2. STATIC PRUNING APPROACHES
2.1 Baseline Static Pruning Algorithms
Term-Centric Pruning (TCP) strategy. TCP, the adaptive version of the top-k algorithm proposed in [4], is reported to be very successful in static pruning. In this strategy, for each term t in the index I, first the postings in t's posting list are sorted by a scoring function (e.g, TF-IDF). Next, the k^{th} highest score, z_t, is determined and all postings that have scores less than $z_t * \varepsilon$ are removed, where ε is a user defined parameter to govern the pruning level. Following the practice in [2], we simply determine ε values according to the desired pruning level. We also employ BM25 as the scoring function for TCP and entirely discard the terms with document frequency $f_t > N/2$ (where N is the total number of documents) as in [2]. In Algorithm 1, we demonstrate TCP strategy as adapted in our framework.

Document-Centric Pruning (DCP) strategy. In this paper, we apply the DCP strategy for the entire index, which is slightly different than pruning only the most frequent terms as originally

Algorithm 1 *Term-Centric Pruning*

```
Input: I, k, ε, N
1:  for each term t in I
2:      fetch the postings list $I_t$ from I
3:      if $|I_t| > N / 2$
4:          remove $I_t$ entirely from I
5:      if $|I_t| > k$
6:          for each posting entry <d>,
7:              compute Score(t, d) with BM25
8:          let $z_t$ be the kth highest score among the scores
9:          $\tau_t \leftarrow z_t * \varepsilon$
10:         for each posting entry <d>
11:             if Score(t, d) $\leq \tau_t$
12:                 remove entry <d> from $I_t$
```

proposed by [3]. Additionally, instead of scoring each term of a document with Kullback-Leibler divergence (KLD), we prefer to use BM25, as it is reported to perform better in [1]. Finally, in [3] it is again shown that the uniform strategy; i.e., pruning a fixed number of terms from each document, is inferior to the adaptive strategy, where a fraction (λ) of the total number of unique terms in a document is pruned. Algorithm 2 conveys the DCP strategy.

Algorithm 2 *Document-Centric Pruning*

```
Input: D, λ
1:  for each document $d \in D$
2:      sort $t \in d$ in descending order w.r.t. Score(d, t)
3:      remove the last $|d|*\lambda$ terms from d
```

2.2 Adaptive Access-based Pruning Strategies

Access-based Term-Centric Pruning (aTCP) strategy. For the first time in the literature, Garcia et al. used the search engine query logs to guide the static index pruning process [6]. However, their work does not use the actual content of the queries, but just makes use of the access count of a document; i.e., the number of times a document appears in top-*k* results of queries, where *k* is typically set to 1000. In particular, their algorithm applies the, so-called, MAXPOST heuristic, which simply keeps a fixed number of postings with the highest number of access count in each term's posting list. In that work, the result of the MAXPOST approach is found to be rather discouraging.

For this study, we decide to implement an adaptive version of the MAXPOST approach. Since it iterates over each term and removes some postings, we classify this approach as term-centric, and call the adaptive version *access-based TCP* (aTCP). In this case, instead of keeping a fixed number of postings in each list, we keep a fraction (μ) of the number of postings in each list. Algorithm 3 shows aTCP strategy.

Algorithm 3 *Access-based Term-Centric Pruning*

```
Input: I, μ, AccessScore[]
1:  for each term t in I
2:      fetch the postings list $I_t$ from I
3:      sort $d \in I_t$ in descending order w.r.t. AccessScore[d]
4:      remove the last $|I_t|*\mu$ postings from $I_t$
```

Access-based Document-Centric Pruning (aDCP) strategy. In this paper, we propose a new access-based strategy. Instead of pruning the postings from each list, we propose to prune documents entirely from the collection, starting from the documents with the smallest access counts. The algorithm is adaptive in that, for an input pruning fraction (μ), the pruning iterates while the total length of pruned documents is less than $|D|*\mu$, where $|D|$ is the collection length; i.e., total number of unique terms in the collection. Algorithm 4 presents this strategy, which we call *access-based DCP* (aDCP).

Algorithm 4 *Access-based Document-Centric Pruning*

```
Input: D, μ, AccessScore[]
1:  sort $d \in D$ in descending order w.r.t. AccessScore[d]
2:  NumPrunedPostings ← 0
3:  while NumPrunedPostings < $|D|*\mu$
4:      remove the document d with the smallest access score
5:      NumPrunedPostings ← NumPrunedPostings + $|d|$
```

3. PRUNING USING QUERY VIEWS

Let's assume a document collection $D = \{d_1,\ldots,d_N\}$ and a query log $Q = \{Q_1, \ldots,Q_M\}$, where $Q_i = \{t_1,\ldots,t_q\}$. After this query log Q is executed over D, the top-*k* documents (at most) are retrieved for each query Q_i, which is denoted as $R_{Qi,k}$. Now, we define the query view of a document *d* as: $QV_d = \cup\ Q_i$, where $d \in R_{Qi,\ k}$.

That is, each document is associated with a set of terms that appear in the queries that have retrieved this document within the top-*k* results. Without loss of generality, we assume that during the construction of the query views, queries in the log are executed in the conjunctive mode; i.e., all terms that appear in the query view of a document also appear in the document.

The set of query views for all documents, QV_D, can be efficiently computed either offline or online. In an offline computation mode, the search engine can execute a relatively small number of queries on the collection and retrieve, say, top-1000 results per query. Note that, as discussed in [6], it may not be necessary to use all of the previous log files; the most recent log or sampling from the earlier logs can be sufficient.

We envision that for a given document, the terms that appear as query terms to rank this document within top results of these queries should be privileged, and should not be pruned to the greatest extent possible. In what follows, we introduce four pruning strategies that exploit the query views, based on the TCP, DCP, aTCP and aDCP strategies, respectively.

Term-Centric Pruning with Query Views (TCP-QV). This strategy is based on Algorithm 1, but employs query views during pruning. In particular, once the pruning threshold (τ_t) is determined for a term *t*'s posting list, the postings that have scores below the threshold are not directly pruned. That is, given a posting *d* in the list of term *t*, if $t \in QV_d$, this posting is preserved in the index, regardless of its score. Note that, by only modifying line 11 as presented in Algorithm 5, the query view heuristic is taken into account to guide the pruning.

Algorithm 5 *Term-Centric Pruning with Query Views*

```
Input: I, k, ε, N, $QV_D$
1-9: //First 9 lines are the same as Algorithm 1 and not repeated
10:     for each posting entry <d>,
11:         if Score(t, d) $\leq \tau_t$ and $t \notin QV_d$
12:             remove entry <d> from $I_t$
```

Document-Centric Pruning with Query Views (DCP-QV). In this case, for the purpose of discussion, let's assume that each term t in a document d is associated with a priority score Pr_t, which is set to 1 if $t \in QV_d$ and 0 otherwise. The terms of a document d are now sorted (in descending order) according to these two keys, first the priority score and then score function output. During the pruning, last $|d|*\lambda$ terms are removed, as before. This strategy is demonstrated in Algorithm 6.

Algorithm 6 *Document-Centric Pruning with Query Views*

```
Input: $D$, $\lambda$, $QV_D$
1:  for each document $d$ in $D$
2:      for each term $t \in d$
3:          if $t \in QV_d$ then $Pr_t \leftarrow 1$ else $Pr_t \leftarrow 0$
4:      sort $t \in d$ in descending order wrt. first $Pr_t$, then Score($d$, $t$)
5:      remove the last $|d|*\lambda$ terms from $d$
```

Access-based Term-Centric Pruning (aTCP) with Query Views (aTCP-QV). In aTCP strategy, , we assume that each posting d in the list of a term t is associated with a priority score Pr_d, which is set to 1 if $t \in QV_d$ and 0 otherwise. Then, the postings in the list are sorted in the descending order of the two keys, first the priority score and then the access count. During the pruning, last $|I_t|*\mu$ postings are removed (Algorithm 7).

Algorithm 7 *Access-based Term-Centric Pruning with QV*

```
Input: $I$, $\mu$, AccessScore[], $QV_D$
1:  for each term $t$ in $I$
2:      fetch the postings list $I_t$ from $I$
3:      for each posting entry <d>,
3:          if $t \in QV_d$ then $Pr_d \leftarrow 1$ else $Pr_d \leftarrow 0$
7:      sort $d \in I_t$ in desc. order w.r.t. first $Pr_d$ then AccessScore[$d$]
8:      remove the last $|I_t|*\mu$ postings from $I_t$
```

Access-based Document-Centric Pruning (aDCP) with Query Views (aDCP-QV). In this case, we again prune the documents starting from those with the smallest access counts until the pruning threshold ($|D|*\mu$) is reached. But, while pruning documents, those terms that appear in the query view of these documents are kept in the index. This is shown in Algorithm 8.

Algorithm 8 *Access-based Document-Centric Pruning with QV*

```
Input: $D$, $\mu$, AccessScore[], $QV_D$
1:  sort $d \in D$ in descending order w.r.t. AccessScore[$d$]
2:  NumPrunedPostings $\leftarrow$ 0
3:  while NumPrunedPostings < $|D|*\mu$
4:      fetch $d$ with the smallest score
5:      for each term $t \in d$
6:          if $t \notin QV_d$
7:              remove $t$ from $d$
8:              NumPrunedPostings $\leftarrow$ NumPrunedPostings + 1
```

4. EXPERIMENTAL EVALUATION

Document collection and indexing. For this study, we crawled around 2.2 million pages from Open Directory Project (ODP) Web directory (www.dmoz.org). We first indexed the dataset using the publicly available Zettair IR system (www.seg.rmit.edu.au/zettair/). Once the initial index is generated, we used our homemade IR system to create the pruned index files and execute the training and test queries over them.

Training and test query sets. We use a subset of the AOL Query Log (http://imdc.datcat.org/collection/1-003M-5) that contains 20 million queries for a period of 12 weeks. The query terms are normalized by case-folding, sorting in the alphabetical order and removing the punctuation and stop-words. We consider only those queries of which all terms appear in the lexicon of the collection.

From the normalized query log subset, we construct training and test sets. The training query set that is used to compute the access counts and query views for the documents are from the first half (i.e., 6 weeks) of the log. The test set that is used to evaluate the performance for different pruning strategies are constructed from the second half (last 6 weeks) of the log. During the query processing with both training and test sets, a version of BM25 scoring function as described in [3], is used.

In the training stage, queries are executed in the conjunctive mode and top-1000 results per query are retrieved. Training set includes 1.8 million queries. We use a test set of 1000 randomly selected queries from the second half of the AOL log (i.e., it is temporally disjoint from the training set). These queries are also normalized. We keep only those queries that can retrieve at least one document from our collection when processed in the conjunctive mode. Furthermore, we guarantee that train and test sets are query-wise disjoint by removing all queries from the test set that also appear in the training set after the normalization.

Evaluation measure. In this paper, we compare the lists of top-*10* results that are obtained using the original and pruned index files. To this end, we employ the symmetric difference measure [4]. The score of 1 means exact overlap, whereas the score of 0 implies that two lists are disjoint.

Performance of the query views: disjunctive mode. In Table 1, we provide average symmetric difference results of all eight pruning strategies for the top-10 results and disjunctive query processing mode. In terms of the four baseline algorithms, the findings in this case confirm the earlier observations in [1, 4, 6]. Our adaptation of the access-based approach, aTCP, is the worst among all and only after 30% pruning, the symmetric difference score drops down to 0.54. On the other hand, the document-centric version of the access-based pruning strategy, aDCP, achieves much better performance; it is clearly superior to its term-centric counterpart and provides comparable results to DCP, at the early stages of the pruning (up to 50%). Among these four strategies, TCP is the clear winner whereas DCP is the runner-up and the access-based strategies are inferior to those, especially at the higher levels of pruning. This implies that solely using access counts is not adequate to guide the static index pruning.

Next, we evaluate the performance of the strategies with query views, namely TCP-QV, DCP-QV, aDCP-QV and aTCP-QV. A brief glance over Table 1 reveals that these approaches are far superior to their counterparts that are not augmented with query views. Remarkably, the order of algorithms is similar in that TCP-QV is still the best performer (though sometimes replaced by DCP-QV) and aTCP-QV is the worst. However, the gaps are now considerably closer. Indeed, the percentage improvement columns reveal that, query views enormously enhance the performance of the poor strategies (e.g., aTCP) at all pruning levels (ranging from 11% to 343%). Even for those strategies that were relatively more successful before, query views provide significant gains, especially at the higher levels of the pruning. For instance, at 50%

Table 1. Avg. symmetric difference scores for top-10 results and disjunctive query processing. Relative improvements w.r.t. the baseline algorithm are shown in the column Δ%. All improvements are statistically significant at 0.05 level using paired t-test.

%	TCP	DCP	aTCP	aDCP	TCP-QV	Δ%	DCP-QV	Δ%	aTCP-QV	Δ%	aDCP-QV	Δ%
10%	0.97	0.94	0.84	0.94	0.98	1%	0.98	4%	0.93	11%	0.96	2%
20%	0.91	0.86	0.68	0.87	0.95	4%	0.95	10%	0.88	29%	0.91	5%
30%	0.83	0.77	0.54	0.77	0.91	10%	0.93	21%	0.84	56%	0.86	12%
40%	0.74	0.68	0.42	0.66	0.86	16%	0.89	31%	0.80	90%	0.81	23%
50%	0.64	0.58	0.31	0.54	0.82	28%	0.84	45%	0.76	145%	0.77	43%
60%	0.55	0.49	0.22	0.41	0.79	44%	0.79	61%	0.74	236%	0.74	80%
70%	0.47	0.40	0.14	0.30	0.71	51%	0.66	65%	0.62	343%	0.66	120%

Table 2. Avg. symmetric difference scores for top-10 results and conjunctive query processing. Relative improvements w.r.t. the baseline algorithm are shown in the column Δ%. All improvements except (*)ed values are statistically significant at 0.05 level.

%	TCP	DCP	aTCP	aDCP	TCP-QV	Δ%	DCP-QV	Δ%	aTCP-QV	Δ%	aDCP-QV	Δ%
10%	0.66	0.80	0.93	0.98	0.94	42%	0.98	23%	0.97	4%	0.98	0%*
20%	0.52	0.66	0.86	0.96	0.90	73%	0.95	44%	0.94	9%	0.96	0%*
30%	0.41	0.54	0.78	0.91	0.86	110%	0.92	70%	0.91	17%	0.93	2%
40%	0.32	0.43	0.70	0.85	0.84	163%	0.88	105%	0.87	24%	0.90	6%
50%	0.25	0.33	0.60	0.79	0.81	224%	0.84	155%	0.84	40%	0.86	9%
60%	0.19	0.25	0.52	0.71	0.79	316%	0.79	216%	0.79	52%	0.81	14%
70%	0.15	0.17	0.43	0.61	0.51	240%	0.58	241%	0.71	65%	0.73	20%

pruning, the symmetric difference score jumps from 0.64 to 0.82 for TCP (a relative increase of 28%), and from 0.58 to 0.84 for DCP (45%). The relative improvements for all strategies exceed 10% after 20% pruning level. In short, query views significantly improve the baseline strategies, and carry them around 75-85% effectiveness at 50% pruning level, which is a solid success.

Performance of the query views: conjunctive mode. In Table 2, we provide symmetric difference results for the conjunctive case. Interestingly, conjunctive processing is mostly overlooked and has been taken into account in only few works [5, 7, 9], whereas it is the default and probably the most crucial processing mode for WSEs. Thus, we first analyse the results for the baseline strategies, which has not been discussed in the literature to this extent, before moving to query view based strategies.

Our experiments reveal that for the conjunctive processing mode, TCP is the worst strategy. This is a rather expectable result (see, for instance, [5, 9]). What is more surprising for this case is the performance of the access-based strategies: aDCP and aTCP outperform TCP and DCP with a wide margin at all pruning levels. This is a new result that has not been reported before in the literature. We think that one reason of this great boost in performance may be the conjunctive processing of the training queries while computing the access counts. In the previous work, both training and testing have been conducted in disjunctive mode. Another remarkable issue is, our document-centric version of the access based strategy, aDCP, significantly outperforms its term-centric adaptation. Indeed, aDCP achieves a similarity of 79% to the original results even when the index is halved.

Turning our attention to the query view based strategies, we again report important improvements. This time, the worst performing strategies, TCP and DCP, have most benefited from the query views, even more than doubling or tripling their similarity scores at certain pruning levels. The gains on access-based strategies are less emphasized, though reaching to 40% and 9% at 50% pruning for aTCP-QV and aDCP-QV, respectively. Note that, aDCP reaches to very high similarity scores of 0.98 and 0.96 at 10% and 20% pruning levels, respectively; and these are the only cases in Table 2 where the query view couldn't achieve any further improvements. For all other cases, query view based strategies again surpass their counterparts with a large margin, and reach to around 80% similarity at a pruning level of 60%.

5. ACKNOWLEDGMENTS

This work is supported by The Scientific and Technological Research Council of Turkey (TÜBİTAK) under the grant no 108E008. We thank Cigdem Gündüz-Demir and Ben Carterette.

6. REFERENCES

[1] Altingovde, I. S., Ozcan, R., and Ulusoy, Ö. A practitioner's guide for static index pruning. In *ECIR'09*, 675-679, 2009.

[2] Blanco, R. and Barreiro, A. Boosting static pruning of inverted files. In *Proc. of SIGIR'07*, 777-778, 2007.

[3] Büttcher, S. and Clarke, C. L. A document-centric approach to static index pruning in text retrieval systems. In *Proc. of CIKM'06*, 182-189, 2006.

[4] Carmel, D., Cohen, D., Fagin, R., Farchi, E., Herscovici, M., Maarek, Y. S., and Soffer, A. Static index pruning for information retrieval systems. In *SIGIR'01*, 43-50, 2001.

[5] de Moura, E. S., Santos, C. F., Araujo, B. D., Silva, A. S., Calado, P., and Nascimento, M. A. Locality-Based pruning methods for web search. *ACM TOIS* 26, 2, 1-28, 2008.

[6] Garcia, S. Search Engine Optimization Using Past Queries. Doctoral Thesis, RMIT University, 2007.

[7] Ntoulas, A. and Cho, J. Pruning policies for two-tiered inverted index with correctness guarantee. In *Proc. of SIGIR'07*, 191-198, 2007.

[8] Poblete, B. and Baeza-Yates, R. Query-sets: using implicit feedback and query patterns to organize web documents. In *Proc. of WWW'08*, 41-50, 2008.

[9] Skobeltsyn, G., Junqueira, F., Plachouras, V., and Baeza-Yates, R. ResIn: a combination of results caching and index pruning for high-performance web search engines. In *Proc of SIGIR'08*, 131-138, 2008.