A Document-Centric Approach to Static Index Pruning in Text Retrieval Systems

Stefan Büttcher Charles L. A. Clarke

University of Waterloo, Canada

November 7, 2006

Static Index Pruning

General Idea

- An inverted index for a text collection often contains data that are never used during query processing.
- We want to predict which postings (i.e., term/document pairs) will be needed to produce the search results and which will not.
- We restrict ourselves to the situation in which the top 20 documents for a given search query have to be found.
- Removing postings from the index that are unlikely to be used during query processing can improve the search engine's response time.
- If we remove the right postings, precision is not harmed (hopefully...).

Using Two Indices

- If the pruned index could be small enough to fit into main memory, then this would be really good.
- Unfortunately, the number of distinct terms is too large. GOV2: 50 million terms. 50M $\times$ 20 bytes $=$ 1GB.
- Solution: Build a pruned index for the frequent terms, e.g. top 1M terms. Load the pruned index into main memory.
- Whenever a term cannot be found in the pruned in-memory index, consult the unpruned on-disk index.

Term-Centric Index Pruning

Term-Centric Index Pruning

Carmel et al. (2001):

- Choose a scoring function, e.g., Okapi BM25.
- Pick an integer k and some ε from $[0, 1]$.
- For each term T in the index, compute the score impact of the k-th best posting, according to the selected scoring function: $S_T^{(k)}$.
- Set the term-specific pruning threshold to $\theta_T := \varepsilon \times S_T^{(k)}$.
- All postings with impact smaller than θ_T are removed from the index.

A Variation

Büttcher and Clarke (2005):

- Very similar to Carmel's method.
- Pick an integer N.
- For each of the N most frequent terms, keep the top k postings and discard the rest.
- N and k are chosen in such a way that the resulting index fits into main memory.
- We made good experience with this method in TREC Terabyte 2005.

In our experiments, we exlusively used this variant of Carmel's method.

Pseudo-Relevance Feedback

Kullback-Leibler Divergence

Given two probability distributions P and Q, the Kullback-Leibler divergence between P and Q is:

$$\text{KLD}(P, Q) = \sum_{x} P(x) \cdot \log\left(\frac{P(x)}{Q(x)}\right).$$

KL divergence is also referred to as *relative entropy*.

If P and Q are unigram language models, P generated from a document D, and Q generated from the whole text collection, then can use $\text{KLD}(P, Q)$ as an indicator for how different D is from the rest of the collection.

Carpineto's Feedback Mechanism

Carpineto et al. (2001) use Kullback-Leibler Divergence to select query expansion terms through pseudo-relevance feedback:

- Intuition: A good expansion term is a term that distinguishes a relevant document from the rest of the collection.
- Let $\mathcal{M}_C$ be the unigram language model for the text collection.
- Let $\mathcal{M}_D$ be the unigram language model for the pseudo-relevant document D.
- Assign each term T in D a score:

$$S_D(T) = \mathcal{M}_D(T) \cdot \log\left(\frac{\mathcal{M}_D(T)}{\mathcal{M}_C(T)}\right).$$

- Pick top f terms as expansion terms, according to their score.

Experiments with KLD-Based Feedback

Probability that *all query terms* are among the top f feedback terms, for top k documents retrieved by BM25:

	$k = 1$	$k = 5$	$k = 10$	$k = 20$
$f = 5$	30.0%	22.4%	18.0%	15.2%
$f = 10$	44.0%	35.2%	28.0%	23.6%
$f = 20$	64.0%	53.6%	47.2%	39.9%
$f = 40$	80.0%	69.6%	64.0%	58.2%

(for 50 ad-hoc topics from TREC Terabyte 2005)

Experiments with KLD-Based Feedback

Probability that *at least one query term* is among the top f feedback terms, for top k documents retrieved by BM25:

	$k = 1$	$k = 5$	$k = 10$	$k = 20$
$f = 5$	94.0%	92.0%	87.6%	79.4%
$f = 10$	98.0%	97.6%	96.8%	92.0%
$f = 20$	100.0%	99.6%	99.4%	97.8%
$f = 40$	100.0%	99.6%	99.8%	99.5%

(for 50 ad-hoc topics from TREC Terabyte 2005)

Document-Centric Index Pruning

Query-Independent Feedback

- Document-centric index pruning uses KLD-based pseudo-relevance feedback.
- Prune the index by performing query-independent pseudo-relevance feedback on individual documents.
- From each document, keep the top terms according to feedback score. Discard the rest.
- How many terms per document should be kept?

$\mathrm{DCP}_{\mathrm{const}}$

The simplest selection strategy is referred to as $\mathrm{DCP}_{\mathrm{const}}$:

- Choose an integer k.
- From each document, keep the top k feedback terms.
- Discard the rest.

DCP_{rel}

$\text{DCP}_{\text{const}}$ is unfair against longer and more diverse documents.

Longer documents may cover a greater variety of topics and should therefore be allowed to contribute a greater number of terms to the pruned index.

This is realized by DCP_{rel}:

- Choose a parameter λ from $[0, 1]$.
- From each document D keep the top $|D| \times \lambda$ terms, where $|D|$ denotes the number of distinct terms in D.
- Discard the rest.

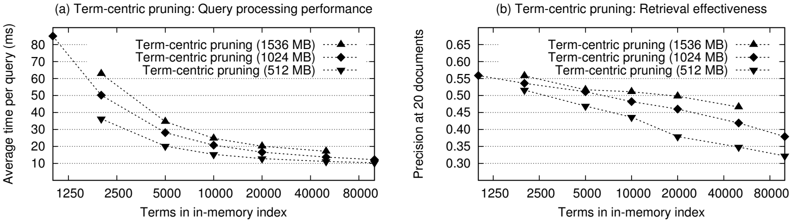
Experimental Results

Experimental Setup

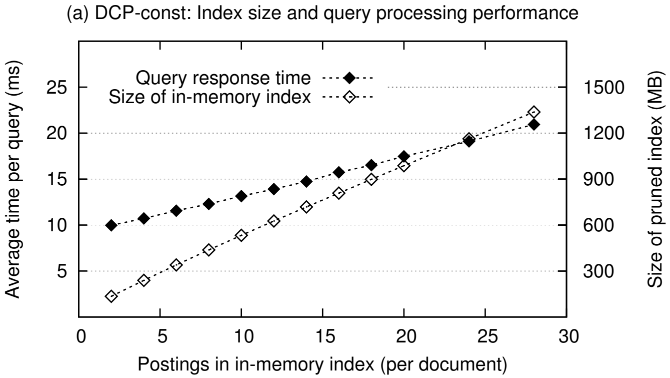
Index pruning is evaluated by measuring different efficiency/effectiveness trade-off points.

- Text collection: GOV2 (25.2 million documents, 44 billion tokens).
- Effectiveness: 50 ad-hoc topics from TREC TB 2005.
- Efficiency: 50,000 efficiency topics from TREC TB 2005.
- Retrieval baseline: BM25 with a document-level frequency index (stored on disk). P@20 = 0.5660. Average time per query: 190.5 ms.

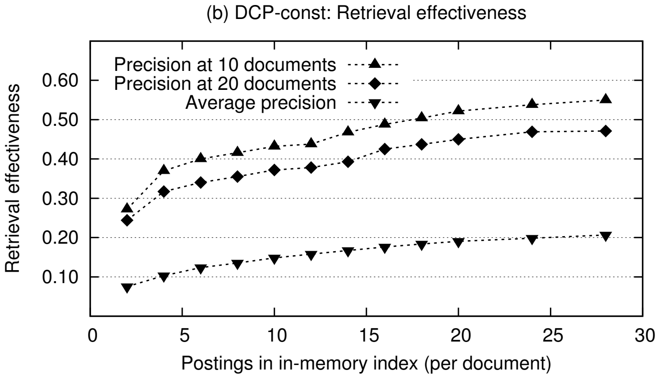
Term-Centric Pruning



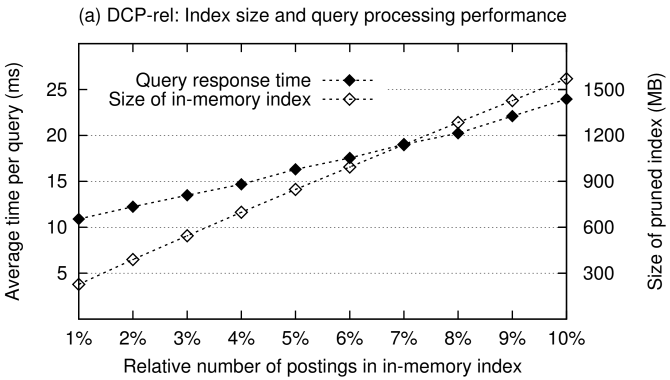
$\text{DCP}_{\text{const}}$



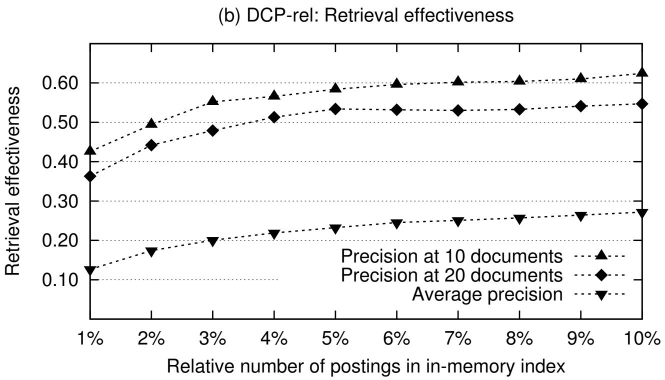
DCP_{const}


(b) DCP-const: Retrieval effectiveness
Precision at 10 documents
Precision at 20 documents
Average precision
Retrieval effectiveness
0.60
0.50
0.40
0.30
0.20
0.10
0
5
10
15
20
25
30
Postings in in-memory index (per document)

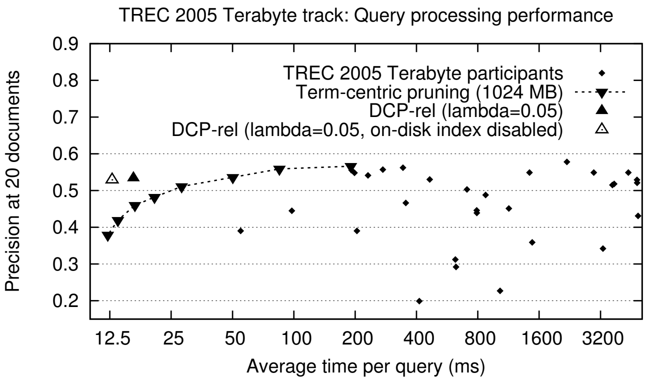
DCP_{rel}


(a) DCP-rel: Index size and query processing performance
Query response time
Size of in-memory index
Average time per query (ms)
25
20
15
10
5
1500
1200
900
600
300
Size of pruned index (MB)
1%
2%
3%
4%
5%
6%
7%
8%
9%
10%
Relative number of postings in in-memory index

DCP_{rel}



Efficiency vs. Effectiveness



Statistical Significance

[TREC 2004 Terabyte queries (topics 701-750)]

	BM25 Baseline	$DCP_{Rel}^{(\lambda=0.062)}$	$DCP_{Const}^{(k=21)}$	$TCP_{(n=16000)}^{(k=24500)}$
P@5	0.5224	0.5020	0.4735	0.4490*
P@10	0.5347	0.4837	0.4755	0.4347*
P@20	0.4959	0.4490	0.4224	0.4163
MAP	0.2575	0.1963	0.1621**	0.1808

[TREC 2005 Terabyte queries (topics 751-800)]

	BM25 Baseline	$DCP_{Rel}^{(\lambda=0.062)}$	$DCP_{Const}^{(k=21)}$	$TCP_{(n=16000)}^{(k=24500)}$
P@5	0.6840	0.6760	0.6000**	0.5640**
P@10	0.6400	0.5980	0.5300*	0.5380**
P@20	0.5660	0.5310	0.4560**	0.4630**
MAP	0.3346	0.2465	0.1923**	0.2364

*: $p < 0.05$ (compared to DCP_{rel})

**: $p < 0.01$ (compared to DCP_{rel})

Similarity to Original Search Results

Comparing the search results produced from the pruned index with the results produced from the unpruned index.

- Symmetrical difference: $1 - \frac{|\text{intersection}|}{|\text{union}|}$ (top 20 documents).
- Kendall's τ: Modified version, restricted to top-20 results.

Pruning level	1 - symm.diff.	Kendall's τ (top-20)
$\lambda = 0.04$	0.4403	0.6915
$\lambda = 0.06$	0.5421	0.7753
$\lambda = 0.08$	0.6090	0.8197
$\lambda = 0.10$	0.6716	0.8557

Summary

- Document-centric index pruning is based on query-independent per-document pseudo-relevance feedback.
- Document-centric index pruning can be used to substantially decrease the average time per query (from 190 ms to 20 ms in our experiments).
- If not applied too aggressively, search quality (P@20) is almost not harmed.
- Search results stay reasonably close to the original results produced from the unpruned index.
- In a search engine, index pruning can be enabled selectively, to overcome short periods of very high query activity.

The End

Thank You!