Effect of Different Docid Orderings on Dynamic Pruning Retrieval Strategies

Nicola Tonellotto
Information Science and Technologies Institute
National Research Council
Via G. Moruzzi 1, 56124 Pisa, Italy
nicola.tonellotto@isti.cnr.it

Craig Macdonald, Iadh Ounis
School of Computing Science
University of Glasgow
Glasgow, G12 8QQ, UK
{craig.macdonald,iadh.ounis}@glasgow.ac.uk

ABSTRACT

Document-at-a-time (DAAT) dynamic pruning strategies for information retrieval systems such as MaxScore and Wand can increase querying efficiency without decreasing effectiveness. Both work on posting lists sorted by ascending document identifier (docid). The order in which docids are assigned – and hence the order of postings in the posting lists – is known to have a noticeable impact on posting list compression. However, the resulting impact on dynamic pruning strategies is not well understood. In this poster, we examine the impact on the efficiency of these strategies across different docid orderings, by experimenting using the TREC ClueWeb09 corpus. We find that while the number of postings scored by dynamic pruning strategies do not markedly vary for different docid orderings, the ordering still has a marked impact on mean query response time. Moreover, when docids are assigned by lexicographical URL ordering, the benefit to response time for is more pronounced for Wand than for MaxScore.

Categories & Subject Descriptors: H.3.3 [Information Storage & Retrieval]: Information Search & Retrieval

General Terms: Performance, Experimentation

Keywords: Dynamic Pruning, Docid Ordering

1. INTRODUCTION

MaxScore [7] and Wand [2] are two document-at-a-time (DAAT) dynamic pruning strategies that can efficiently score documents without decreasing retrieval effectiveness at rank K, nor requiring impact-sorted posting lists. Instead, the postings for every term have the same global ordering, such that they can be read in parallel.

MaxScore achieves efficient retrieval by omitting the scoring of postings for documents that will not make the final retrieved ranking. In contrast, Wand takes a different approach, by repeatedly calculating a *pivot term*. The next document to contain the pivot term is called the *pivot document*, which will be the next document to be fully scored. A major benefit of Wand over MaxScore is that skipping [5] forward in posting lists is supported, which reduces posting list decompression overheads, and can reduce disk IO.

The ordering of index docids has been studied within the literature. In particular, as the posting lists typically record delta gaps between docids, ensuring that these gaps are as small as possible increases compression, while allowing efficient memory use at retrieval time. For instance, Blelloch and Blandford [1] argued that maximising the *clustering property* of posting lists so that documents with similar content should have similar docids resulted in smaller delta gaps. Silvestri [6] later showed that high clustering could be cheaply approximated by ordering docids according to the lexicographical order of their URLs.

Nevertheless, while URL docid ordering can attain highly compressed indices, there has been no published work examining the impact of docid orderings on the efficiency of dynamic pruning strategies. To address this, this work contributes a study of several dynamic pruning strategies using different docid orderings.

2. EXPERIMENTAL SETUP

Our investigation is conducted using the TREC ClueWeb09 (cat. B) collection, which comprises 50 million English Web documents, aimed to represent the first tier index of a commercial search engine. We index this collection using the Terrier information retrieval platform[1], with stemming and stopword removal. Each posting consists of only the Elias-Gamma encoded docid gap and the Elias-Unary encoded frequency. Skip points within the posting lists are placed in a separate file, with $L = 10,000$ [5].

For index orderings, we consider three alternatives, detailed below. Basically, the idea is to put documents sharing many common terms close to each other and assign identifiers accordingly. In this way the likelihood of having small gaps in posting lists is higher. Smaller gaps correspond to smaller (in terms of bit occupancy) codewords and, thus, smaller indexes.

Random ordering: As a baseline, we randomise the ordering of documents in our index, assigning identifiers to documents without any specific order.

Doclength ordering: Longer documents are more likely to share common terms. In this paper, we assign document identifiers by decreasing length of documents (i.e. the longest document takes docid 0), which has been shown to have a positive impact on the size of the resulting index [3].

URL ordering: Docids are assigned according to the lexicographical ordering of the URLs[2]. URL ordering is good for compression, as it approximates the clustering property [6].

By indexing ClueWeb09 with these different variants of docid ordering, we obtain several different inverted files. Statistics of the inverted files are detailed in Table 1. We

[1] `http://terrier.org`

[2] We would also liked to have used crawl ordering, however URL ordering is all that is available for ClueWeb09.

SIGIR'11, July 24–28, 2011, Beijing, China.
ACM 978-1-4503-0757-4/11/07.

Ordering	Inverted File Size	
Random	22.4 GB	100%
Doclength	19.6 GB	87.5%
URL	14.7 GB	65.6%

Table 1: Statistics of the created inverted files.

note that URL ordering gives the highest compression, followed by doclength. As expected, a random ordering increases the delta gaps in the posting lists, resulting in the largest inverted file size.

In the following experiments, we compare the MaxScore and Wand DAAT dynamic pruning strategies, as well as an exhaustive "Full" DAAT strategy, which scores all postings for each query term. For testing retrieval efficiency, we extract streams of user queries from a real search engine log [4]. In particular, we select the first 1,000 queries of the query log, applying stemming and stopwords removal (empty and no-match queries are removed). Two experiments are performed ranking documents using BM25, namely to evaluate the query average response time and the number of postings fully scored by each strategy. The number of documents retrieved for each query is set to the TREC standard value of $K = 1,000$. All experiments are made using a dual quad-core Intel Xeon 2.6GHz, with 8GB RAM and a 2TB SATA2 disk containing the index. Note that, as MaxScore and Wand do not impact on retrieval effectiveness to rank K, there is no need to compare effectiveness using measures such as NDCG.

		Ordering						
		random		doclength		URL		
postings	Full	3,973	100%	3,973	100%	3,973	100%	
	MaxScore	1,717	43.2%	1,871	47.0%	1,732	43.6%	
	Wand	428	10.8%	572	14.4%	436	10.9%	
times	Full	1.36	100%	1.36	100%	1.32	100%	97.1%
	MaxScore	1.24	91.1%	1.24	91.1%	1.17	88.6%	94.4%
	Wand	0.96	70.5%	0.96	70.5%	0.84	63.6%	87.5%

Table 2: Total number of postings scored in millions (top) and mean query response time in seconds (bottom), for 1,000 queries. Relative percentages compared to Full are reported. Relative percentages for URL ordered response time compared to random are reported on right.

3. RESULTS

Table 2 reports the results of our experiments in terms of both millions of postings scored and mean query response time for the 1,000 queries.

We first compare different docid orderings. In general, we note that response times are lower for URL ordering, while random and doclength are approximately the same. This is in marked contrast to Table 1, where the random baseline ordering resulted in a markedly larger inverted file than doclength, and hence a larger response time would be expected for the random ordering. In fact, the number of postings scored for the random index are actually the lowest. This suggests that even if less time is spent decoding and processing less postings, this is counter-balanced by more time spent by the operating system in managing IO operations for the larger random posting lists. On the other hand, while URL ordering does not markedly decrease the number of postings scored, it markedly benefits query response time, particularly for the Wand strategy.

With respect to dynamic pruning strategies, MaxScore scores 43-47% of all postings, while Wand scores 11-15%. This difference is also reflected, although with less strength, in the mean response times, where MaxScore and Wand respectively take about 90% and 63-70% of the response time of the exhaustive Full strategy. Once again, this can be explained by the fact that larger inverted files require more IO operations, which counter-act the benefits given by the reduced decoding and processing of postings.

Of all results, we note with particular interest that, while URL ordering scores more postings than random, the response time benefit is larger for Wand (87.5%) than for the other strategies. We hypothesise that by applying URL ordering, not only is the average gap between docids reduced, but the postings for the retrieved documents are placed closer together, which benefits the skipping used by Wand. To confirm this, for the 1,000 queries, we examined the mean standard deviation of the docids for the retrieved documents. On average, the URL ordered index had a smaller mean standard deviation (245k postings) than random (382k postings) and doclength (359k postings). The lower mean standard deviation suggests that the retrieved documents are more likely to be clustered within the URL ordered posting lists. As a consequence, the response time of Wand is enhanced by this URL ordering, as more clustered results permit more skipping (with the associated benefit of avoiding unnecessary posting decompression).

4. CONCLUSIONS

This paper is the first examination of index ordering in the context of DAAT dynamic pruning strategies. Our results show that the number of postings scored does not markedly vary for different index orders when applying the MaxScore and Wand pruning strategies. However, mean query response time does exhibit marked reductions, and comparably more for the dynamic pruning strategies than for an exhaustive Full DAAT strategy. These results show the value in URL docid ordering for both increasing inverted file compression and enhancing the efficiency of dynamic pruning strategies. Moreover the clustering property of posting lists benefits both query processing time (particularly when using skipping) and index compression. Finally, we note that, with 50 million documents, our experiments are carried out at a larger scale than previous works – for instance, 100 times as many documents than [1], and 10 times as many as [6].

5. REFERENCES

[1] D. Blandford and G. Blelloch. Index compression through document reordering. In *Proceedings of the Data Compression Conference 2002*.

[2] A. Z. Broder, D. Carmel, M. Herscovici, A. Soffer, and J. Zien. Efficient query evaluation using a two-level retrieval process. In *Proceedings of CIKM 2006*, 426–434.

[3] S. Büttcher, C. L. A. Clarke, and G. V. Cormack. Information retrieval: implementing and evaluating search engines. MIT Press, 2010.

[4] N. Craswell, R. Jone, G. Dupret, and E. Viegas. *Proceedings of the 2009 workshop on Web Search Click Data*. ACM Press.

[5] A. Moffat and J. Zobel. Self-indexing inverted files for fast text retrieval. *Transactions on Information Systems*, 14(4):349–379, 1996.

[6] F. Silvestri. Sorting out the document identifier assignment problem. In *Proceedings of ECIR 2007*, 101–112.

[7] H. Turtle and J. Flood. Query evaluation: strategies and optimizations. *Information Processing and Management*, 31(6):831–850, 1995.