Dublin City University at the TREC 2006 Terabyte Track

Paul Ferguson, Alan F. Smeaton and Peter Wilkins
Centre for Digital Video Processing & Adaptive Information Cluster
Dublin City University, Glasnevin, Dublin 9, Ireland
{pferguson, asmeaton, pwilkins}@computing.dcu.ie

Abstract

For the 2006 Terabyte track in TREC, Dublin City University's participation was focussed on the ad hoc search task. As per the pervious two years [7, 4], our experiments on the Terabyte track have concentrated on the evaluation of a sorted inverted index, the aim of which is to sort the postings within each posting list in such a way, that allows only a limited number of postings to be processed from each list, while at the same time minimising the loss of effectiveness in terms of query precision. This is done using the Físréal search system, developed at Dublin City University [4, 8]

1 Introduction

As in the previous two years, the Terabyte Track experiments in TREC 2006 were run on the GOV2 collection; a collection of over 25 million documents, crawled from the .gov domain in early 2004.

This year Dublin City University participated in the ad hoc search task. The aim of this task was to investigate the performance of systems on a static set of documents with a set of previously unseen topics. For this task NIST provided the participants with 50 new topics to search for relevant documents on. Participants were then and asked to return a ranked set of the 10,000 most highly-ranked documents for each of these topics.

In this paper section 2 will describe the sorted index used in our experiments. Section 3 outlines the search engine set up that we used, while section 4 explains the experiments we carried out for this years ad hoc search task. Finally, section 5 explores the outcome of these results and draws some conclusions from the work.

2 Sorted Inverted Index

The conventional approach for sorting posting lists, within an inverted index is to sort them in ascending order based on the document identifier. This allows for easy compression, based on storing the difference between consecutive identifiers (*d-gaps*). However it has been found that sorting based on other measures may allow for faster processing of queries, by processing only the postings early in the list, as defined by the sorting metric.

Persin et al [11, 12] proposed a *frequency sorted* index, based on the within document term frequency for each term, so that documents may be filtered, in order to reduce the number of documents to be processed, and in doing so, reducing both the main memory in use, as well as the query time. This reduction is achieved by "processing only the most informative parts of the term entries", and to allow this the postings lists are sorted in descending order of the within document frequency of each posting.

Anh and Moffat [2, 3, 1] proposed an *impact* sorted index. They suggest a term-impact, where the impact is calculated based on the influence of a term within a document, similar to [5], as well

as a document-centric impact, which calculates the impact at a document, rather that a global level.

Garcia et al [10], alternatively suggest sorting the inverted index based on *access counts*. Founded on the idea that even with a large number of different queries the same documents are quite often ranked highly, while other documents are rarely if ever returned to the user, the postings are ordered so that the documents relevant for most queries are towards the top of the lists and the less retrieved documents are stored towards the bottom.

2.1 BM25 Sorting

The Okapi BM25 model was proposed by Robertson et al [13], and remains one of the most widely used methods of assigning similarity between documents and a query, and so is widely using in information retrieval.

For an ad hoc retrieval and ignoring any repetition of terms in the query, as is the case for the vast majority of web queries, this function can be simplified to:

$$bm25(q,d) = \sum_{t \epsilon q} log\left(\frac{N - df_i + 0.5}{df_i + 0.5}\right) \times \frac{(k_1+1)tf_i}{k_1((1-b) + b\frac{dl}{avdl}) + tf_i} \quad (1)$$

where df_i is the number of documents in the collection that contain the term i.

This calculates a similarity score between a query and a document, and with this formulation the scores for all query terms are summed to give a final score. However if we use this at indexing time on a term by term basis to calculate the BM25 scores for each of the documents within that posting list and then sort the posting list using these scores, this is in effect a pre-calculated BM25 ranking for each posting list, and the only missing element from calculating a full query-dependent score is the summation with other scores from the relevant query terms. This of course is something that can only be known at query time. Therefore it is as close at we can get to a query-dependent estimation of the query-dependent BM25 score.

Also if we are only concerned with calculating a score that is only to be used as a means by which to sort postings and this is done on a term by term basis, there is then no need to include any global term information such as df_i (the number of documents in the collection that contain the term i). This leaves a simplified calculation of:

$$bm25(d,i) = \frac{(k_1+1)tf_i}{k_1((1-b) + b\frac{dl}{avdl}) + tf_i} \quad (2)$$

We used this formula for sorting our inverted index for this years experiments.

3 Físréal Search Engine Setup

For the 2006 TREC experiments, our search system was run on a single machine, a Pentium 4, 2.6 GHz with 1.5 GB of RAM. The entire collection was indexed on this machine and queries were also run locally on this machine.

3.1 Early Termination

Similar to our previous involvements in the Terabyte track in 2004 and 2005, we again utilised the *top subset* approach [7, 4, 9] for selecting the postings to be processed from each query term. We have found that this is also the same approach that is utilised by Garcia et al [10], where it is referred to as *maxpost*. Using this strategy, a maximum number of postings is chosen to be processed from each posting list, e.g. 100,000, then at query time, for each query term, this is the maximum number of postings that will be processed for each query term. We have also experimented with selecting the number of postings to process, based on a percentage of the the postings, however this is outperformed by the fixed size approach. We believe this approach works well as it means that a greater percentage of postings from uncommon terms, i.e. those that tend to affect the ranking more, are processed, while a large percentage of postings from a frequently occurring term may not be processed, as these terms affect the ranking the least.

3.2 Retrieval

In order to provide the matching between documents and queries, and to allow document ranking, we again utilised the BM25 formula, as shown in equation 1.

4 Experiments

For this years ad hoc task we submitted a single run, DCU05BASE (which incidentally should have been named DCU06BASE !). For this run we elected to use a *top subset* size of 200,000 for each of the posting lists, where the run is automatically processed, using only the title information for each topic.

The table 1 shows the performance of this run in terms of mean average precision (MAP), bpref, precision at 5 documents (P5) and precision at 10 documents (P10)

Table 1: Performance figures for DCU05BASE

MAP	bpref	P5	P10
0.2748	0.3577	0.6013	0.5671

4.1 Analysis

To illustrate the usefulness of this approach it is beneficial to look at the tradeoff that is being made between the accuracy of the retrieval, compared with the amount of processing that is required.

Figure 1 shows that by processing less postings there is certainly a small degradation in terms of MAP, compared with processing all postings for each query term. Figure 2 shows that there is less of a decline in terms of precision at 10 documents, by choosing to process less postings per query term. For example there is only a very small difference between processing 200,000 postings per term, compared with processing all postings: 0.5671 and 0.5732 respectively. Meanwhile processing less postings per term requires much fewer postings to be processed overall, as can be seen from Figure 3, for example by again choosing the 200,000 cutoff point, only 13.9% of all postings are processed, although it should be clear that this does not translate to 13.9% of the performance of processing all postings, as this approach aims to process the document entries most likely to be relevant, first.

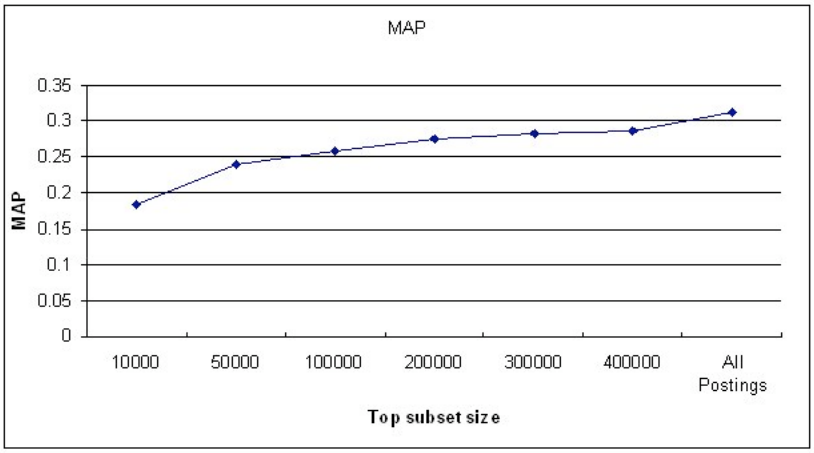

Figure 1: MAP performance, using different top subset sizes

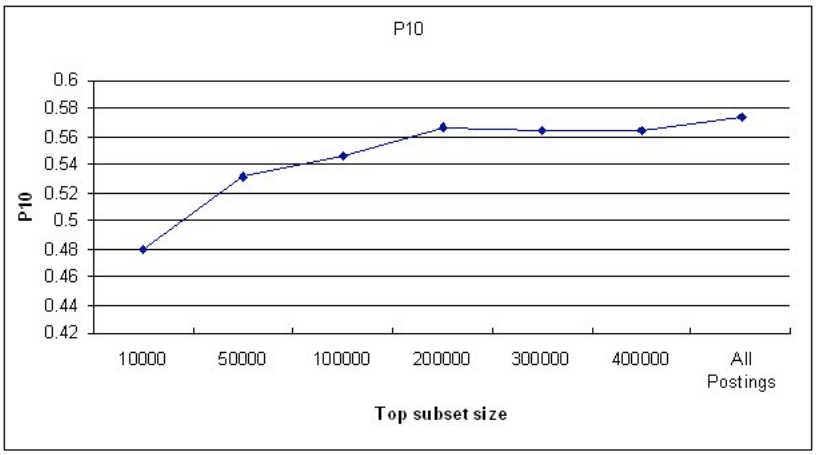

Figure 2: P10 performance, using different top subset sizes

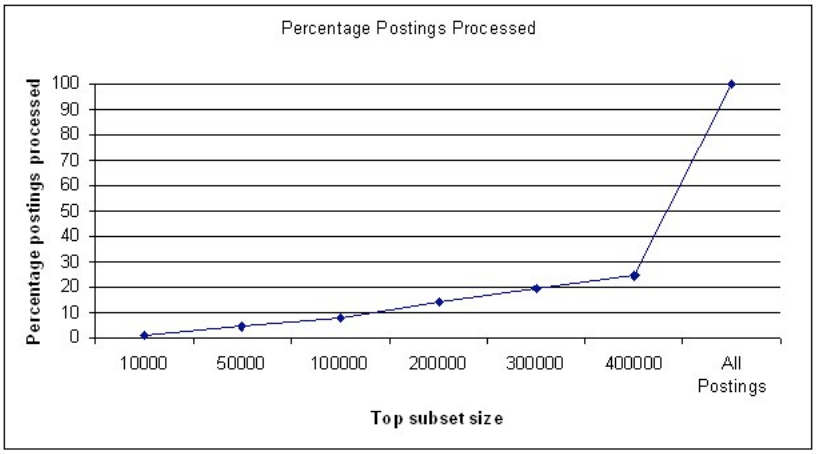

Figure 3: Percentage of postings processed with different top subset sizes

5 Conclusions

In our participation in the TREC 2006 terabyte ad hoc search task, our main objective was to further investigate the effectiveness of our sorted index approach in order to allow tradeoffs between the number of postings that are processed in response to a query, and retrieval effectiveness. To this end the approach we presented once again seems to offer a viable way of reducing the search space with only a 0.0366 absolute value reduction in MAP and a 0.0061 absolute value reduction in P10, having to process only 13.9% of all postings available.

In terms of our retrieval performance in comparison with other groups in the TREC terabyte track in 2006, this year our submitted run performed below the median for MAP and bpref, and above the median for p10. This is contrary to the terabyte track in 2005 where all of our ad hoc runs performed above the median for MAP, p10 and bpref, when compared with other participants [7, 6]. However If we were simply concerned with system performance in terms of these measures then by processing all postings, this would give performance that would be above the median in terms of MAP, as shown in Figure 1. Future work may include experimenting with both alternative forms of sorting, as well as document ranking, using functions other than BM25, in order to increase the overall performance of the system.

Acknowledgement: This work was supported by Science Foundation Ireland, under grant number 03/IN.3/I361.

References

[1] V. N. Anh, O. de Kretser, and A. Moffat. Vector-Space Ranking with Effective Early Termination. In *SIGIR '01: Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR, 2001.

[2] V. N. Anh and A. Moffat. Impact transformation: effective and efficient web retrieval. In *SIGIR '02: Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 3–10, 2002.

[3] V. N. Anh and A. Moffat. Improved retrieval effectiveness through impact transformation. In *CRPITS '02: Proceedings of the Thirteenth Australasian Conference on Database Technologies*, pages 41–47, Darlinghurst, Australia, 2002. Australian Computer Society, Inc.

[4] S. Blott, O. Boydell, F. Camous, P. Ferguson, G. Gaughan, C. Gurrin, N. Murphy, N. E. O. Connor, A. F. Smeaton, B. Smyth, and P. Wilkins. Experiments in Terabyte Searching, Genomic Retrieval and Novelty Detectionn for TREC-2004. In *Proc. Third Text REtrieval Conference*. Proc. Thirteenth Text Retrieval Conference (TREC-13), November 2004.

[5] C. Buckley and A. F. Lewit. Optimization of inverted vector searches. In *Proceedings of the 8th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 97 – 110, 1985.

[6] C. Clarke, F. Scholer, and I. Soboroff. The TREC 2005 Terabyte Track. In *The Text Retrieval Conference (TREC)*, 2005.

[7] P. Ferguson, C. Gurrin, A. F. Smeaton, and P. Wilkins. Dublin City University at the TREC 2005 Terabyte Track. In *Proc. Third Text REtrieval Conference*. Proc. Fourteenth Text Retrieval Conference (TREC-14), November 2005.

[8] P. Ferguson, C. Gurrin, P. Wilkins, and A. F. Smeaton. Físréal: A Low Cost Terabyte Search Engine. In *Proceedings of European Conference in IR*. Springer LNCS, March 2005.

[9] P. Ferguson, A. F. Smeaton, C. Gurrin, and P. Wilkins. Top Subset Retrieval on Large Collections using Sorted Indices. In *SIGIR '05: Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 599–600, 2005.

[10] S. Garcia, H. E. Williams, and A. Cannane. Access-ordered indexes. In *CRPIT '04: Proceedings of the 27th Conference on Australasian Computer Science*, pages 7–14, Darlinghurst, Australia, 2004. Australian Computer Society, Inc.

[11] M. Persin. Document filtering for fast ranking. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 339 – 348, 1994.

[12] M. Persin, J. Zobel, and R. Sacks-Davis. Filtered Document Retrieval with Frequency-Sorted Indexes. *Journal of the American Society of Information Science*, 47, 1996.

[13] S. E. Robertson, S. Walker, K. Sparck Jones, M. M. Hancock-Beaulieu, and M. Gatford. Okapi at TREC-3. In *Proc. Third Text REtrieval Conference (TREC-3)*, pages 109–126, 1994.