Dublin City University at the TREC 2005 Terabyte Track

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

Abstract

For the 2005 Terabyte track in TREC Dublin City University participated in all three tasks: Adhoc, Efficiency and Named Page Finding. Our runs for TREC in all tasks were primarily focussed on the application of "Top Subset Retrieval" to the Terabyte Track. This retrieval utilises different types of sorted inverted indices so that less documents are processed in order to reduce query times, and is done so in a way that minimises loss of effectiveness in terms of query precision. We also compare a distributed version of our Físréal search system [1][2] against the same system deployed on a single machine.

1 Introduction

As in the 2004 Terabyte Track the experiments were run on the GOV2 collection. This is a collection of over 25 million documents crawled from the .gov site in early 2004.

The 2005 Terabyte track consisted of three task:

1. **Adhoc:** 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 and asked to return a ranked set of the 10,000 most relevant documents for each of these topics.
2. **Efficiency:** The aim of the efficiency task was to provide a means for comparing efficiency and scalability issues in IR systems. For this task participants were given 50,000 topics that had been mined from the search logs of an operation search engine. For each of these topics the participants executed all topics and reported the top 20 results for each, along with the total query processing time for the full set.
3. **Named Page Finding:** For the named-page finding task participants were given a set of 252 topics, each of which specified a document by name. The goal of the task was to return this page as close to the number 1 rank as possible. For each of the topics the participants were asked to return their top 1000 results.

In this paper section 2 will outline the search engine setup that was used to run our experiments. This will describe the different types of indices that we used as well as the two system architectures that we employed. Section 3 will describe our experiments for each of the tasks: Adhoc, Efficiency and Named page finding. Finally in section 4 we will draw conclusions from our experiments.

2 Físréal Search Engine Setup

Our runs for TREC 2205 in all tasks were mainly focussed on the experimentation with different types of sorted inverted indices that allow only a subset of documents associated with each term in the query to be evaluated and still retain effective search results, while at the same time processing the query much quicker (as the number of documents being considered for each term is reduced). This process of "top subset retrieval" has been explained in greater detail in [3]. This type of retrieval requires an index that sorts the document IDs and their corresponding term frequency for each term so that the most important documents are towards the top of the list for each term. For these experiments we utilised three different types of sorted indices to evaluate this process. These types of indices are described in detail in the sections 2.1.1, 2.1.2 and 2.1.3.

Another element that we wished to investigate in our experiments was whether we could utilise the text within certain HTML markup elements in the document, that when combined with standard text results could give us an improvement over the standard text only. For this we constructed separate indices for certain tags as described in section 2.1.4. We provided runs in both the adhoc and named page finding tasks to investigate this.

For these experiments (and most specifically for the efficiency task) we also wanted to compare our previously used distributed system architecture [1][2] and compare its performance with the same system running on a single machine. An outline of both these architectures: distributed and single machine, is given in section 2.2.1 and 2.2.2 respectfully.

2.1 Inverted Index Types

We created different inverted index files for terms occurring in documents and sorted the entries (document IDs and their corresponding term frequency) in different ways as described below.

2.1.1 BM25 Sorting

This BM25 index consists of a sorted list of Document IDs for each term by using the Okapi BM25 value [5]. This should provide an inverted index that for each term has a sorted list of document IDs and their associated term frequency, with the most influential documents for a given term at the top of this list. This type of index acts as a baseline index to compare the performance of our other indices against, as it uses a standard form of document ranking, in the postings list or in the final retrieval.

2.1.2 WeightedTF Sorting

The WeightedTF index is sorted using a metric known as weightedTF sorting, which was introduced in [3] and is defined as follows:

$$Weight_{tf} = \log(BiDist_{avg} + e) \times log(TF + e) \tag{1}$$

where e is the base of natural logarithms and $BiDist_{avg}$ is a measure of the distance of each document from the average document length, calculated as:

$$BiDist_{avg} = \begin{cases} \frac{dl_t}{avg_{dl}} & \text{if } dl_t \leq avg_{dl} \\ 1 - \frac{dl - avg_{dl}}{max_{dl} - avg_{dl}} & \text{otherwise} \end{cases} \tag{2}$$

where avg_{dl} is the average and max_{dl} is the maximum document length in the collection.

This sorting scheme works as follows: $BiDist_{avg}$ is a measure of the distance of the document length from the average document length, and is smaller the further the document length deviates from the average. When this is combined with the normalised TF to calculate $Weight_{tf}$, it gives a good measure of the term's overall influence on the document. Again for each term in the inverted index the document IDs and their associated term frequencies are sorted based on this measure so that the more important documents are towards the top of the list. This provides a mechanism to process only a subset of these documents for each term in the query.

2.1.3 Document ID Sorting

As described by Najork & Wiener in [4] a breadth-first crawl (as used to crawl the GOV2 collection) the higher quality pages are discovered early in the crawl. Assuming this to be true we could factor this into a metric to sort documents by. The GOV2 collection is organised into 274

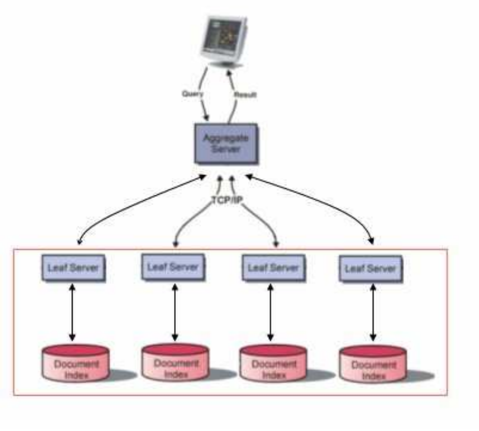

Figure 1: Distributed Search Architecture.

separate directories (000 - 273) which reflect the order in which the documents were crawled i.e. the documents in the lower number directories were crawled first. The GOV2 document IDs are of the form “GX000-01-0000000” where in this case the “GX000” part of the ID represents that this came from the 000 directory (and so was found early in the crawl). When creating the index we sorted the document IDs and their associated term frequencies for each term based on the following metric:

$$DID_{weight} = \frac{1}{\log(\alpha + e)} \times \log(tf + e) \tag{3}$$

where α represents the number of the directory that the document came from, e is the base of the natural log and *tf* is the term frequency of the document for that particular term.

This essentially gives all the documents from the same directory the same weighting, but gives an increased weighting to the folders with lower numbers (i.e. those that are crawled first) and penalises those that are crawled later. This is then combined with the normalised term frequency(*tf*) of the term in the document to reflect the term's influence in the document, as well as the quality of the document.

2.1.4 Tag indices

It is believed that certain HTML tags contain text that is more representative of the content of the document than other text in the document. For example, text in the title tag would generally be more reflective of the content of the document than the body text of the document, and should therefore be given more weighting in retrieval. In order to infer this extra weighting in addition to the sorted indices that have been described we also created separate indices for the text contained in the following HTML tags: Bold, Italic, Title, Underline, Meta. We considered other tags to be included however based on our experiments on the TREC Terabyte 2004 Adhoc task we used this final set of tags.

2.2 Search Engine Architecture

2.2.1 Distributed

Using a distributed architecture we distributed the GOV2 collection evenly across four machines, each being a Pentium 4, 2.6GHz with 1GB of RAM. The queries were handled by an aggregate machine which broadcast the queries to each of the four search machines (or “leaf servers”), who processed the query and send back their results to the aggregate machine. The aggregate then merged these results and produced the final output.

The interaction between each of the machines and a “user” machine can be seen from Figure 1.

2.2.2 Single Machine

In contrast to the distributed set up described previously we felt it would be an interesting comparison to search the collection using a single machine. For this we used a Pentium 4, 2.6GHz with 2GB of RAM. The entire collection was indexed on this machine and queries were run locally on this machine.

3 Terabyte Experiments

3.1 Adhoc Task

For this task we submitted four automatic runs (processed with no human intervention), using only the title text for each topic. All of these TREC runs used the distributed architecture (as in section 2.2.1). Our first run (DCU05ABM25) used a BM25 sorted index (as described in section 2.1.1). Our second run (DCU05ADID) uses a Document ID sorted index (section 2.1.3). Our third run (DCU05AWTF) uses a WeightedTF sorted index (section 2.1.2).

All the above runs achieved the same average query time of 2.4 seconds. Each of these runs processed a maximium of 100,000 documents for each term in the query using the approach of "top subset retrieval" described previously.

For our final adhoc run (DCU05ACOMBO) we aimed to combine the results from multiple inverted indices outlined in section 2.1.4, with the results of our baseline text results (i.e. DCU05BM25). It was hoped that we could combine the results from these multiple HTML tags in order to give an increase in performance. We combined these different results together using weighted fusion, in which we gave each of the results from the separate tag indices specific weights according to their importance. These weights for each set of results were chosen based on training runs using the Terabyte 2004 Adhoc topics and query relevance judgments. Although with an average query time of 19.4 seconds this run took much longer than the previous runs, we were more concerned with achieving an improvement over the baseline DCU05BM52 run. As can be seen from Table 1 this run achieved a slight increase over the baseline in terms of MAP and BPREF. We also present a Precision-Recall graph, showing the precision-recall tradeoffs of all our runs for the adhoc task in Figure 2.

Table 1: Adhoc Runs and Performance

Run Name	**MAP**	**BPREF**	**P10**
DCU05ABM25	0.2887	0.3098	0.588
DCU05ADID	0.2886	0.3154	**0.594**
DCU05AWTF	**0.3021**	**0.3267**	0.592
DCU05ACOMBO	0.2916	0.3191	0.578

3.2 Efficiency Task

For the efficiency task we had two runs (DCU05DISTDID & DCU05DISTWTF) that used the distributed search system where the first run accessed a document ID sorted index (section 2.1.3) while the latter used a WeightedTF sorted index (section 2.1.2). Each of these runs processed a "top subset" of at most 50,000 documents for each term in the query. Then, as a comparison to the distributed setup, our other runs (DCU05WTF & DCU05WTFQ) were both run on a single machine and both used a WeightedTF sorted index. DCU05WTF processed a subset of at most 50,000 documents for each term in the query, while DCU05WTFQ only processed a maximium of 20,000 documents per term, in order to provide a faster throughput of queries.

As the top 20 results for each topic were to be returned to NIST for evaluation we present the results for precision at 20 (P20), as well as the query time for each run in Table 2.

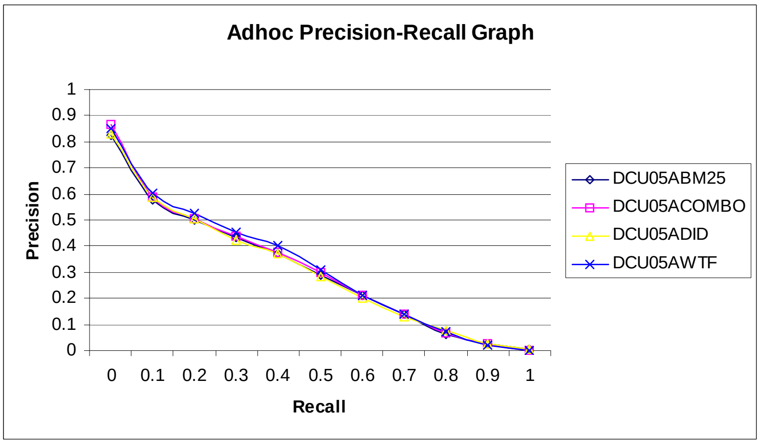

Figure 2: Precision Recall Graph for Adhoc Runs.

Table 2: Efficiency Runs and Performance

Run Name	**P20**	**Query Time(seconds)**
DCU05DISTDID	0.5210	0.97
DCU05DISTWTF	**0.5290**	0.97
DCU05WTF	0.4880	0.87
DCU05WTFQ	0.4660	**0.35**

3.3 Named Page Finding Task

For this task we used a similar approach to the adhoc task to investigate if the method of ”top subset retrieval” is conducive to the task of finding a specific document.

As with the adhoc task three of our runs (DCU05NPBM25, DCU05NPDID & DCU05NPWTF) were conducted using our distributed architecture (section 2.2.1), each of which accessed a different sorted index: DCU05NPBM25 used the BM25 sorted index, DCU05NPDID used the Document ID sorted index and DCU05NPWTF used the WeightedTF sorted index. Also, akin to our adhoc task, we submitted a run that combined the results of our baseline run (DCU05NPBM25) with the results from multiple indices (as described in section 2.1.4), using weights trained on the adhoc topics from 2004.

The performance figures for these runs can be seen in Table 3

Table 3: Named Page Finding Runs and Performance

Run Name	**Recip. Rank**	**Num. in Top 10**
DCU05NPBM25	**0.32**	**115 (45.6%)**
DCU05NPDID	0.256	98 (38.9%)
DCU05NPWTF	0.254	96 (38.1%)
DCU05NPCOMBO	0.287	100 (39.7%)

4 Conclusions

Our chief objective in these experiments was to further evaluate the effectiveness of the approach of “top subset retrieval” in the Terabyte Track in TREC 2005. In the adhoc task our experiments were quite effective as all runs performed better in terms of MAP, P10 and Bpref than the average of the results from all participants, while our combination run which merged results from multiple tags gained an increase in performance over our baseline run.

The efficiency task provided us an ideal opportunity to evaluate and contrast the performance of a distributed version of our system using five machines against a similar system on a single machine. The runs DCU05DISTWTF and DCU05WTF provide a direct comparison between the two types of systems. Although the distributed system achieves a higher precision at 20(P20) of 0.529 as opposed to 0.488 on the single machine, the single machine provides a faster query throughput of 0.87 seconds, as opposed to 0.97 seconds. Our "quick" run DCU05WTFQ also achieved satisfactory results: The difference between this run and the DCU05WTF was that the former accessed a maximum top subset of 20,000 documents per term, as opposed to 50,000, in order to process queries faster. It achieved this goal with reduction in query time from 0.87 seconds to 0.35 seconds with a relatively small drop off of 0.022 in P20.

With our participation the named page finding task we aimed to investigate if the process of "top subset retrieval" was suitable for returning a specific page. Our performance figures for this task (in term of reciprocal rank) lie slightly below that of the average figures achieved by other groups. The explanation for this would be in large part be due to the use of a rather small top subset of 100,000 documents for each term. This would mean that if the relevant document was not ranked in the top 100,000 documents for any of the terms of the query then it would not be scored for retrieval. This could easily be remedied by choosing to process a larger subset of documents for each term, however this would also adversely affect the query time (depending on the size of the subset chosen). Also the fact that our combination run for this task (DCU05NPCOMBO) performed below that of the baseline of (DCU05NPBM25) may have been due to the fact that the weights chosen to merge the results from different tag indices were trained on the 2004 adhoc topics. As both the tasks and the two types of queries vary i.e. the topics for adhoc are often quite vague and are generally relevant to several documents, in contrast to the much more concise queries for the named page finding task that are relevant to only one page. There is most likely gains to be made by experimenting with the way in which these sources of information are merged, as well as possibly incorporating other sources of information such as anchor text.

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

References

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

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

[3] 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.

[4] M. Najork and J. L. Wiener. Breadth-First Crawling Yields High-Quality Pages. In *Proceedings of the 10th International World Wide Web Conference*, pages 114–118, Hong Kong, May 2001. Elsevier Science.

[5] S. E. Robertson and S. Walker. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *SIGIR '94: Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 232–241, New York, NY, USA, 1994. Springer-Verlag New York, Inc.