Melbourne University 2005: Enterprise and Terabyte Tracks

Vo Ngoc Anh William Webber Alistair Moffat

Department of Computer Science and Software Engineering
The University of Melbourne
Victoria 3010, Australia

Abstract: *This report describes the work done at The University of Melbourne for the TREC-2005 Enterprise and Terabyte Tracks. In the Enterprise Track, we participated in the Discussion Task. We tried a number of different methods to make use of special features of mailing lists to improve retrieval effectiveness, and found the use of thread context to be promising. In the Terabyte Track, we continued our work with impact-based ranking and sought to reduce indexing as well as query time. A new retrieval system has been developed for this purpose and has shown several improvements over our system from last year.*

1 Introduction

In TREC 2005, The University of Melbourne participated in the Enterprise and Terabyte Tracks.

In the inaugural Enterprise Track, we took part in the Discussion Task. A baseline run was made on an index from which all quoted text had been stripped. The document scores in this baseline run were then supplemented, first by scores from a parallel index of the quoted text, then by scores of other messages in the same thread, and finally by the frequency with which the message's author posted to the W3C mailing lists. A separate run was made using an impact-based system. In the event, the official results showed the impact-ordered run to be superior to the chosen baseline, and retaining quoted text to be superior to removing it. Enhancing document scores with thread information was a promising technique.

In the Terabyte Track, we participated in the Ad-hoc and Efficiency Tasks. For both tasks, a retrieval system employing impact-based ranking was used, modified since last year to improve efficiency. For the Ad-hoc Task, runs using anchor text and query term proximity were made. For the Efficiency Task, the system was run both on a single machine and on an eight-machine cluster. Our results indicate a significant improvement in efficiency compared to last year, as well as modest effectiveness gains.

2 Enterprise Track

We participated in the Discussion Task of the Enterprise Track. Two base runs were made, using different ranking metrics and software. One of these base systems was then used as a starting point for three different enhancements. As it turned out, the less effective of the two base systems, as assessed by the official relevance judgments, was the one used for the experimental enhancements. Also, the approach taken of stripping quoted text (usually verbatim parts of the thread of messages being replied to) before indexing was not vindicated by the final effectiveness scores. On the other hand, the experimental technique of enhancing an email's score by reference to the surrounding thread does seem to offer effectiveness gains.

Subsequent investigation discovered a number of errors in our runs. First, while roughly 90% of the documents in the collection follow the same consistent template for marking up mailing list posts in HTML, another 10% do not. Our preprocessing tool was designed assuming that the said template was used universally, and produced either garbled or empty documents for the 10% of posts that did not follow this format, something we did not discover until after our runs were submitted. Second, our parameters were hand-trained using our own set of document judgments and an older version of the `trec_eval` tool; however, we later discovered that this version of `trec_eval` did not properly handle the slightly different format of qrel file specified for the Discussion Track, and silently produced incorrect evaluation scores. As a result, we mis-trained our parameters. These problems with deviant formats seriously degraded the effectiveness of our runs.

2.1 Run Descriptions

The runs described below allow for a number of different metric tuning parameters. In order to determine suitable values for these parameters, ten questions were selected from the track topics, and around seventy documents for each were judged prior to run submission, to provide sample qrels for evaluation purposes.

`Base` (`MU05ENd4`). Quoted text was stripped from emails before indexing, so that only the content inserted by that author was matched. The scoring metric used was a language model with Dirichlet smoothing, as described by Zhai and Lafferty [2004], in the context of the Zettair retrieval system (see `www.seg.rmit.edu.au`). This run was used as the basis for the following three variants.

`Base+Quote` (`MU05ENd2`). For this run, a parallel corpus was created from all of the quoted text. This corpus was then separately indexed. Each document was scored using a weighted sum of its scores for the base and parallel indexes. The idea was that, although quoted text was not meant to be directly considered when judging whether a document was relevant, it would nevertheless help indicate the subject of the main body of the email. Once the weighting parameter was tuned, the sample evaluations suggested that this was the second most effective technique.

`Base+Threads` (`MU05ENd1`). The idea behind this run was that an email message is more likely to be relevant to a topic if the thread it is part of is relevant. The thread structure of each mailing list was analyzed and separately stored. The score for each document was then enhanced at query time by the scores of the ancestor and descendant emails in its thread, with the contribution decaying the more distant the ancestor or descendant was. Properly tuned, this metric did best in the sample evaluations.

`Base+Author` (`MU05ENd3`). A simplistic attempt at an authority metric, this run was prepared more out of interest than with the expectation that it would perform very well. The number of messages posted by each author (as identified by email address) was calculated as a proportion of the total number documents in the collection. At query evaluation time, each email's score was then increased by this much percent, multiplied by a tunable parameter. The sample judgments showed this method as providing only a slight improvement over the base run.

`Impacts` (`MU05ENd5`). This run was made with a different system, based on an impact-ordered index, and the similarity computation used in the Terabyte Track, see the description below of run `MU05TBa1`. The same collection, stripped of quoted text, was used as for `Base`. Because of delays in readying the software, this run was not assessed during the sample evaluation process.

Run	MAP
Submitted runs:	
Base	0.1834
Base+Quote	0.2031
Base+Threads	0.2085
Base+Author	0.1708
Impacts	0.2182

Run	MAP
Post-judgment runs:	
NoQuoted	0.2132
Text	0.2641
HTML	0.2746
HTML+Threads	0.2951

Table 1: *Mean average precision for the five submitted Discussion Track runs, and for four subsequent experiments using the relevance judgments.*

After the judgments were released, and indeed after the TREC conference itself, it became apparent that something was quite wrong with our runs (see the previous subsection), and we performed further runs with a corrected corpus and re-trained parameters. The Okapi BM25 metric was used for all of these runs, as our experiments with the final judgments and corrected corpus and `trec_eval` found it superior to the language model used for the submitted runs. These runs were as follows:

`NoQuoted` Message bodies were extracted from the HTML document collection, and all quoted text was stripped off, similar to the submitted `Base` run.

`Text` Message bodies were extracted from the HTML document collection, with the surrounding HTML markup discarded. Similarly, indexing pages in the HTML collection were discarded entirely. Quoted text was neither stripped out nor indexed separately, but left in place.

`HTML` The HTML corpus was indexed directly, with no pre-processing.

`HTML+Threads` The HTML corpus was indexed directly, and message thread information was used as for the submitted `Base+Threads` run.

2.2 Results

Table 1 gives the official results for the five submitted runs, as well as unofficial results for four further runs that were computed post-judgment using the official qrels. Of the five submitted runs, the best result was achieved by `Impacts`. The ordering of the others is similar to that predicted by the sample evaluations, except that `Base+Author` did not do as well as `Base` alone. Tuning the various metric parameters against the official judgments, rather than the sample ones, can improve these first four scores slightly, but does not change the ordering.

The post-submission runs reveal the problems with our submitted runs caused by corpus-parsing errors, as can be seen by the improved MAP for `NoQuoted` over the equivalent submitted `Base` run. However, they also demonstrate fundamental problems with our approach. The far superior retrieval effectiveness of `Text` over `NoQuoted` shows that stripping out quoted text is a very bad idea, even if quoted text is not meant to directly contribute to an entry's relevance (whether this requirement was strictly adhered to in the judging process is another question). Even more surprising, indexing the raw HTML, with all of its repeated and extraneous markup, and even its index pages, is more effective than stripping down just to the email bodies. Part of the reason is that a number of index pages have been incorrectly assessed as "relevant" in the judging process. But in addition, the presence of the email subject, and its repetition in links to follow-up messages, appears to be a significant bonus, whereas the portions of the HTML template that are repeated

on every page are correctly down-weighted by the metric scheme. Fortunately, these results do demonstrate that taking thread information into account is still a very useful technique.

We note in passing that the crawl made of the W3C mailing lists is incomplete. In particular, where a four-month period for any given list had more than 500 messages, only the last 500 were included in the collection. This had the effect, apart from anything else, of decapitating threads on some of the most active lists. It is possible that with a complete collection, and therefore complete thread information, the `+Threads` enhancement technique would perform even better.

3 Terabyte Track

In this year's Terabyte Track, we participated in the Ad-hoc and Efficiency Tasks using locally-developed software.

3.1 Retrieval Approach

In the 2004 Terabyte Track we employed an impact-based ranking technique [Anh and Moffat, 2005b] as the retrieval mechanism. In the framework of the track, the technique turned out to provide an excellent combination of retrieval efficiency and retrieval effectiveness.

Impact ranking specifies a method to map each pair (t,x), where t is a term in text x, to an integer impact value that represents the importance of t in x. When x is a document d from a document collection, the resultant impact $\omega_{d,t}$ is referred to as the *document term impact* of t in d. When x is a query q, the value $\omega_{t,q}$ is the *query term impact* of t in q. Document and query term impacts normally correlate with the term frequencies in documents or queries, but might or might not also depend on some collection-wide statistics such as collection frequency f_t.

In contrast to conventional information retrieval systems, where raw term statistics are stored in indexes, and most of the similarity calculation is carried out during query processing, impact-based ranking allows computation of all of the document term impacts when the index is being constructed. The document term impacts are then stored in the index, reducing the computational burden during query processing. Normally, the indexes are impact-sorted.

This year, we continued our investigation of impact-based ranking, focusing mainly on further improving efficiency, even where that risked possible degradation in effectiveness. A concern with the impact indexing scheme used last year was the need to read the whole input document collection twice in order to build the index. The underlying reason was that document term impacts for a term t depended on the collection-wide IDF factor f_t. To facilitate faster indexing, this year we employed an impact scheme that allowed the computation of document term impacts without referring to the IDF factor or to any other collection-wide statistics. This method of assigning impacts is reported in Anh and Moffat [2005b] under the name of *Local-By-Rank-(TF)*. With *Local-By-Rank-(TF)* impacts, the indexing can be done by reading the text document collection just once.

Note that although the indexing does not depend on any collection-wide statistics, the IDF factor is still used in ranking. At query time, the IDF factor and query term frequencies are taken into account when calculating query term impacts. Then, a scoring computation is performed based only on the integral document and query term impacts. No floating-point operations are required, and no document weights are used or even computed.

3.2 Index Structures

For a document collection, the principal component of its index is the inverted file, where each distinct term of the collection is associated with an inverted list. For this Terabyte Track, two different inverted list structures were employed:

impact-sorted: The impact-sorted inverted list for a term t is a list of equal-impact blocks. Each block represents one distinct impact value ω and contains the sequence of document numbers in which t appears and has an impact score of ω. Inside a block, document numbers are arranged in increasing order to facilitate compression of these values. The blocks are arranged in decreasing order of associated impacts to support effective pruning.

impact-positional: Here, the inverted list for t is similar to that of a conventional positional inverted list, where a document d in the list is also accompanied by a sequence of positions of t in d. The elements of each inverted list are sorted in increasing order of document numbers, and no equal-impact blocking is performed. Unlike conventional positional lists, however, each document d in the inverted list of t is also accompanied by the document impact $\omega_{d,t}$.

Compression is applied to inverted files. In all of our Terabyte Track experiments a word-aligned compression scheme [Anh and Moffat, 2005a] was used for inverted list compression. This method provides a good balance between index space and decoding speed.

3.3 Hardware Configurations and Efficiency Metrics

Two hardware configurations were used in our experiments this year:

`Single`: A machine with a single 2.8 GHz Intel Pentium-4 processor, 1 GB of memory and 250 GB of local SATA disk, running Debian GNU/Linux.

`Cluster`: A Beowulf-style cluster, consisting of a server and eight additional nodes, where each node is a `Single`. The server is a dual 2.8 GHz Intel Xeon with 2 GB of memory, again running Debian GNU/Linux. In this configuration, the document data are uniformly distributed across the nodes in a cyclic manner. Indexing and querying are done in parallel, without communication between the nodes, in an autonomous document-distributed manner. The server only plays the role of a broker: it receives queries, broadcasts queries to the nodes, receives the answer lists from the nodes, and selects the final document output based on the local similarity scores computed by the nodes.

The efficiency metrics reported in our experiments include *Index Time* (elapsed time for indexing, not including any time needed to distribute documents to the nodes), *Index Size* (total size of the index, including vocabulary and inverted files) and *Query Time* (average elapsed time to process a query). Over a sequence of queries, the total of *Query Time* was measured from the moment when the first query arrived, until the last query had been processed, not including the initialization costs associated with loading a range of memory-resident files. Note that in this implementation queries are processed sequentially, with a single query active in the system at any given time. Further throughput gains would be possible by multi-threading the system, so that, for example, CPU idle periods arising from disk operations in one process could be exploited by another process.

3.4 Effectiveness Performance

The following four runs were submitted for the Ad-hoc Task. All the four runs were conducted using the hardware configuration `Single`.

`Base` (`MU05TBa1`). This is the baseline, performed with a standard impact-sorted index, using the *Local-By-Rank-(TF)* computation [Anh and Moffat, 2005b]. For this run, all the normal content of documents are indexed. No special treatment was applied to any fields (meta, title, heading, and so on). Incoming anchor text is not considered.

RunID	Efficiency			Effectiveness				
	Index size (GB)	Index time (minutes)	Query time (secs)	R.Rank	P@10	P@20	MAP	bpref
Base	5.99	350	0.2	0.8240	0.6260	0.5760	0.3199	0.3366
Base+Anchor	6.07	749	0.2	0.8086	0.6200	0.5730	0.3218	0.3399
Base+Prox	38.79	438	0.9	0.8174	0.6360	0.5760	0.3063	0.3264
Base2	5.99	350	0.2	0.8228	0.6260	0.5760	0.3092	0.3267

Table 2: *Performance in the Ad-hoc Task. All results relate to the GOV2 collection.*

Run ID	Efficiency			Effectiveness		
	Index size (GB)	Index time (minutes)	Query time (sec)	R.Rank	P@10	P@20
BaseSingle	5.99	350	0.1983	0.8069	0.6000	0.5480

Table 3: *Performance of Efficiency runs in the single computer. All results relate to the GOV2 collection.*

Base+Anchor (MU05TBa2). This run differs from the baseline by also considering incoming anchor text. The incoming anchor text is simply treated as normal text within the target document. That is, incoming anchor text redefines the within-document frequencies, which in turn result in different document term impacts being assigned.

Base+Prox (MU05TBa3). This run is similar to the baseline, except that positions of terms in documents are taken into account. The index is document-sorted rather than impact-sorted (that is, it is an impact-positional index). Impact scoring is also applied, but the total score for a document is enhanced by another integer score which is calculated based on the inverse of the least distance in words between the positions of any of the query terms in the document.

Base2 (MU05TBa4). This run is almost identical to the baseline run. The intention for the run was to make some change in the way impacts were defined. Unfortunately, we subsequently discovered an error that made the intended changes almost silent. For completeness, we report the result for this run, but do not discuss it.

Table 2 shows the performance of these Ad-hoc runs. In terms of effectiveness, all of the runs had similar performance. On the other hand, the baseline run is slightly better than others, suggesting that the methods chosen to integrate incoming anchor text and positional information were ineffective.

3.5 Efficiency Performance

In addition to the Ad-hoc runs, the following four runs were submitted for the Efficiency Task:

BaseSingle (MU05TBy2). This is the baseline run for the Efficiency Task. It has the same setting as MU05TBa1 – the baseline of the Ad-hoc runs. However an oversight in selecting the documents meant that the final list of documents differed slightly from that of MU05TBa1. The run was performed using the hardware configuration Single.

Run ID	Efficiency			Effectiveness		
	Index size (GB)	Index time (minutes)	Query time (sec)	R.Rank	P@10	P@20
BaseCluster	6.89	44	0.0429	0.8169	0.6360	0.5620
Cluster+Prune	6.36	44	0.0240	0.8095	0.6280	0.5550
Cluster+2004	7.18	104	0.0884	0.7646	0.5540	0.5030

Table 4: *Performance of Efficiency runs in the cluster. All results relate to the GOV2 collection.*

BaseCluster (MU05TBy1). This run had the same setting as the Base-Single. The only difference is that it was conducted using the hardware configuration Cluster. Retrieval effectiveness changed slightly because in the cluster environment the IDF components are local to each node.

Cluster+Prune (MU05TBy3). This run had Base-Cluster as a starting point, but used a static index pruning technique, under which the low-impact blocks were excluded from the index at index construction time.

Cluster+2004 (MU05TBy4). This run employed the software that we used last year in the Terabyte Track, in order to quantify the level of improvement obtained by the software re-write.

Efficiency performance on a single-computer system is described in Table 3 (and also, for the effectiveness runs, in Table 2). Using a single commodity machine, we can index the GOV2 collection in under six hours, at a rate of approximately 70 GB per hour; and can process queries at the rate of approximately five per second. Moreover, the index is small – just 1.4% of the size of the original document collection.

In terms of efficiency, Table 2 reveals the disadvantage of using anchor text and positional index information. Using anchor text more than doubles the indexing time, while using an impact-positional index increases the index size by a factor of six, and the query time by a factor of five. To retain a high level of efficiency, innovative methods for handling the anchor text and word positions are required.

Table 4 compares the performance of the three runs which were performed on the cluster. The comparison between BaseSingle (in Table 3) and BaseCluster shows the benefits of using the cluster – indexing time was cut by a factor of almost eight, and query time by a factor of approximately five. Table 4 also shows that static pruning (method Cluster+Prune) helps to reduce the query time significantly, with only a small degradation in effectiveness.

Finally, Table 4 shows a pleasant result for our group when comparing last year's system with this year's system. Lines BaseCluster and Cluster+2004 demonstrate that this year's system is twice as fast as last year's, in terms of both indexing time and querying time, vindicating the considerable effort that went into the software re-write. This year's system also has slightly better effectiveness.

Figure 1 illustrates the relative performance of our system in the Efficiency Task. The graphs were compiled from data covering the 16 runs submitted by the eight groups with the best scores according to the metric P@20, as reported in Table 4 of Clarke and Scholer [2005]. Two of our runs from these 16 are BaseCluster and Cluster+Prune, which were conducted on the cluster. To complete the picture, the run BaseSingle is added, to show the performance of our the system on the single CPU configuration. Recall that apart from using the single CPU system, the run BaseSingle is similar to BaseCluster in all other aspects. Figure 1 demonstrates that our system is highly competitive in terms of both efficiency and effectiveness.

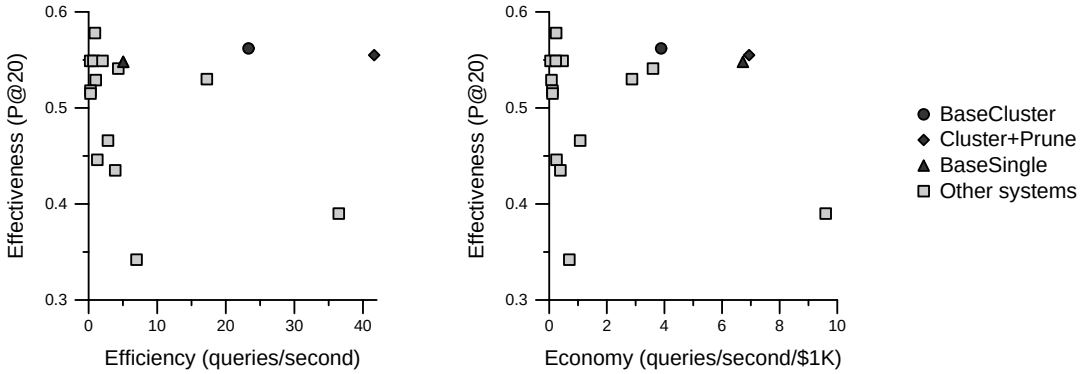

Figure 1: Relative performance in the Efficiency Task of the Terabyte Track (using the GOV2 *collection), with effectiveness compared to efficiency (left-hand graph) and to economy (right-hand graph), the latter calculated as throughput normalized by estimated system cost (in terms of purchase price converted to mid-2005 $US). Note that efficiency is measured over* $50{,}000$ *real-life queries; effectiveness over a subset of* 50 *queries.*

4 Future Directions

We intend to complete the construction of the new retrieval system and undertake further investigations in connection with anchor text and word positions. In the field of document retrieval from email corpora, we are interested in further exploring the use of thread context and summarization.

Acknowledgment This work was supported by the Australian Research Council and the ARC Center for Perceptive and Intelligent Machines in Complex Environments.

References

V. N. Anh and A. Moffat. Inverted index compression using word-aligned binary codes. *Information Retrieval*, 8(1):151–166, January 2005a.

V. N. Anh and A. Moffat. Simplified similarity scoring using term ranks. In G. Marchionini, A. Moffat, J. Tait, R. Baeza-Yates, and N. Ziviani, editors, *Proc. 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 226–233, Salvador, Brazil, August 2005b. ACM Press, New York.

C. L. A. Clarke and F. Scholer. The TREC 2005 Terabyte Track. In *The Fourthteenth Text REtrieval Conference (TREC 2005) Notebook*, Gaithersburg, MD, November 2005. National Institute of Standards and Technology. `http://trec.nist.gov/act_part/t14_notebook/t14.notebook.html`.

C. Zhai and J. Lafferty. A study of smoothing methods for language models applied to information retrieval. *ACM Transactions on Information Systems*, 22(2):179–214, April 2004.