# Melbourne University at the 2006 Terabyte Track

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-2006 Terabyte Track. For this track, we participated in all three main tasks. We continued our work with impact-based ranking and sought to reduce indexing as well as query time. However, to support the named-page task, more conventional retrieval mechanisms were also employed. The results show that, in general, the efficiency performance is slightly better than the previous year. The effectiveness level remains the same.*

## 1 Introduction

In TREC 2006, The University of Melbourne participated in three tasks of the Terabyte Track: ad-hoc, efficiency, and named-page. For the first two tasks impact-based ranking was employed. For the named-page task, BM25 was also used. All the experiments were performed using our locally-developed software. The system was developed last year and enhanced further with new index compression schemes and index structures.

## 2 Ranking Schemes

This year, we continue to employ an impact-based ranking technique [Anh and Moffat, 2005] as the main retrieval mechanism. In the framework of the track, the technique turned out to provide an excellent combination of retrieval efficiency and retrieval effectiveness. However, for the named-page task, we also applied the BM25 model.

### 2.1 Impact-Based Ranking

This sub-section expands the description of impact-based ranking described in our 2005 TREC paper. 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 $\omega_{t,d}$ 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 $f_t$.

The mapping from each term-document pair ($t$,$d$) to the corresponding impact $\omega_{d,t}$ is done locally within each document during the indexing phase in the following manner. First, frequencies of terms appearing in $d$ are counted. Then, the list of within-document terms is sorted in decreasing order of frequencies. Next, the list is then partitioned in $k$ intervals, with interval lengths forming a geometric sequence. Finally, all terms in the $i$-th interval ($1 \leq i \leq k$) are assigned the same integral impact value of $k - i + 1$. Since document term impacts do not rely on collection-wide statistics (in particular, they are independent of IDF), the indexing can be done with only one pass through the document collection.

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 while 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.

The mapping from each term-query pair ($t$,$q$) to the corresponding impact $\omega_{q,t}$ is done at query time, using within-query frequencies and collection frequencies. First, for each term $t \in q$, the collection frequency $f_t$ and within-query frequency $f_{q,t}$ are determined. Then, a vector-space formulation of within-query term weight is applied to generate the term weight $w_{q,t}$ (in our experiences, any vector-space formulation that involves IDF factor can be used for this purpose). Finally, the value of $w_{q,t}$ is transformed in a uniform manner, and truncated to an integral impact $\omega_{q,t}$, so that the maximal value of $\omega_{q,t}$ for all terms in the query is exactly $k$.

Once the document and query term impacts have been determined, the ranking of documents $d$ against a query $q$ is done based on the similarity score:

$$S_{d,q} = \sum_{t \in d \cap q} \omega_{d,t} \cdot \omega_{q,t} \ . \tag{1}$$

Note that although the indexing does not depend on any collection-wide statistics, the IDF factor is still used in ranking as an integrated part of the query impact. Also, the above 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. This makes the ranking process fast. Anh and Moffat [2005] also describe a ranking technique which is also employed in our runs to further enhance the querying speed.

### 2.2 BM25

For the named-page task, in addition to each original document, we also employ the incoming anchor text to it. This year, we simply add all incoming texts found in the collection to the respective document, creating a new document collection. This collection is then used as the input to the indexing process.

We noticed that after adding anchor text to the destinated documents, there might be a great change in within-document term frequency distribution. For example, a small number of terms might be boosted to a very high frequency level. That, in turn, might affect the above described process of assigning document impacts, and perhaps in a negative manner. To test this hypothesis, we also employ BM25 [Robertson et al., 1994]. The exact formulation used is:

$$S(d,q) = \sum_{t \in d \cap q} \ln \frac{N - f_t + 0.5}{f_t + 0.5} \cdot \frac{f_{d,t}}{0.5 + 1.5 \cdot W_d / W_d^{\text{avg}} + f_{d,t}} \ , \tag{2}$$

where $W_d$ is the size of $d$, measured in bytes, and $W_d^{\text{avg}}$ is the average value of $W_d$ over the collection.

## 3 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 the 2006 Terabyte Track, a number of 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 $k$, and contains the sequence of document numbers in which $t$ appears and has an impact score of $k$. Inside a block, document numbers are arranged in increasing order, to facilitate compression. The blocks are arranged in decreasing order of associated impacts, so as to support effective pruning.

*block-interleaved:* This is a common name for a family of document-sorted indexes described by Anh and Moffat [2006b]. In terms of index content there are three variants of indexes, depending on the components used to represent each pair $(d, t)$. Namely, the pair is represented by $(d, \omega_{d,t})$ in $I_{DI}$ indexes, by $(d, f_{d,t})$ in $I_{DF}$ indexes, by $(d, \omega_{d,t}, f_{d,t})$ in $I_{DIF}$ indexes, and by $(d, \omega_{d,t}, f_{d,t}, p^*)$ in $I_{DIFP}$ indexes. Note that the last variant is used for fully positional indexes, with $p^*$ representing the list of positions of $t$ within $d$. In terms of structure, each inverted list is arranged in the block-interleaved manner. For example, an inverted list of the $I_{\text{DIF}}$ is a sequence of index blocks, where each index block is a sequence of three fixed-length blocks, one for each of $d$, $\omega_{d,t}$, and $f_{d,t}$. In each block, the elements are sorted in increasing order of document numbers, and no equal-impact blocking is performed.

Compression is applied to inverted files. In all of our Terabyte Track experiments the word-synchronized compression scheme *slide* [Anh and Moffat, 2006a] was used for inverted list compression. This method provides a good balance between index space and decoding speed. In comparison with the compression scheme employed last year, *slide* in general has better compression effectiveness and roughly the same decoding speed.

## 4 Hardware Configurations and Efficiency Metrics

As in 2005, 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, and, for the case of using incoming anchor text, not including the time to locate and link the anchor text to the destinated document); *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 all but one run, queries are processed sequentially, with a single query being active in the system at any given time. There is one exceptional case in the efficiency task, where multi-threading (with 4 threads) was used so that CPU idle periods arising from disk operations in one process could be exploited by another process.

| RunID | Topics 801–850 | | | | |
|---|---|---|---|---|---|
| | MAP | R-prec | R.Rank | P@10 | P@20 |
| `AdhocBase` | 0.3039 | 0.3479 | 0.6969 | 0.5460 | 0.5130 |
| `AdhocBase+Prox` | 0.2888 | 0.3324 | 0.7150 | 0.5300 | 0.4860 |
| `AdhocBase,Prox` | 0.2926 | 0.3414 | 0.7067 | 0.5300 | 0.4880 |

| RunID | Topics 701–850 | | | | |
|---|---|---|---|---|---|
| | MAP | R-prec | R.Rank | P@10 | P@20 |
| `AdhocBase` | 0.3073 | 0.3491 | 0.7694 | 0.5765 | 0.5426 |
| `AdhocBase+Prox` | 0.3025 | 0.3422 | 0.7693 | 0.5711 | 0.5191 |
| `AdhocBase,Prox` | 0.3007 | 0.3448 | 0.7492 | 0.5651 | 0.5211 |

Table 1: *Effectiveness performance in the automatic ad-hoc task. All results relate to the* `GOV2` *collection.*

## 5 Ad-hoc Performance

Four runs were submitted for the ad-hoc task, including one manual and three automatic run.

### 5.1 Automatic Runs

In doing the automatic runs this year, we targetted the tie-breaking problem in impact ranking. Because the maximal value of impact, $k = 8$, is relatively small, the number of documents that have equal similarity score with respect to a query is potentially high. This year we experimented with using proximity for breaking ties. That is, the original impact score is further adjusted by a score calculated from the proximity of query terms in the document. The three submitted runs are:

`AdhocBase` (`MU06TBa2`). This is the baseline, performed with a standard impact-sorted index, using the *Local-By-Rank-(TF)* computation as described earlier. 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.

`AdhocBase+Prox` (`MU06TBa5`). 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.

`AdhocBase,Prox` (`MU06TBa6`). This run is the same as `Base+Prox`, except that the proximity score is employed just to break the tie of the main impact score. That is, the final ranking is performed by sorting the documents using impact score as the primary sort key, and proximity score as a secondary sort key.

Table 1 shows the effectiveness performance of these ad-hoc runs. In terms of effectiveness, all of the runs had similar performance. It suggests that the treatment by proximity distance fails to give the desired effects.

### 5.2 Manual Runs

`AdhocBase+Prox+Manual` (`MU05TBa1`). This run is almost identical to the `AdhocBase+Prox`, except that manual queries are used instead of automatic. Manual queries are developed by

| RunID | Topics 801–850 | | | | |
|---|---|---|---|---|---|
| | MAP | R-prec | R.Rank | P@10 | P@20 |
| `AdhocBase+Prox+Manual` | 0.2927 | 0.3186 | 0.8301 | 0.6160 | 0.5420 |
| `AdhocBase+Prox` | 0.2888 | 0.3324 | 0.7150 | 0.5300 | 0.4860 |

Table 2: *Effectiveness performance of the manual run in comparison with the similar automatic run. All results relate to the* `GOV2` *collection.*

| RunID | Hardware | Efficiency | | |
|---|---|---|---|---|
| | | Index size (GB) | Index time (minutes) | Query time (secs) |
| `AdhocBase` | `Single` | 6.04 | 334 | 0.16 |
| `AdhocBase+Prox` | `Cluster` | 38.60 | 57 | 0.21 |
| `AdhocBase,Prox` | `Cluster` | 38.60 | 57 | 0.20 |
| `AdhocBase+Prox+Manual` | `Cluster` | 38.60 | 57 | 0.31 |

Table 3: *Efficiency performance of all ad-hoc run. The fist run employed impact-sorted index, while all other runs used an $I_{DIFP}$ index.*

us based on our perception of the topics. In many cases, a simple process of adding some related terms to the query is applied. Consequently, manual queries are in general longer than respective automatic queries.

Table 2 compares the effectiveness of using manual queries with that of using automatic queries. It is interesting to notice that although manual run does not improve MAP, it significantly increases R.Rank, P@10, and P@20.

The main problem of manual queries is their inconsistent quality. There are some cases when manual query gives worse effectiveness relative to the respective automatic queries. Naturally there are some cases when manual queries do very well. The best two manual queries of our group, that gave best performance amongst all Terabyte runs this year, are for topics 816 and 849 that have MAP scores of 0.9032 and 0.5459 respectively. The original topics are "816. USAID assistance to Galapagos" and "849. Scalable Vector Graphics", while the manual versions are "816. USAID galapagos islands biodiversity activities" and "849. scalable vector graphics svg portability zoom".

Efficiency performance of all ad-hoc run is presented in Table 3. As can be seen from the table, the last three runs, that employed $I_{\text{DIFP}}$ indexes, are much slower than the first one, which used impact-sorted indexes.

## 6 Efficiency Performance

In addition to the ad-hoc runs, the following four runs were submitted for the efficiency task:

`SpeedBase` (`MU06TBy1`). This is the baseline run for the efficiency task. It employed basically the standard impact-based ranking process, with a slight modification intended to improve R.Rank. Unfortunately, it is not as good as `AdhocBase` in terms of P@10 and P@20.

`SpeedBase+Smoothing` (`MU06TBy2`). This run had the same initial setting as the `SpeedBase`, but with a smoothing process designed to reduce the relative gaps between lower contribution to the similarity score.

| RunID | Hardware | Efficiency | | |
|---|---|---|---|---|
| | | Index size (GB) | Index time (minutes) | Query time (secs) |
| `SpeedBase` | `Single` | 6.04 | 334 | 0.19 |
| `SpeedBase+Smooth` | `Single` | 6.04 | 334 | 0.23 |
| `SpeedBase+Prune` | `Single` | 6.04 | 334 | 0.08 |
| `SpeedBase+Prune+Multithread` | `Single` | 6.04 | 334 | 0.05 |

Table 4: *Efficiency performance of efficiency runs. All runs employed impact-sorted indexes. All results relate to the `GOV2` collection.*

| RunID | Topics 801–850 | | | | |
|---|---|---|---|---|---|
| | MAP | R-prec | R.Rank | P@10 | P@20 |
| `SpeedBase` | 0.0793 | 0.1124 | 0.7102 | 0.5440 | 0.4820 |
| `SpeedBase+Smooth` | 0.0838 | 0.1190 | 0.6998 | 0.5540 | 0.5050 |
| `SpeedBase+Prune` | 0.0821 | 0.1166 | 0.7059 | 0.5440 | 0.4890 |
| `SpeedBase+Prune+Multithread` | 0.0821 | 0.1166 | 0.7059 | 0.5440 | 0.4890 |

| RunID | Topics 751–800 | | | | |
|---|---|---|---|---|---|
| | MAP | R-prec | R.Rank | P@10 | P@20 |
| `SpeedBase` | 0.0633 | 0.0758 | 0.8169 | 0.6240 | 0.5520 |
| `SpeedBase+Smooth` | 0.0668 | 0.0805 | 0.8162 | 0.6180 | 0.5770 |
| `SpeedBase+Prune` | 0.0641 | 0.0772 | 0.8312 | 0.6220 | 0.5590 |
| `SpeedBase+Prune+Multithread` | 0.0641 | 0.0772 | 0.8312 | 0.6220 | 0.5590 |

Table 5: *Effectiveness performance of efficiency runs. All results relate to the `GOV2` collection.*

`SpeedBase+Prune` (`MU06TBy5`). This run had `SpeedBase` as a starting point, but used a dynamic index pruning technique, under which the low-impact blocks were excluded from the index during indexing time.

`SpeedBase+Prune+Multithread` (`MU06TBy6`). This run has the same setting as `SpeedBase+Prune`, but queries are processed with four threads as required by the task.

Efficiency performance is described in Table 4. Using a single commodity machine, we can index the `GOV2` collection in 8.2 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.

Table 4 shows the significant improve in query speed of the pruning strategy (with no loss on effectiveness, as shown in Table 5). However, it also shows that the multi-threading run is not up to the desired level. We expect that the situation can be improved with more programming effort.

Effectiveness performance is shown in Table 5. As mentioned earlier, the baseline run is not in line with that of the adhoc runs, and the pruned run is even slightly better than the full run. In addition, the smoothing process does help to improve P@20, but only slightly.

| RunID | Query time (secs) | Effectiveness | | | | |
|---|---|---|---|---|---|---|
| | | R.Rank | Found at 1 | Found at 5 | Found at 10 | Not Found |
| `NP,Impact` | 0.20 | 0.317 | 23.2% | 39.8% | 47.0% | 14.4% |
| `NP,BM25` | 0.27 | 0.397 | 28.7% | 52.5% | 62.4% | 13.8% |
| `NP,IVSM` | 0.26 | 0.329 | 23.2% | 43.6% | 50.8% | 14.9% |
| `NP,Impact+BM25` | 0.49 | 0.396 | 29.8% | 54.1% | 58.6% | 13.3% |

Table 6: *Performance of named-page runs. All results relate to the modified `GOV2` collection, where incoming text is added to destonated documents.*

## 7 Performance in the Named-Page Task

As described earlier, the document collection used for the named-page (NP) task is not the original one. In the new collection, all incoming anchor text is added to the destinated documents. Given that, the time for indexing can be greatly depends on the time to modify the collection. We were not able to measure this additional time, and the indexing time officially reported excludes the time for modifying the collection. This indexing time is not reported here.

Different index structures are employed, as described in the list of runs belows. All runs were performed in the `Single` hardware.

`NP,Impact` (`MU06TBn2`). This is the baseline run, using an $I_{\mathrm{DI}}$ index. The scoring mechanism is the standard impact-based one.

`NP,BM25` (`MU06TBn5`). This run employs an $I_{\mathrm{DF}}$ index and BM25 scoring mechanism. The motivation behind this run is that the original impact-based ranking might be badly affected by the modification of term frequencies using incoming impact. Hence, `NP,BM25` represents a second baseline.

`NP,IVSM` (`MU06TBn6`). This run employ an $I_{\mathrm{DI}}$ index, and a modified vector-space scoring mechanism over impact. That is, the impact score is now modified by document length, which in turn is calculated based on document impacts (ie. not based on within-document frequencies as usually).

`NP,Impact+BM25` (`MU06TBn9`). This run employs $I_{\mathrm{DIF}}$, and is operationally equivalent to merging the results of `NP,Impact` and `NP,BM25`. Let $s_1$ and $s_2$ be the scores of a document in two different ranking systems, the merged score $s$ is defined as:

$$s = 1 - (1 - \frac{s_1}{s_1^{\mathrm{max}}}) \times (1 - \frac{s_2}{s_2^{\mathrm{max}}}) \ ,$$

where $s_i^{\mathrm{max}}$ is the maximal score for the considered query, using retrieval scheme $i$.

Table 6 describes the performance of the named-page runs. The table shows that the original impacts are in-competitive for this approach of modifying document collection. The `NP,IVSM` is better than its baseline, but is still not as good as the `NP,BM25` baseloine. Finally, the merged approach seems to have a positive affect.

## 8 Conclusions and Future Directions

This year, we applied a number of changes to our system, mainly to allow different kinds of indexes and retrieval approaches. In general, the efficiency performance is slightly better than the previous year. The effectiveness level remains the same.

Note that we failed to perform the additional task, where the requirement is to employ a publicly available system to compare the efficiency. The reason is that the indexing using the public software was unable to operate on the hardware configuration that was used for the efficiency task.

**Acknowledgement** This work was supported by the Australian Research Council and the ARC Centre for Perceptive and Intelligent Machines in Complex Environments.

## References

V. Anh and A. Moffat. Improved word-aligned binary compression for text indexing. *IEEE Transactions on Knowledge and Data Engineering*, 18(6):857–861, 2006a.

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 2005. ACM Press, New York.

V. N. Anh and A. Moffat. Structured index organizations for high-throughput ext querying. In F. Crestani, P. Ferragina, and M. Sanderson, editors, *Proc. 13th Int. Symp. String Processing and Information Retrieval*, pages 304–315, Glasgow, Scotland, October 2006b. LNCS 4209, Springer.

S. Robertson, S. Walker, S. Jones, M. Hancock-Beaulieu, and M. Gatford. Okapi at TREC–3. In D. Harman, editor, *Proc. Third Text REtrieval Conference (TREC–3)*, pages 109–126, Gaithersburg, MD, November 1994. National Institute of Standards and Technology (Special Publication 500-225). URL `http://potomac.ncsl.nist.gov:80/TREC/t3_proceedings.html`.