Deriving Very Short Queries for High Precision and Recall (MultiText Experiments for TREC-7)

Gordon V. Cormack[1] Christopher R. Palmer[1] Michael Van Biesbrouck[1]
Charles L. A. Clarke[2]

MultiText Project
[1] Department of Computer Science, University of Waterloo, Waterloo, Ontario, Canada
[2] Department of Electrical and Computer Engineering, University of Toronto, Toronto, Ontario, Canada
`mt@plg.uwaterloo.ca`
`http://multitext.uwaterloo.ca`

Abstract

The main aim of the MultiText experiments for TREC-7 was to derive very short queries that would yield high precision and recall, using a hybrid of manual and automatic processes. Identical queries were formulated for adhoc and VLC runs. A query set derived automatically from the topic title words, with an average of 2.84 terms per query, achieved a reasonable but unexceptional average precision for the adhoc task and a median precision @20 for the 100 GB VLC task. However, these short queries achieved very fast retrieval times — less than 1 second per query over 100 GB using four inexpensive PCs. Two further query sets were derived using post-processing of the results of interactive searching on the adhoc corpus. Queries comprising a single conjunction, averaging 1.86 terms, achieved high precision on both adhoc and VLC tasks, and achieved faster retrieval times than the title-word queries. Compound queries averaging 6.42 terms achieved precision values competitive with the best runs, and retrieval times of 1.51 seconds per query on the 100 GB VLC corpus.

1 Introduction and Background

The MultiText search engine finds passages of text that exactly match a manually or automatically generated boolean query. Documents containing many short matching passages are assumed most likely to be relevant, and are ranked accordingly.

This approach has been found to be effective in several information retrieval contexts: “manual fixed query” (TREC-4 manual adhoc and routing [2]) in which a query is formulated from the topic statement and then run without modification; “interactive query” (TREC-5 adhoc [1]) in which a query is formulated and then refined interactively after viewing the top–ranked documents; “interactive routing” (TREC-5 routing and TREC-6 VLC [1, 5]) in which a query is formulated and then refined interactively after viewing results and judgements on another corpus; “interactive search and judging” (TREC-6 adhoc and high-precision [5]) in which simple queries are formulated, the top-ranked documents are viewed and judged, and the process is repeated – the documents judged relevant are recorded and submitted as the run; “cover density ranking” [3] in which a query is constructed automatically from a very small number of terms.

Each of these approaches offers advantages and drawbacks. Manual fixed queries are appropriate when the search system is slow or off-line, where interaction time is prohibitively expensive, or where the data does not yet exist (e.g. routing or filtering) and no suitable training data is available. Interactive query formulation is likely more appropriate in modern contexts where fast response is available. Yet formulating the "ideal" query is still a difficult task – it is possible to remove poor search terms and to add new ones discovered through interaction, but it quickly becomes difficult to determine whether a given refinement improves or degrades the query. Furthermore, it quickly becomes apparent that many of the same documents are retrieved again and again as the query is revised — to be effective the system must have some provision for previously viewed documents.

This provision for previously viewed documents leads immediately to interactive search and judging. The system cannot simply eliminate previously viewed documents, or the user will be unable to determine whether a refinement improves or degrades a query. Therefore, the user is asked to record a judgement for each viewed document, and these judgements (but not the documents) are reported in subsequent queries. Once judgements are recorded, it would appear that the best approach to adhoc retrieval would be to rank those documents judged relevant first, and to rank those documents judged not relevant last. It turns out that this assumption is not entirely accurate — documents judged not relevant still have a significant probability of being judged relevant in the official qrels — about 10% for our TREC-6 adhoc effort. For routing or filtering, the judgements cannot be used directly, as they apply to documents in the training corpus rather than the test corpus. But they can be used indirectly to test the efficacy of a given query.

All of the methods above require an amount of user interaction ranging from minutes to hours. It is not obvious which approach yields the best results for a given amount of user time: for fixed and interactive manual query construction a fair amount of time is spent composing sophisticated queries; for interactive search and judging, the queries tend to be simple and the majority of time is spent judging documents. It is our impression that interactive search and judging is more effective for any amount of user time, but we have yet to conduct experiments to confirm this impression.

The smallest possible amount of user time is afforded by the use of cover density ranking. A very small number of terms (not greater than 3) are used as a conjunctive query and ranked as described above. If this query yields an insufficient number of documents, a weaker query is constructed automatically. This weaker query consists of a subset of the terms. If there are still an insufficient number of documents, an even weaker query is chosen, and so on. The advantage with this approach is that it requires little user time (none, if you use the title field of a TREC topic). For short queries, cover density ranking yields performance that is comparable to other automatic techniques that do not use pseudo-relevance feedback.

2 TREC-7 Runs

Our primary goal was to construct simple boolean queries that we used for both the adhoc task and Very Large Collection track. Interactive search and judging was used to help develop the queries for the manual runs. The queries that were generated do not include the judgements which allows us to run these queries for the adhoc and VLC tasks. Our official runs were:

`uwmt7a0`, `uwmt7v0`:	Boolean queries consisting of exactly the title words. Plurals expanded to singular and plural forms. Weaker queries (using subsets of the terms) were derived automatically to be used as necessary for improving recall.
`uwmt7a1`, `uwmt7v2`:	3-term (or less) boolean queries automatically constructed using the information gathered during interactive search and judging on the adhoc corpus. Weaker queries (with a subset of the terms) derived automatically as necessary to improve recall.
`uwmt7a2`, `uwmt7v1`:	Larger (6.5 term average) boolean queries automatically constructed using the information gathered during interactive search and judging on the adhoc corpus. No weaker queries used to improve recall.
`uwmt7v3`:	Title words only; no expansion of plurals; no weaker queries. Not reported further in this paper.

In addition, we conducted a number of unofficial runs:

`uwmt7isj`:	A run consisting of all documents judged "relevant," followed by all documents judged "iffy" followed by all documents judged "not relevant."
`uwmt7a210`:	A run consisting of all documents in `uwmt7a2`, followed by all documents in `uwmt7a1`, followed by all documents in `uwmt7a0`, to a maximum of 1000 per topic.
`uwmt7isj210`:	A run consisting of all documents in `uwmt7isj`, followed by all documents in uwmt7a210, to a maximum of 1000 per topic.

Precision results for the six adhoc and four VLC runs are given in figures 1 through 4. Precision recall graphs are given in figure 5.

3 Title Only Automatic (`uwmt7a0, uwmt7v0`)

For TREC-6, the best automatic runs used only the title terms, expanded using pseudo-relevance feedback. We submitted such a run, achieving an average precision of 0.24. As an unofficial run, we tried cover density ranking with no expansion and achieved an average precision of 0.20. Since our interest this year was in pure boolean queries, we used an enhanced cover density ranking procedure for TREC-7.

We took the title words and expanded all plural words into their plural and singular forms. This expansion used naive suffix matching as shown in figure 6

This was the only form of expansion or stemming performed. The rationale is that we observed from previous TREC efforts that the topics used a collective phrasing when documents describing individual instances were relevant; for example "automobile accidents."

We did not use other forms of stemming because we found previously that stemming compromised early precision, and did not necessarily improve average precision. We did not wish to compromise early precision, especially for our VLC runs.

We constructed weakened queries by taking all subsets of the title terms and ordering them by their inverse frequency of occurrence in the database. For example, topic 366 (commercial cyanide uses) was expanded to:

```
commercial AND cyanide AND (use OR uses)
commercial AND cyanide
cyanide AND (use OR uses)
cyanide
commercial AND (use OR uses)
use OR uses
```

Recall that the second and subsequent queries are used only if the previous query or queries yield insufficient results. Weakened queries were further restricted to be contained in some interval of 128 words. The average number of terms per (un-weakened) query was 2.84, where depluralized words count as 2 terms (e.g. there are 4 terms in the query for topic 366, above).

This technique was applied to the TREC-6 adhoc task yielding an average precision of 0.22; we were hoping to achieve similar results on TREC-7. We have no explanation at this time as to why the average precision was significantly worse (0.19). As expected, on the VLC runs precision @20 improved with collection size, from 0.190 on 1 GB to 0.442 on 100 GB.

4 Interactive Search and Judging

The remaining runs were based on interactive search and judging. For TREC-6, four researchers spent an average of two hours per topic searching and judging documents. Our analysis of this effort lead us to conclude that we could achieve good results with less time [6]. Therefore, we spent less than 30 minutes per topic searching and judging. We tried to find a reasonable number of documents for each topic, but were not exhaustive.

As for TREC-6, documents were judged to be "relevant", "not relevant", or "iffy". In total, we judged 5529 documents: 2561 relevant, 629 iffy, 2339 not relevant. 2237 of the documents we judged relevant were judged also by NIST; 1543 of these (about 2/3) were judged relevant by NIST. This level of agreement is nearly identical to last year. However, NIST judged 2836 documents relevant that we did not judge, and judged relevant a further 133 documents that we judged not relevant. These sets are shown as a Venn diagram in figure 7.

We did not submit ISJ to NIST as an official run. We did, however, evaluate the documents we judged using the official judgements — this run is reported here as `uwmt7isj`. The average precision of 0.3458 is much lower than that achieved by ISJ last year. No doubt this is due to the poorer recall reported above — last year we judged about 2/3 of those documents the NIST found relevant. We were a bit surprised by the low recall of ISJ this year. While we spent considerably less time at it (by more than a factor of 4), we did not feel that we were overlooking large numbers of relevant documents. Apparently our feelings were inaccurate.

We used our ISJ judgements to evaluate `uwmt7a0` (see figure 8). Early precision was significantly better with NIST judgements while average precision and R-precision was significantly better with ISJ judgements. `uwmt7a1` and `uwmt7a2` show much better precision figures, which is no surprise because they were constructed to optimize their performance on these judgements (see next section). `uwmt7isj` and `uwmt7isj210` exhibit perfect scores with respect to this set of judgements because they place all the relevant documents first.

Run	P@1	P@2	P@4	P@20	P@40	RP	AP
`uwmt7a0`	.5600	.4700	.4700	.3560	.2800	.2289	.1866
`uwmt7a1`	.7200	.7700	.7000	.5330	.4205	.3402	.2983
`uwmt7a2`	.7800	.7900	.7300	.5610	.4730	.4012	.3587
`uwmt7isj`	.8000	.7800	.7400	.6060	.5115	.3952	.3458
`uwmt7210`	.8200	.7900	.7150	.5620	.4740	.4092	.3868
`uwmt7isj210`	.8000	.7800	.7400	.6060	.5135	.4384	.4112

Figure 1: Adhoc Runs

Run	P@1	P@2	P@4	P@10	P@20
`uwmt7v0`	.5000	.4800	.4900	.4680	.4420
`uwmt7v2`	.6200	.6100	.5950	.5860	.5740
`uwmt7v1`	.6600	.6700	.6600	.6380	.5980

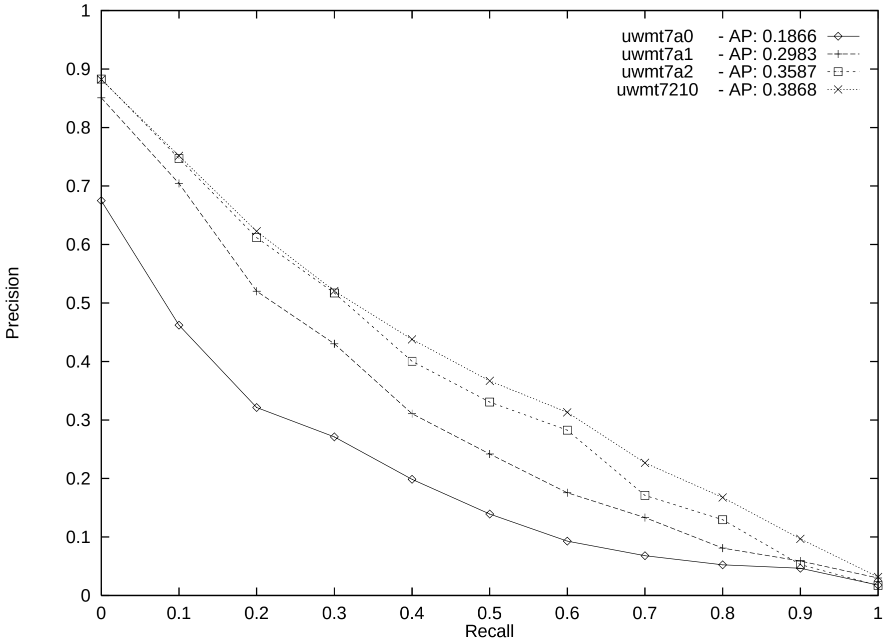
Figure 2: VLC Runs (100 GB Corpus)

Run	P@1	P@2	P@4	P@10	P@20
`uwmt7v0b10`	.5200	.4500	.4500	.4160	.3690
`uwmt7v2b10`	.5800	.5500	.5150	.4660	.4110
`uwmt7v1b10`	.6000	.5900	.5800	.5180	.4740

Figure 3: VLC Runs (10 GB Sample)

Run	P@1	P@2	P@4	P@10	P@20
`uwmt7v0b1`	.4000	.3200	.3100	.2420	.1900
`uwmt7v2b1`	.4400	.4000	.3850	.2820	.2230
`uwmt7v1b1`	.5200	.4500	.4000	.3140	.2350

Figure 4: VLC Runs (1 GB Sample)



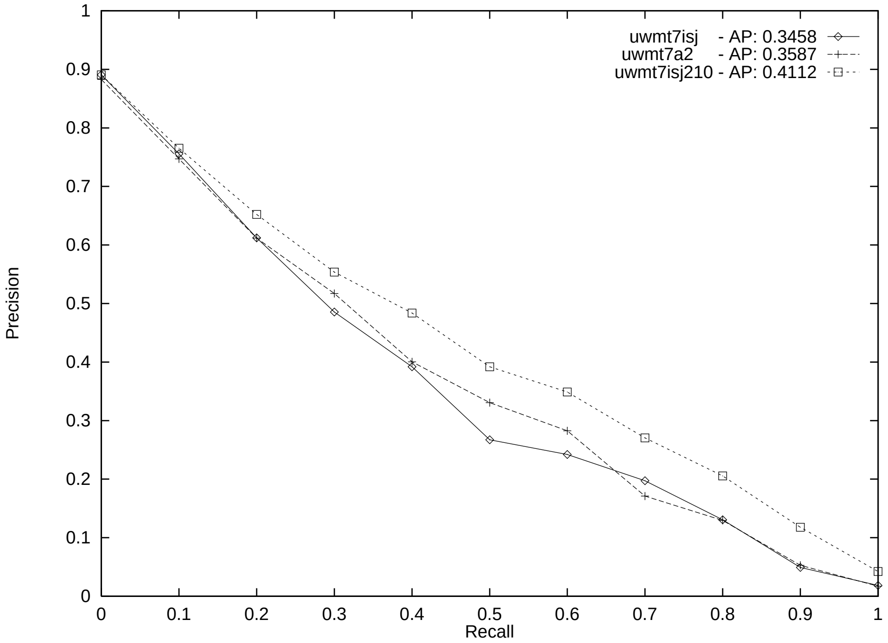


Figure 5: Precision-Recall for Adhoc Runs

5 Automatic Query Extraction

Rather than submit the results of ISJ as an official run, we used an automatic process to extract query terms from the ISJ logs and to build queries that optimized AP on the ISJ judgements.

As mentioned in the introduction, the process of formulating an ideal query can be quite difficult and time consuming. Yet such a query may be desirable for a number of reasons. First, if ISJ is performed on a subset of the corpus, or on a separate training corpus, a query is needed to gather additional documents not subject to ISJ. Second, if ISJ is incomplete, that is, if not all potentially relevant documents are judged, a query may include relevant documents that would otherwise be missed. Third, because of poor judging agreement between searchers and the official judgements, it may be desirable to include documents judged "not relevant" by the searchers.

We had a second purpose in extracting queries from ISJ. We wished to test the hypothesis that queries with a very small number of terms can yield good retrieval performance. It was seen in TREC-6 that the topic titles contained enough information to give the best performance of all automatic approaches. The VLC queries we constructed by hand for TREC-6 were very short and yielded much better performance. We wished to explore the possibility of building still better queries using an automatic optimization procedure.

To this end, we extracted all the queries used in ISJ from our logs. We further reduced each query to sum-of-products form and considered each product separately. We then considered all subsets of each product. For example, the query

```
(amazon OR rain) and forest and (brazil or colombia)
```

would yield:

```
amazon AND forest AND brazil
amazon AND forest AND colombia
rain AND forest AND brazil
rain AND forest AND colombia
amazon AND forest
amazon AND brazil
amazon AND colombia
rain AND forest
rain AND brazil
rain AND colombia
forest AND brazil
forest AND colombia
amazon
rain
forest
brazil
colombia
```

All such queries were evaluated using the ISJ judgements. The run `uwmt7a1` is the single query for each topic that yielded the highest average precision. No query exceeded 3 terms, and the average number of terms was 1.86.

pattern	expansion
*ss	*s, *ss
*sses	*s, *sses
*xes	*x, *xes
*zes	*z, *zes
*ches	*ch, *ches
*shes	*sh, *shes
*eys	*ey, *eys
*ies	*y, *ies
*s	*, *s

Figure 6: Suffix Pattern Matching Rules

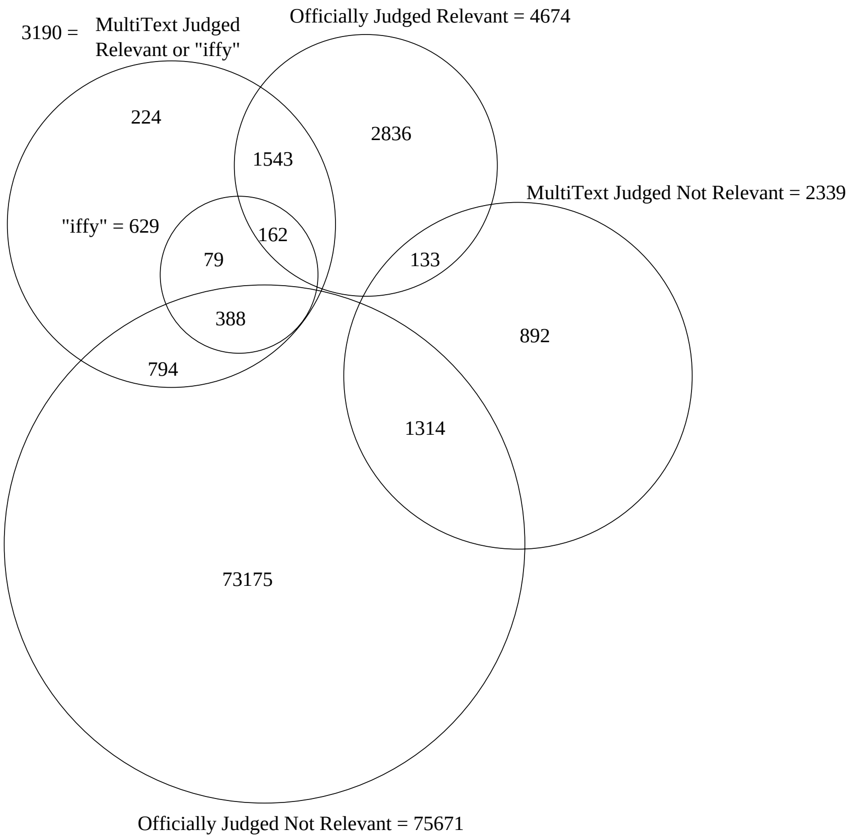

3190 = MultiText Judged Relevant or "iffy"
Officially Judged Relevant = 4674
224
2836
1543
MultiText Judged Not Relevant = 2339
"iffy" = 629
162
79
133
388
892
794
1314
73175
Officially Judged Not Relevant = 75671

Figure 7: Judging Coverage and Agreement

Run	P@1	P@2	P@4	P@20	P@40	RP	AP
`uwmt7a0`	.4800	.4500	.4350	.3110	.2280	.2603	.2245
`uwmt7a1`	.9600	.9100	.8650	.6270	.4635	.5348	.5206
`uwmt7a2`	.9800	.9600	.8900	.6970	.5480	.6760	.7006
`uwmt7isj`	1.000	1.000	.9900	.8750	.7097	1.000	1.000
`uwmt7a210`	.9600	.9600	.8900	.6970	.5490	.6746	.7043
`uwmt7isj210`	1.000	1.000	.9900	.8750	.7097	1.000	1.000

Figure 8: Adhoc Runs (ISJ Judgements)

For `uwmt7a1`, we considered combinations of up to five of these elementary queries, choosing the one yielding the best average precision with respect to the ISJ judgements. Then we considered each of the queries in the query log. If one of the original queries (as entered by the searcher) yielded a higher average precision, it replaced the automatically generated query. Such was the case for only 12 topics. The average number of terms per query (counting repeated words only once) was 6.42 terms.

As seen in figures 1 and 5, `uwmt7a1` significantly outperforms `uwmt7a0` (title only) although the queries are equally simple. `uwmt7a2` performs better still. These figures also show that `uwmt7a2` and `uwmt7isj` have nearly identical performance, with `uwmt7isj` having slightly higher early precision and `uwmt7a2` having slightly higher average precision.

6 Combining Results

Some of the runs, most notably `uwmt7a2` and `uwmt7isj`, produced less than 1000 documents for some topics, compromising recall and therefore average precision. To `uwmt7a2`, we appended the results of `uwmt7a1` and `uwmt7a0` (up to 1000 documents per topic) to form a new run, `uwmt7a210`. The performance of `uwmt7a210` is, of course, better than `uwmt7a2`, as shown in figures 1 and 5: average precision, for example, rises from 0.3587 to 0.3868. ISJ (`uwmt7isj`) can similarly by extended by these other runs, yielding `uwmt7isj210`. Figures 1 and 5 illustrate that this combination yields the best overall result: average precision improves from 0.3458 to 0.4112.

These combinations show two ways in which extracting queries from ISJ logs can be an advantage: the queries themselves yield good performance, especially when combined; the queries can be used to extend the set of documents found by ISJ. We would expect the latter approach to be particularly effective when the time for interactive search and judging is severely restricted.

7 Very Large Corpus

We used the same queries for the 2 GB adhoc, 1 GB VLC, 10 GB VLC, and 100 GB VLC collections. `uwmt7v0` used the same queries as `uwmt7a0`, fulfilling the requirement for an automatic run based on only title and description fields. `uwmt7v1` and `uwmt7v2` used the same queries as `uwmt7a2` and `uwmt7a1` respectively, and may therefore be regarded as routing tasks.

It was observed at TREC-6 that P@20 improves with collection size. This should be no surprise. Indeed, we would expect that P@(k/c) would be constant for a given query, where k is any constant and c is the size of the collection. We modified the trec_eval program to report relevant P@(k/c) values and observed that this value was roughly constant for our TREC-6 VLC queries, as measured between the 2 GB sample and the 20 GB VLC corpus, and also as measured between the 2 GB adhoc corpus and the 20 GB VLC corpus.

This observation led us to conclude that, when building queries based on ISJ with the 2 GB adhoc corpus, optimizing P@0.4 would yield the best result with respect to the 100 GB VLC2 corpus. On the other hand, optimizing P@4 would be best with respect to the 10 GB VLC corpus, and P@40 would be best with respect to the 1 GB VLC corpus. If we had had sufficient time, we would have constructed such a run in addition to the ones we submitted. Nevertheless, our runs, yielded P@1 (P@0.4 is of course meaningless) of 0.48, 0.96 and 0.98 on our ISJ judgements. Due to judging disagreement, we predicted that the last two might achieve about 2/3 of these values, or 0.64 and 0.65. In fact, our runs achieved P@20 of 0.4420, 0.5740, and 0.5980; about 10 percent less than predicted. As an aside, we note that the P@1 values on NIST judgements (which were, of course, unknown when we prepared these runs) turned out to be 0.5600, 0.7200, and 0.7800.

Our predictions for the smaller sizes were much less accurate. P@40 on the 2 GB adhoc collection should predict P@20 on the 1 GB VLC. P@4 should predict P@20 on the 10 GB VLC. They do not. From this we conclude that the relationship between k and P@(k/c) is radically different between the adhoc and VLC collections. We conjecture that this might be due to a much smaller fraction of relevant documents in VLC. A more appropriate formula might be P@(k/R) where R is the number of relevant documents in the collection. Unfortunately, unlike c, R is very difficult to anticipate for an unknown collection.

The relationship between k and P@(k/c) is invariant among the various VLC collections. For example consider `uwmt7a0`: P@2 (1 GB) and P@20 (10 GB) are 0.3200 and 0.3200; P@2 (10 GB) and P@20 (100 GB) are 0.4444 and 0.4420. The insensitivity of P@(k/c) to c can be seen visually in figure 9.

8 Architecture and Efficiency

For VLC, we used four commodity Intel Pentium II 300 MHz personal computers, connected by a 10 MB/s ethernet. Each computer ran two copies of the search engine, so as to afford CPU/IO overlap during processing. A dispatcher sent each query in turn to all 8 search engines, and waited to receive 20 results from each. Then the best 20 of these results were returned as the result of the query.

The central data structure for each engine is an inverted index, structured so that the entire occurrence list for a word appears contiguously on disk. Furthermore, the index is structured so that a random access into the occurrence list can be performed with a single disk operation [4]. This allows intervals containing a set of terms to be calculated with effort roughly proportional to the frequency of the least common term. We further reduced the effort by restricting the interval size to 128 words or less — this restriction allowed fruitless partial results to be discarded. To further improve performance, the queries were normalized and redundancies were eliminated. Execution times for the VLC runs (seconds/query) are given in figure 10.

The index structures were built in two passes. In the first pass, blocks of approximately 100MB were scanned and indexed, and the index written to disk. In the the second pass, these index blocks

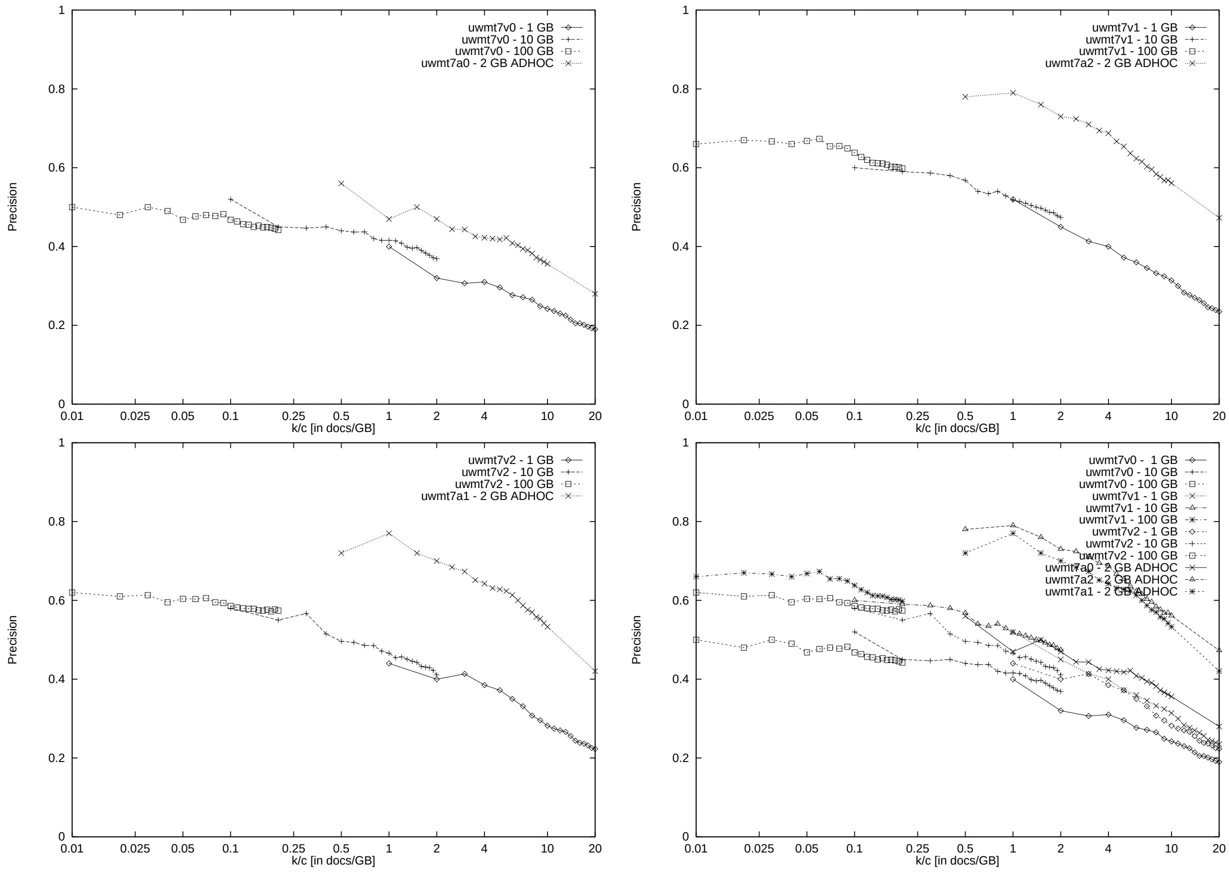


Figure 9: P@(k/c) for each VLC and Adhoc Run

Run	1 GB	10 GB	100 GB
uwmt7v0	.306	.294	.708
uwmt7v2	.251	.299	.882
uwmt7v1	.216	.377	1.51

Figure 10: VLC Query Execution Times

were merged to form a single index. This process was done sequentially for the two engines per computer, and in parallel across the four computers. Total build time for the 1 GB, 10 GB, and 100 GB systems was 0.052 hours (187 seconds), 0.504 hours (30.2 minutes), and 5.33 hours.

References

[1] Charles L. A. Clarke and Gordon V. Cormack. Interactive substring retriveal. In *Fifth Text REtrieval Conference (TREC-5)*, 1996.

[2] Charles L. A. Clarke, Gordon V. Cormack, and Forbes J. Burkowski. Shortest substring ranking. In *Fourth Text REtrieval Conference (TREC-4)*, pages 295–304, Gaithersburg, Maryland, November 1995.

[3] Charles L. A. Clarke, Gordon V. Cormack, and Elizabeth A. Tudhope. Relevance ranking for one to three term queries. In *Fifth RIAO Conference*, pages 388–400, Montreal, June 1997.

[4] Clarlie L. A. Clarke and G. V. Cormack. Dynamic inverted indexes for a distributed full-text retrieval system. Technical Report MT-95-01, MultiText Project, University of Waterloo, 1994. Available at `http://plg.uwaterloo.ca:80/~ftp/mt/TechReports/MT-95-01/`.

[5] Gordon V. Cormack, Charles L. A. Clarke, Christopher R. Palmer, and Samuel S. L. To. Passage-based information retrieval in TREC-6. In *Sixth Text REtrieval Conference (TREC-6)*, 1997.

[6] Gordon V. Cormack, Christopher R. Palmer, and Charles L. A. Clarke. Efficient construction of large test collections. In *Proceedings of SIGIR 98, the ACM Annual Conference on Research and Development in Information Retrieval*, pages 282–289, Melbourne, Australia, August 1998.