Passage-Based Refinement
(MultiText Experiments for TREC-6)

Gordon V. Cormack[1] Charles L. A. Clarke[2] Christopher R. Palmer[1] Samuel S. L. To[1]

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 MultiText system retrieves passages, rather than entire documents, that are likely to be relevant to a particular topic. For all runs, we used the reciprocal of the length of each passage as an estimate of its likely relevance and ranked accordingly. For the manual adhoc task we explored the limits of user interaction by judging some 13,000 documents based on retrieved passages. For the automatic adhoc task we used retrieved passages as a feedback source for new query terms. For the routing task we estimated probability of relevance from passage length and used this estimate to construct a compound (tiered) query which was used to rank the new data using passage length. For the Chinese track we indexed individual characters rather than segmented words or bigrams and used manually constructed queries and passage-length ranking. For the high precision track we performed judgements on passages using an interface similar to that used for the manual adhoc task. The Very Large Collection run was done on a network of four cheap computers using very simple manually constructed queries and passage-length ranking.

1 Introduction

The MultiText Project participated in the routing and adhoc tasks, and in the Chinese, high precision and very large collection tracks. For the adhoc task we submitted both automatic and manual runs; for the routing task and the tracks we submitted manual runs. These experiments explored a variety of methods for query expansion and refinement based on arbitrary passage retrieval.

The major research focus of the the MultiText Project is the development and prototyping of scalable technologies for distributed information retrieval systems. The MultiText system is based on the federated architecture shown in Figure 1. The system is composed of several major components: The *index engines* maintain the index file structures and provide search capabilities. The *text servers* are specialized by document type and provide retrieval capabilities for arbitrary text passages specified at the word level. Finally, the *marshaller/dispatcher* interacts with clients and coordinates query and update activities.

Research issues are addressed in the context of this distributed architecture. Issues of concern to the MultiText Project include data distribution, load balancing, fast update, compression, fault tolerance, document structure, ranking, and user interaction. Support for document structure is a

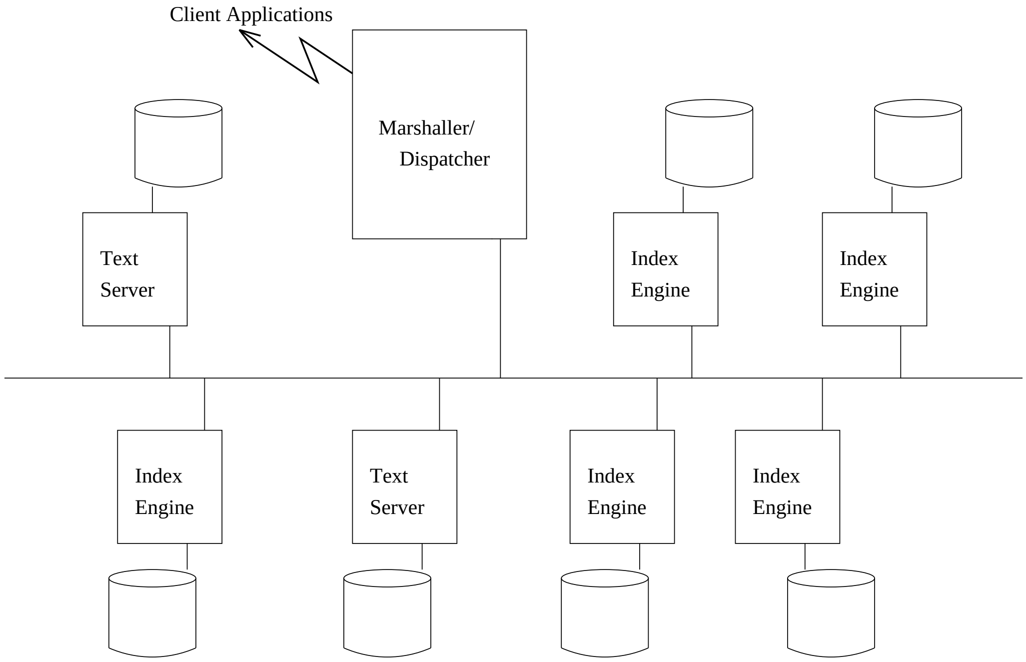

Figure 1: Architecture of the MultiText System

particular feature of the MultiText System. The system can support multiple document formats within a single integrated database and provide specific support for structure inherent in each document type. The MultiText query language, GCL, provides facilities for directly referencing document structure and allows queries to reference equivalent structural elements across differently formatted documents [2]. Ranking in the MultiText system is based on passage retrieval, with the score of a passage based on its length, and the score of a document based on the score of the passages contained within it. As well as full documents, the method allows ranking of arbitrary document components. Scores do not depend on collection-wide statistics, making the ranking method particularly suitable for use in a distributed environment.

Our TREC-4 paper introduced the basic ranking technique, *shortest substring ranking* [3]. Our TREC-5 paper extended this work and introduced a method of passage-based interactive query expansion and refinement [1]. For TREC-6 we have extended this interactive query refinement method for participation in a wide range of tracks and tasks. In addition, for the first time, we submitted a fully automatic run that extends and validates methods introduced since TREC-5 [4].

The next section summarizes the GCL query language and shortest substring ranking. The subsequent section provides an overview of our TREC-6 work. Section 4 describes our work for each of the tasks and tracks in detail.

2 Shortest Substring Ranking

We model the text in a database as a sequence of terms. Document structure is indicated by structural markers indexed at positions between the terms. A solution to a query is a set of intervals from this text sequence. Each interval is represented by an *extent*, an ordered pair (p, q), with $p \leq q$, representing a start and end position in the text sequence.

The simplest GCL query is a phrase. The solution to the query

```
"Hubble Telescope"
```

is the set of extents corresponding to the locations of this phrase within the text sequence.

GCL supports the standard boolean operators. Each interval in the solution set for the query

```
"Africa*" and "civilian" and ("death*" or "casualties")
```

satisfies the conditions implied by the boolean operations. In order to limit its size, we restrict the solution set to include only those intervals that do not contain smaller intervals that satisfy the query conditions. This *shortest substring rule* provides linearity and ordering properties that make efficient query evaluation possible and is central to the document scoring technique. Ranking is based on the length and density of these solution intervals.

An ordering operator (“`...`”) is provided to link the start and end positions of text intervals defined by sub-queries. The query

```
"<title>"..."</title>"
```

links the start and end tags for titles, and has as its result the set of all titles. Similarly, the query

```
"<doc>"..."</doc>"
```

has as its result the set of all documents. The shortest substring rule guarantees that the solution set contains only single documents. Start and end tags that are more than a single document apart are not linked. The language includes four *containment operators*— `containing`, `not containing`, `contained in`, and `not contained in`— which may be used to query structural relationships. The query

```
("fiber"..."globe") contained in ("<doc>"..."</doc>")
```

finds occurrences of “fiber” followed by “globe” within the same document. Note that elements in the solution set for this query are text intervals, not documents. In our queries for TREC-6 we made frequent use of the “`contained in`” operator for passage length restriction. A major element in our work for the routing task was a method for determining appropriate values for passage length restrictions. The query

```
("fiber"..."globe") contained in 6 words
```

finds occurrences of “fiber” followed by “globe” that are six words or less in length.

For ranking purposes, an extent (p, q) is assigned a score using the formula

$$I(p,q) = \begin{cases} \frac{\mathcal{K}}{|q-p+1|} & \text{if } |q-p+1| \geq \mathcal{K} \\ 1 & \text{otherwise.} \end{cases}$$

Intervals whose length is less than the “cutoff” parameter $\mathcal{K}$ are assigned a score of 1; larger intervals are scored in proportion to the inverse of their length. Our TREC-6 experiments used values of $\mathcal{K}$ between 1 and 16.

Document scores are computed by combining the scores for the solution intervals contained within them. Given a document D and a query Q, and assuming the document contains solutions

(p_1, q_1), (p_2, q_2), ..., (p_n, q_n) ordered such that $I(p_i, q_i) \geq I(p_j, q_j)$ if $i < j$, the document's score is computed using the formula

$$S(D, Q) = \sum_{i=1}^{n} \gamma^{i-1} I(p_i, q_i)$$

Where $0 < \gamma \leq 1$ is a geometric "decay" parameter. Values for γ of 1 and 0.5 were used in our TREC-6 experiments.

This scoring method may be generalized to apply to any set of text intervals, not just documents. Using this method, the solution set for an arbitrary GCL query may be ranked in terms of a second GCL query without any need for special indexing. Since there are no collection statistics and every document is independently ranked the method is particularly suited to the distributed architecture of the MultiText System.

A GCL query produces a fixed solution set, which is then ranked using the formula given above. If a larger set of ranked documents is required —as is the case of many of the TREC tasks and tracks where up to a thousand ranked documents may be submitted — secondary queries may be used to provide these extra documents. A GCL query may be augmented by an ordered set of additional query "tiers", which are used, in order, when the initial query is exhausted. For example, a tiered query for topic 10004 ("Iranian Arms to Bosnia") might be:

```
("iran*" or "tehran") and "bosnia*" and ("arms" or "weapons")
("bosnia*") and ("arms" or "weapons")
("iran*" or "tehran") and ("arms" or "weapons")
```

The query in the first tier is intended to be a precise expression of the requirements underlying the topic. Queries in later tiers are "weaker" and are intended to pick up a large number of possibly relevant documents. The use of user-constructed tiered queries proved to be of limited value in our previous TREC work [1]. However, we have developed methods for constructing tiers based on relevance information. As a consequence, we have retained the method for further exploration, particularly as an element of our work for the routing task.

3 Passage-Based Refinement

Our TREC-6 efforts represent a diversity of exploratory and comparative work. These efforts are unified by a common theme of passage-based refinement. The MultiText System is distinguished by its support for arbitrary passage retrieval: the retrieval of passages defined at query time rather than at build time.

Our TREC-5 work introduced an interactive method for query expansion and refinement based on the selection of terms from relevant passages. This method avoids any requirement for users to examine full documents by focusing attention on the portions of the documents that are likely to be of greatest interest.

This interaction method was used in most of our TREC-6 work. Our Chinese track results represent a direct application of the TREC-5 methodology to a Chinese language environment. For the manual adhoc task and the high precision track, the method was used to make fast relevance judgements.

For the automatic adhoc task we used automatic feedback to select terms from high-scoring passages as a method of query expansion.

For the routing task we developed a method of query re-structuring that depends on passage length restrictions.

4 MultiText Experiments for TREC-6

4.1 Routing

We submitted two routing runs. The baseline run (`uwmt6r0`) used queries created using the manual technique from TREC-5: a tiered boolean query was created from combinations of terms that were found to select relevant documents. An initial query was created manually from the topic statement and was refined interactively by selecting words and phrases from relevant passages. For the comparison run (`uwmt6r1`) we decomposed these queries into elementary combinations and re-tiered them to place the best combination of terms first. The queries were further refined using length restriction and further expanded using additional interaction. The methods used to create the comparison run from the baseline run were partially automatic and partially manual. Our intention is to make the process fully automatic in the future.

Our method for re-tiering uses relevance information to place a query more likely to select a relevant document in an earlier tier. For a given query Q and document D, the probability $p(D)$ that the document is relevant is assumed to be a monotone function of the score of D with respect to Q; that is, $p(D) = F_Q(S(D, Q))$. If F_Q were known, it would be possible to construct an optimal tiered query as follows:

available-queries $\leftarrow \{(Q_i, \text{BEST-SCORE})\}$
repeat until $|$*available-queries*$| = 1$
  find (Q_1, S_1) in *available-queries* such that $F_{Q_1}(S1)$ is maximized
  remove (Q_1, S_1) from *available-queries*
  find (Q_2, S_2) in *available-queries* such that $F_{Q_2}(S_2)$ is maximized
  find S_1' such that $F_{Q_1}(S_1') = F_{Q_2}(S_2)$
  add (Q_1, S_1') to *available-queries*
  next tier of query is Q_1, restricted to scores better than S_1'
end loop
only one element, (Q_i, S_i) remains in *available queries*
final tier is Q_i

We used the training data to estimate F_Q for elementary queries consisting of a simple conjunction of terms. To estimate F_Q we ranked the documents in the training set by Q and plotted the number of relevant documents retrieved as a function of the total number of documents retrieved; we use the slope of this curve as an estimate of the probability of relevance at any point. We labeled each curve with the reciprocal of exponentially decreasing score values; from the slope and these labels we were able to estimate F_Q.

Consider the example in figure 2. The initial slope of the curve labeled "q5" is approximately 0.85; at 100 documents retrieved the slope diminishes to approximately 0.5. On the other hand, the curve labeled "q3" has an initial slope of .54; at 175 documents this slope diminishes to 0.42. The slope of "q1" diminishes much more abruptly: up to 200 documents it is about the same as "q3" (0.54) but beyond this point it rapidly approaches zero. Both "q0" and "q2" have initial slopes of 0.3, which diminishes after 300 documents. Applying the re-tiering algorithm manually to this query set, we conclude that the best tiered query would be:

q5a (q5 up to the score of the 100th document)
q3a (up to the score of the 175th document)
q1a (up to the score of the 200th document)
q5b (beyond the score of the 100th document)

q3b (beyond the score of the 175th document)
q2a (up to the score of the 300th document)
q0a (q0 up to the score of the 300th document)
q4
q2b (q2 beyond the score of the 300th document)
q1b (q1 beyond the score of the 300th document)
q0b (q0 beyond the score of the 300th document)

Because of the uncertainty in the estimates, and in order to reduce the number of tiers, we combine tiers with nearly equal slopes:

q5a
q3a+q1a+q5b
q3b
q2a+q0a
q4
q2b+q1b+q0b

At present, GCL does not directly permit the score restriction described above. However, with a decay parameter of $\gamma = 0.5$, the score of document is bounded to twice the score of the best passage in the document. Furthermore, the score of a passage is based directly on its length. Therefore, the score restriction may be approximated using a length restriction. This approach was taken for our TREC-6 routing queries.

Figure 3 shows that this refinement process provided an improvement over our baseline. We found the the insight gained from our analysis of the routing task to be useful in the manual ad hoc task, as well as in the high precision and very large collection tracks. We aim to automate the refinement process in future.

4.2 Automatic Adhoc

For the automatic adhoc task our primary interest is in the title-only run, where the title of each topic is treated as the query. These queries are very short: the longest consists of four terms; the average length is less than three terms; three consist of a single term. Previous MultiText research [4] introduced a new method for ranking very short queries. The method, called *cover density ranking*, is related to the ranking method described in Section 2. Cover density ranking forms the basis for our TREC-6 automatic adhoc experiments.

Similar to the tiered ranking of Section 2, cover density ranking is a two step process. Documents are ranked first by *coordination level*, the number of distinct query terms appearing in the document. For example, given a query consisting of three terms, a document containing at least one occurrence of each term is ranked ahead of a document containing only two of the terms, which in turn is ranked ahead of document containing only one of the terms. A secondary ranking procedure is then applied to the documents within each coordination level to produce the final document order. For this secondary ranking, the score of a document is determined by the applying shortest substring ranking formula of Section 2 to the logical conjunction of the query terms appearing in the document.

Cover density ranking may be expressed in terms of the procedure of Section 2. The GCL query language supports a "generalized combination" operator with syntax

$$N \texttt{ of } (Q_1, Q_2, \ldots, Q_M)$$

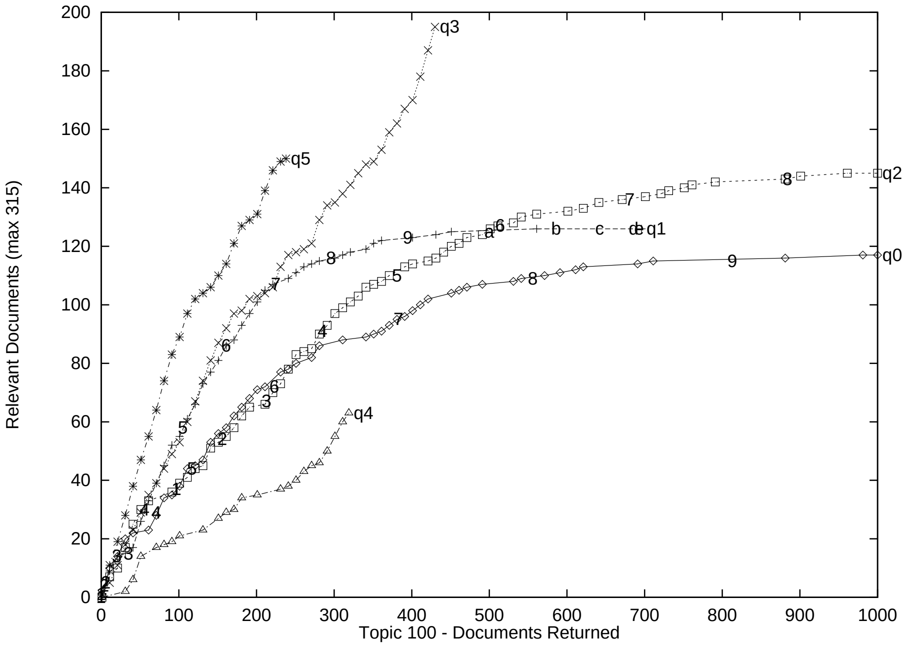

Figure 2: Slope Estimation

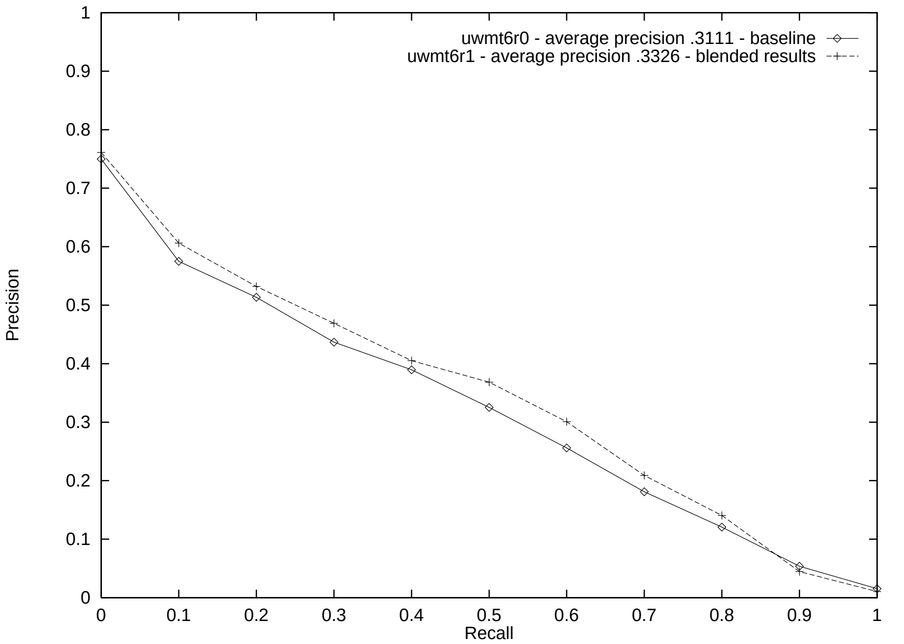

Figure 3: Routing Runs

where $Q_1 ... Q_M$ are GCL subqueries and $N \leq M$. The operator finds the set of text intervals that contain a solution to exactly N of the M subqueries under the shortest substring rule. Given query terms $T_1 ... T_M$, cover density ranking is implemented using the tiered query set

$$
\begin{array}{rl}
M & \texttt{of} \; (T_1, \; T_2, \; ..., \; T_M) \\
M-1 & \texttt{of} \; (T_1, \; T_2, \; ..., \; T_M) \\
M-2 & \texttt{of} \; (T_1, \; T_2, \; ..., \; T_M) \\
 & \cdots \\
1 & \texttt{of} \; (T_1, \; T_2, \; ..., \; T_M)
\end{array}
$$

For very short queries, a user preference for coordination level ranking has been observed by several groups [5, 8, 9]. For these queries, cover density ranking can provide retrieval effectiveness comparable to that of more traditional ranking methods [4]. Cover density ranking directly satisfies the user preference while maintaining good retrieval effectiveness. In addition, cover density ranking is fast and requires only a simple word-position index for implementation.

Our TREC-6 experiments for the automatic adhoc task had two goals: The first goal was to gain additional experience with the cover density ranking method in a comparative setting. In addition to the comparisons made possible by TREC experimental environment, we intended to directly compare cover density ranking with an established ranking method. The second goal was to provide a preliminary evaluation of local feedback based on the passages identified during the cover density ranking procedure. These passages are often short and were expected to be a valuable source of terms for query expansion.

For feedback purposes, passages were extracted using the generalized combination operator

$$N \; \texttt{of} \; (T_1, \; T_2, \; ..., \; T_M)$$

using the largest value of N for which solutions existed. An upper bound of 64 words was set as a maximum passage length for feedback; passages longer than this 64 word bound were eliminated from consideration. Passages shorter than 64 words were expanded symmetrically to 64 words, with the original length retained for scoring. Words appearing in these passages were scored on the basis of their frequency of occurrence in the passages, their frequency of occurrence in the database, and the (original) length of the passages. The 24 highest scoring words were added to the query. For example, the original (title-only) query for topic 301

```
crime international organized
```

expanded into the query

```
activity combating cooperation crime crimes criminal drug enforcement fight
intelligence internal international main ministry narcotics nations
organization organizations organized organs republic sec smuggling struggle
terrorism
```

One of the goals of our TREC-6 automatic adhoc experiments was a direct comparison of cover density ranking with an established ranking method. In order to permit this direct comparison we implemented a version of the Okapi measure [6, 7] as part of the MultiText System. For a document D and a query Q we compute

$$\sum_{t \in D \wedge t \in Q} \log \left(\frac{N - n_t + 0.5}{n_t + 0.5} \right) \left(\frac{f_{D,t}}{f_{D,t} + l_D / l_{avg}} \right)$$

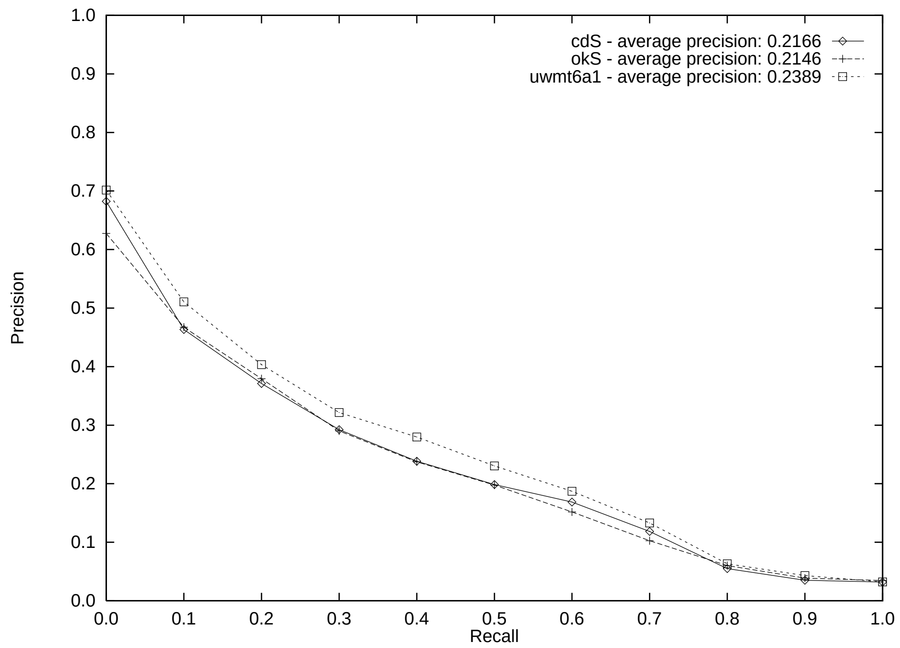

Figure 4: Automatic Adhoc Title-Only Runs

where:

$$
\begin{aligned}
N &= \text{number of documents in the collection;} \\
n_t &= \text{number of documents containing term } t\text{;} \\
f_{D,t} &= \text{frequency of ocurrence of term } t \text{ in document } D\text{;} \\
l_D &= \text{length of document } D\text{;} \\
l_{avg} &= \text{average document length.}
\end{aligned}
$$

In experiments reported by the RMIT group at TREC-4, a similar version of this measure proved to be particularly appropriate for short queries [9]. In addition, since cover density ranking is not suitable for use with the expanded queries, the implementation of this measure permitted us to combine the results of feedback with the results of the original cover density ranking.

Figure 4 presents the results of our experiments with the title-only queries. The figure plots recall-precision curves and provides average precision values for three runs. The run labeled `cdS` used cover density ranking. The run labeled `okS` used the MultiText implementation of the Okapi measure. For both of these runs the terms were stemmed, but no other query modification or expansion methods were used. The run labeled `uwmt6a1` incorporates query expansion through the local feedback procedure described earlier. For expediency, we created this run by combining the results of three runs: `cdS`, `okS` and `fbfS` (not shown). The `fbfS` run was generated by applying the Okapi measure to the fully expanded queries. For this run, the terms were not stemmed. To create `uwmt6a1`, the document scores in `fbfS` and `okS` were first normalized and added, and the result was then combined with `cdS` by adding ranks. We submitted `uwmt6a1` as an official TREC-6 run; the run was not judged.

As expected, the performance of cover density ranking was comparable to that of the okapi measure. At low ranks the performance was slightly superior: precision at 5 documents was 0.4560 for `cdS` and 0.4000 for `okS`. Nonetheless, preliminary experiments had suggested that the difference would be more pronounced. Local feedback based on the passages identified during cover density ranking improved performance, but once again our preliminary experiments had indicated that the improvement would be more pronounced. An initial examination of the detailed results indicates that the coordination level at which a query first matches has a influence on the retrieval performance; this observation appears to be related to the differences between our preliminary runs and our TREC-6 results.

As required, we submitted a run based on the topic descriptions (`uwmt6a2`). This run used a weighted variant of cover density to select passages for local feedback. Final document scoring was based on the MultiText implementation of the Okapi measure.

4.3 Manual Adhoc

For the manual adhoc task, we explored the effects of large amounts of interaction using shortest substring ranking and an interface that displayed relevant passages and allowed judgements to be recorded (Figure 5). No limit was placed on interaction time; four people spent a total of 105 hours over eight days (an average of 2.1 hours/topic) creating queries and making judgements for the 50 topics. It was our aim to use the high precision track and the manual adhoc task to explore the limits of the effects of user interaction — from 5 minutes (maximum) per query to 2 hours (average) per query. In addition, we wished to investigate the viability of this approach for creating a reasonable pool of judged documents with minimal effort.

The aim of the searchers was to find and judge as many relevant documents as possible. In judging the documents, the searchers placed them into three categories: relevant, not relevant, and borderline or "iffy". Queries were constructed manually to find potentially relevant documents, which were displayed in order of shortest substring ranking. In general, all documents were judged in this order until it was felt that further relevant documents were unlikely to be found. For most topics, several queries were issued to investigate different aspects of the topic.

We rendered 13,064 judgements in the process of completing the manual adhoc task. Agreement between our relevance judgements and those of the TREC judges was 77%, treating iffy as not relevant. However, for those documents judged to be relevant, agreement was relatively poor. Of the 3900 we found relevant, 2812 (72%) were officially judged: 1912 (68%) relevant and 900 (32%) not relevant. That is, only 49% of the documents we found and judged relevant were found and judged relevant using TREC-6 pooling and judging. Of the 4611 documents found relevant by the TREC judges, we judged 2699 (59%): we found 1912 (71%) relevant, 302 (11%) not relevant, and 485 (18%) iffy. Of the documents officially found relevant, we judged and found relevant 41% and judged and found iffy another 11%. These numbers are shown as a Venn diagram in figure 6.

Our final submission was derived from our judgements and from the queries constructed in the judging process. The judgements partitioned the documents into four categories: relevant, iffy, not relevant, and unjudged. The first tier of our submission was restricted to those documents judged relevant. Within this tier, documents were ranked using a manually created query and shortest substring ranking. The second and subsequent tiers were restricted to some combination of iffy, not relevant, and unjudged documents, selected subjectively by the searchers. For some topics, iffy or not relevant documents were deemed to be more probably relevant than unjudged documents; for others, the opposite. For example, for topic 301 ("International Organized Crime") we felt there remained many unjudged relevant documents and these were ranked before those judged iffy. The norm, however, was to rank iffy documents before unjudged ones, and to rank not relevant documents after unjudged ones.

Figure 5: User Interface for Manual Adhoc Task

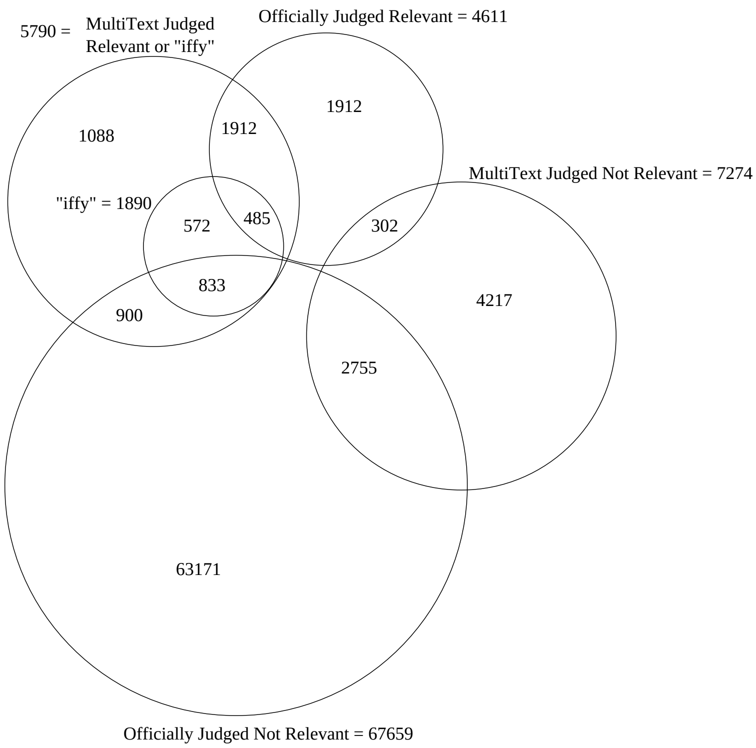

Figure 6: Judging Comparision

We examined user time, relevant documents, average precision, best average precision, and median average precision for all the topics. In general, large interaction times were correlated with large numbers of relevant documents, but not always correlated with precision. In particular, two topics for which high effort was expended and low precision results were achieved were topics 319 ("New Fuel Sources") and 301 ("International Organized Crime"). We attribute these particular results to misinterpretation of the topics by our searchers.

4.4 High Precision

For the high precision track, searchers attempted to find up to ten relevant documents within a fixed time period (five minutes). Any type of interaction was permitted. The MultiText high precision user interface is shown in Figure 7. The interface is based on "faceted" boolean queries. Each facet consists of a disjunction of terms; a query consists of a conjunction of facets.

A search begins when the participant selects the appropriate topic number. The five minute interval officially commences, the text of the topic is displayed, and the title of the topic is transformed into the initial query. The title is parsed into words and filtered against a list of stop words. The remaining words become distinct facets. Faceted queries are entered in the *Query Generator* window. Each line contains a facet and may contain any number of terms. Each term may be a word, a quoted phrase or even an arbitrary GCL query.

Once the query has been prepared, it is used to partition the documents in the database according to the maximal subset of the facets that occur in the document. For example, with facets A, B and C, we might partition the documents into the 8 possible sets ABC, $AB\bar{C}$, $A\bar{B}C$, etc. While

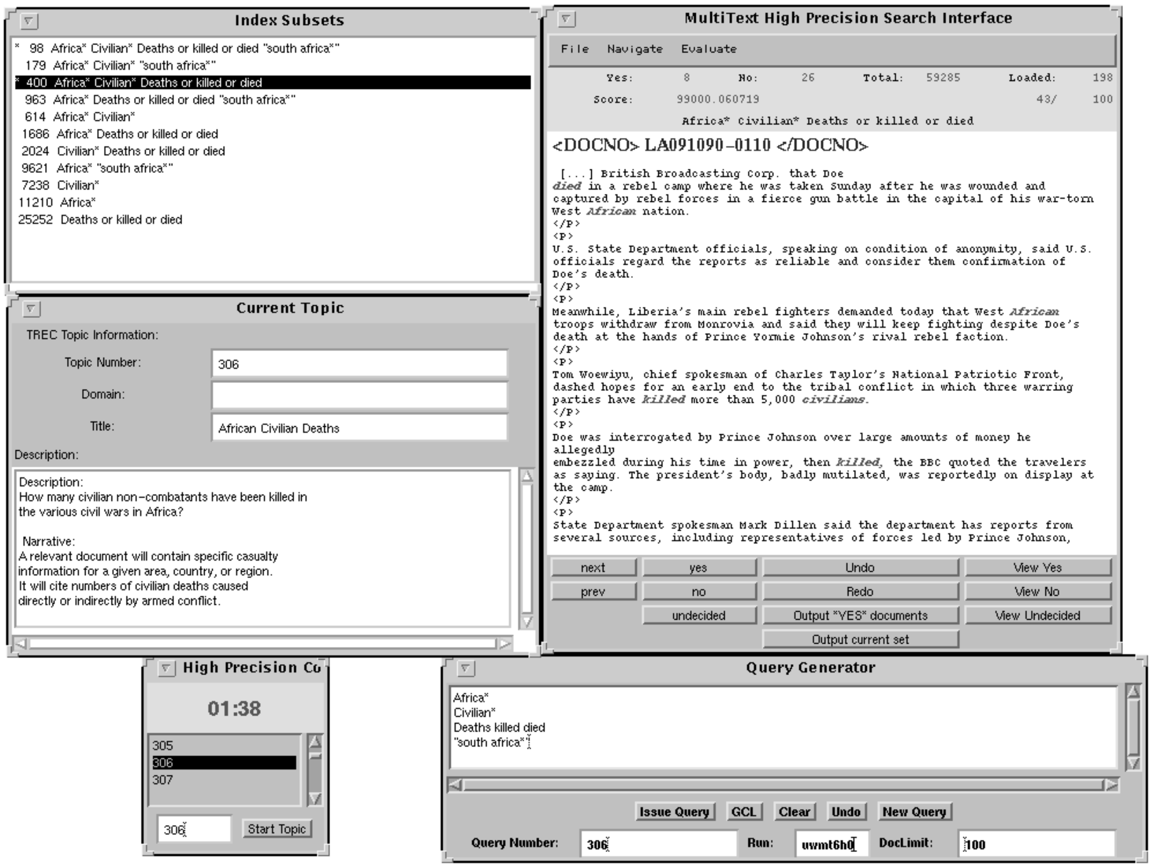

Figure 7: User Interface for High Precision Track

this could create an exponential number of sets, by using the maximal subset of the facets, in our experience, the number of subsets doesn't become unmanageable until there are approximately ten facets.

To summarize this partition, the *Index Subsets* window lists the cardinality of each set of documents and lists the facets present in the subset. For example, in figure 7 we see that there are 179 documents that contain "`africa*`", "`civilian*`", at least one of "`deaths`", "`killed`" or "`died`" but do not contain "`south africa*`". The list of subsets is sorted in descending order by the number of facets and then by the cardinality of the corresponding set of documents (in ascending order). This ordering tends to cluster the more promising sets at the top of the list.

Each entry in the subset list corresponds to a set of documents. The sets of documents may be individually navigated. Each set is ranked and displayed in order. Instead of displaying the entire document, the system displays relevant passages. The selected passages are those that contributed most to the score assigned to the document by the ranking function. The search terms in the facets will be displayed in a different color and typeface to facilitate locating the most relevant information in the passages. It is also possible to look at the entire document with the search terms in an alternate type and color. This feature was rarely used in the experiments. The passages were usually sufficient to determine a document's relevance.

Combining the subset construction and the ability to quickly read passages suggests a query

Run	Prec. @ 10	Rel. Prec @ 10	Avg Prec @ 10
uwmt6h0	0.5720	0.5977	0.0902
uwmt6h1	0.5680	0.5834	0.0982
uwmt6h2	0.5640	0.5951	0.0997
avg of best	0.7660	0.8084	0.1718
manual adhoc	0.6820		
avg of median	0.5340	0.5507	0.0782

Figure 8: High Precision Results

	Baseline (2GB)	Full (20GB)	Factor
Build Time (mins)	25.2	268.8	10.66
Query Time (secs)	11.34	67.3	5.93
Query Time (no overhead)	5.42	53.6	9.9
Time per topic (secs)	0.23	1.35	
Precision @ 20	0.498	0.643	

Figure 9: Very Large Corpus Results

refinement strategy. Starting from any given query, the user typically finds words that either strengthen or weaken the passage's likelihood of being relevant. For example, "shelling" may occur in many relevant passages and the user can then construct subsets that contain this word. Alternatively, passages about South Africa tend to be about police actions and therefore may not be relevant. By splitting the subsets on the phrase `"south africa*"`, the user can separate the subsets not containing South Africa from those that do. This refinement process is an interactive, manual version of the re-tiering used in the routing task. The searcher uses knowledge about the terms and information observed in the passages to build a mental model that resembles the relevance graphs described in Section 4.1

The table of Figure 8 compares the three submissions to the average of the best result for each topic and the median result for each topic. All runs were above the median. At least one of the three runs achieved the best precision @ 10 for 33 of the topics. At least two of the three runs tied with the best precision @ 10 for 12 of the topics. However, there is a high variance between topics; the average precision is 25% below the average of the best results. This indicates that certain users have problems with certain topics. It may be appropriate to work at integrating approaches used in the automatic adhoc to attempt to assist users when they do encounter difficulties.

4.5 Very Large Corpus

The MultiText system was designed as a distributed, scalable text database system. The Very Large Corpus track allowed experiments in software scalability and demonstrated the viability of the distributed architecture. The 2GB and 20GB collections of data were distributed over four inexpensive PCs. Queries were generated manually by interacting over the adhoc data. The official run measured software scalability and post-hoc experiments validated hardware scalability. The table of Figure 9 shows our VLC results.

To ensure that the baseline and full collections runs were comparable, the data was partitioned over the four PCs in an identical fashion. The four machines used a Cyrix processor (Pentium clone), three 3.8GB EIDE hard-drives, 64MB memory and consisted of inexpensive components.

No. Machines	1 (5GB)	2 (10GB)	3 (15GB)	4 (20GB)
Query Time (secs)	62.1	63.3	64.2	64.7
% Increase		1.9%	1.4%	0.8%
% Increase over 1 node		1.9%	3.4%	4.1%

Figure 10: Hardware Scalability

The machines ran a version of the Linux operating system. Most of the available storage was actually used. One drive held the system files as well as the original data. One drive held the index for 5GBs of data. The last drive held the document numbers for the 5GBs of data as well as the index and document numbers for the baseline data (500MBs). The remaining storage was used as temporary space during the original build.

Distributing the data over multiple nodes allows for parallel query processing. Since there are no collection statistics and every document is independently ranked, each processor can act independently. A simple marshaller/dispatcher can be used to dispatch a query to each of the machines and to collect the results by merging the lists of ranked documents. The marshaller/dispatcher can then contact the appropriate text databases in parallel to locate the document identifiers to provide the final answer to the query. The marshaller/dispatcher can be scaled to any number of nodes with a small per-machine overhead. The table of Figure 10 shows that there is only a moderate increase in time as nodes are added to the distributed database.

Database creation trivially scales to any number of nodes because each node builds a normal database. That is, each node will contain a valid database that may be accessed independently of the entire collection. The software scalability for the build is linear. Fixed sized hash tables and buffers are used to build partial indices that are written out to disk as the available memory is exhausted. The final step of the build is to merge the partial indices into the final index. Since I/O costs dominate, the total build time would be expected to scale approximately linearly with the size of the data. This expectation was observed in practice, with a 10.66 times increase in the build time when 10 times more data was added.

The queries were generated by interacting over the adhoc data using the adhoc queries as our basis. Short and precise queries were created. The queries averaged approximately 5.5 terms per topic. To improve performance, length restrictions were added to the queries to discard spurious solutions. Based on the probability estimations used in the routing task, discarding these solutions was not expected to affect retrieval effectiveness. The queries ran for an average of 1.35 seconds/topic. The GCL query algebra can be implemented in time that is at most linear in the length of the postings lists. Consequently, we expected a linear increase in time from the baseline to the full run. In the final analysis, the time taken to retrieve the document numbers was a significant overhead. It was this overhead that led to a sub-linear time increase. If we consider only the query processing time, we see the expected factor of 9.9 times.

There was increased precision in the full run over the baseline run. This increase is evidence that the ranking function is stable. Given more data, it is expected that a ranking method should find more relevant documents within a given rank. This expectation is consistent with the routing relevance graphs. Since there are more documents with higher scores, there are more documents that have a higher probability of relevance.

4.6 Chinese

This is the first year that we participated in the Chinese track. Our main goal was to evaluate how our approaches to ranking documents will perform in Chinese information retrieval environments.

As there are no explicit word boundaries in Chinese text, a decision has to be made on how to index the documents. We decided to index the collection as individual characters. To search for a word, we relied on the phrase searching capability provided by the MultiText retrieval system and looked for a sequence of adjacent characters. This post-coordination approach is well supported by our retrieval system and we believe it is more flexible than indexing segmented text. We can also avoid many of the problems with segmentation, such as the need for a large dictionary, the potential for erroneous character groupings, and the difficulty in handling new terms and proper nouns [10].

The queries for each topic were constructed manually in an interactive manner by one member of our research group. For each topic, an initial query was formulated and submitted to the retrieval system, and the top 10-15 documents returned were evaluated manually for their relevance. Additional terms that appeared salient in the relevant documents were added to the query, and terms that did not seem to be useful were eliminated. This process was iterated 2-3 times before the final version of the query was constructed.

For most topics, instead of using a single query that encompass the different aspects of the topic, a series of queries were formulated with each query focusing on a particular aspect of the topic. For example, several queries were constructed for topic 49: one focused on the extension of the non-proliferation treaty on nuclear arms, one focused on underground nuclear tests, and another focused on the promotion of peaceful use of nuclear power. If the queries all focused on important aspects of the topic they were combined disjunctively in a single tier. If some queries were considered more important than others, tiered ranking was used to put these queries ahead of the more peripheral ones. The queries for each topic took about 1.5-2 hours to formulate. The final queries would be suitable for use in a future routing task.

The results of our only run, `uwmt6c0`, were encouraging. We were above the median in average precision in 21 of the 26 topics, while finishing top in 7 of them. The results show that our ranking approach performs well for Chinese document collections.

References

[1] Charles L. A. Clarke and Gordon V. Cormack. Interactive substring retriveal. In *Fifth Text REtrieval Conference (TREC-4)*, 1996.

[2] Charles L. A. Clarke, Gordon V. Cormack, and Forbes J. Burkowski. Schema-independent retrieval from hetrogeneous structured text. In *Fourth Annual Symposium on Document Analysis and Information Retrieval*, pages 279–289, Las Vegas, Nevada, April 1995.

[3] 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.

[4] 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.

[5] W. Bruce Croft, Robert Cook, and Dean Wilder. Providing government information on the Internet: Experiences with THOMAS. In *Digital Libaries Conference (DL'95)*, pages 19–24, Austin, Texas, June 1995.

[6] S. E. Robertson and S. Walker. Some simple effective approximations to the 2-Possion method for probabalistic weighted retrieval. In *17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'94)*, pages 311–317, Dublin, July 1994.

[7] S. E. Robertson, S. Walker, S. Jones, M. M. Hancock-Beaulieu, and M. Gatford. Okapi at TREC-3. In *Third Text REtrieval Conference (TREC-4)*, pages 109–126, 1994.

[8] Daniel E. Rose and Curt Stevens. V-Twin: A lightweight engine for interactive use. In *Fifth Text REtrieval Conference (TREC-5)*, 1996.

[9] Ross Wilkinson, Justin Zobel, and Ron Sacks-Davis. Similarity measures for short queries. In *Fourth Text REtrieval Conference (TREC-4)*, pages 277–285, 1995.

[10] Zimin Wu and Gwyneth Tseng. Chinese text segmentation for text retrieval: Achievements and problems. *Journal of the American Society for Information Science*, 44(9):532–542, 1993.