Interactive Substring Retrieval
(MultiText Experiments for TREC-5)

Charles L. A. Clarke* Gordon V. Cormack

MultiText Project
Department of Computer Science
University of Waterloo
Waterloo, Ontario N2L 3G1, Canada
mt@plg.uwaterloo.ca
http://multitext.uwaterloo.ca

Abstract

Queries for TREC-5 were formulated in the GCL query language using an interactive system that showed short passages containing relevant terms. Solutions to the queries were ranked by the shortest substring method introduced at TREC-4, resulting in good precision/recall performance in the adhoc and routing tasks. Performance results were found to be insensitive to a document length normalization adjustment. Shortest substring ranking was augmented by the use of a progression of successively weaker queries to improve recall, but this augmentation provided only a slight improvement to overall retrieval effectiveness.

1 Introduction

For TREC-5, the MultiText Project focused on the development of interaction methods for query creation and refinement using a ranking technique introduced at TREC-4. In the MultiText system, ranking is based entirely on term proximity, with queries expressed in a boolean subset of the general structured text retrieval language GCL [3, 4, 5]. In the GCL query language, solutions to queries represent intervals of the underlying text and are not restricted to reporting pre-defined elements such as documents, paragraphs or sentences. For ranking purposes, a document's score is based on the size and density of the query solutions contained within it. The development and initial validation of this ranking technique was the subject of our participation in TREC-4 [6].

Since relevance depends on term proximity, the technique may be used to identify small, relevant passages. During interactive query development, these passages are presented to the user for the selection of additional query terms and to drive other refinements. Initial exploration of this interaction method was the primary subject of our TREC-5 work.

Hearst [10] argues that efforts in interactive information retrieval should be directed toward "providing the user with information that is informative and compact enough that it can be interpreted swiftly." Toward this end, Hearst presents a visualization tool that summarizes term distribution and density within documents in an iconic form. Interactive retrieval and presentation of short relevant passages may be considered as a complementary technique, providing the user with specific textual context. Along these lines, interactive passage retrieval has been used to speed up

* *Author's current address:* Department of Electrical and Computer Engineering, University of Toronto.

the process of making relevance judgements [12] and has been suggested for avoiding user overload when dealing with long documents [14].

Apart from this use in interactive retrieval, passage-level evidence has been widely used to improve retrieval effectiveness [1, 2, 11, 14, 17, 18]. Research in this area has generally assumed that documents must be divided *a priori* into passages. Wilkinson [17] based passage retrieval on explicit structural elements — such as abstracts, titles and summaries — described by the internal document markup. Callan [2] compared passages based on paragraphs with passages based on fixed-length text segments. Wilkinson and Zobel [18] based retrieval on a variety of passage types, including sentences, paragraphs, sequences of paragraphs, and sections defined by document markup. Hearst and Plaunt [11] used a similarity measure between successive blocks of text to partition documents according to their subtopic structure.

In contrast, our approach to interactive retrieval bases the decomposition into passages on the query itself. High-scoring passages satisfying the query are presented to the user, who may then select additional terms from these passages for inclusion in the query. This approach to interactive query expansion, with the user selecting from suggestions made by the system, is similar to that taken by Koenemann and Belkin [13], using terms suggested by relevance feedback, and by Harman [7], using terms gathered from relevance feedback, reverse stemming and nearest neighbors procedures.

2 The GCL Query Language

The GCL query language was initially developed for structural document retrieval. Text in a database is viewed as a long sequence of words. Solutions to queries are the intervals of this text sequence that satisfy the conditions of the query, expressed as start and end positions within the sequence. In order to limit the total number of solutions, 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. Structural information — for example, the start and end positions of documents, titles, paragraphs, headlines and document identifiers — are indexed at appropriate positions within the word sequence but do not necessarily occupy positions in the sequence.

The simplest GCL query is a phrase, for example:

```
"heart attack"
```

The solution to this query is the set of intervals corresponding to the locations of the phrase within the word sequence.

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. Document boundaries require no special treatment. 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

```
("angiogram"..."bypass surgery") contained in ("<doc>"..."</doc>")
```

finds occurrences of "angiogram" followed by "bypass surgery" within the same document. Note that elements in the solution set for this query are text intervals, not documents.

The query language includes the boolean operators "`and`" and "`or`" interpreted over intervals of text under the shortest substring rule. Rather than selecting members from a document set, as might normally be expected, the query

```
"angiogram" and ("bypass surgery" or "angioplasty")
```

specifies intervals of text that satisfy the boolean expression. Ranking of documents containing solutions to this query would be based on the length and density of these solution intervals.

Further details of the query language, its implementation and the underlying theory may be found elsewhere [3, 4, 5].

3 Shortest Substring Ranking

The ranking technique is based on two assumptions:

A) the shorter a solution interval, the greater the likelihood that the interval is relevant

B) the more solution intervals within a document, the greater the likelihood that the document is relevant

An interval in the text is represented by an range or *extent* (p, q), where p is the start position of the interval within the word sequence and q is the end position of the interval. An extent is assigned a score using the formula:

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

A value of 16 for the parameter $\mathcal{K}$ has been found to give reasonable performance. Using this formula, intervals of 16 words or smaller are assigned a score of 1, and larger intervals are scored in proportion to the inverse of their length.

If a document D contains solution extents (p_1, q_1), (p_2, q_2), ..., (p_n, q_n) the score for the document is taken to be the sum of the scores of these extents:

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

Query evaluation and ranking proceeds in five steps:

1. Determine extents which satisfy the query: `Q`

2. For each solution extent, either determine the document extent that contains it, or if it overlaps document boundaries eliminate it:

   ```
   ("<doc>"..."</doc>") containing Q
   ```

3. For each document containing solution extents, determine the identifier for the document:

```
("<docno>"..."</docno>")
        contained in (("<doc>"..."</doc>") containing Q)
```

4. For each document containing solution extents, calculate a document score using the scoring method above.

5. Sort the document identifiers by document score.

Shortest substring ranking has several important properties. A document's score is independent of the characteristics of other documents in the collection, making special consideration for distributed collections unnecessary. Phrase queries, structural queries and boolean queries are naturally integrated in the query model. Any set of document elements that can be defined by a GCL query may be ranked, without any need for supplementary indexing. Since the scoring technique is based on term proximity, short relevant passages may easily be identified. This last property forms the bases for the interaction method.

Apart from the work of the MultiText Project, document ranking based solely on term proximity has been the subject of research at the Australian National University [8, 9]. Although details of the scoring technique differ, that work supports our view that ranking can reasonably be based on term proximity alone.

4 Interactive Substring Retrieval

Under shortest substring ranking, scores are the output of an evaluation procedure executed over predetermined text intervals (e.g. documents). For interactive retrieval, a *score threshold* becomes an input parameter to an evaluation procedure whose result is the smallest intervals achieving the threshold score. Given a query Q, let

$$(p_1, q_1),\ (p_2, q_2),\ ...,\ (p_m, q_m)$$

be the solution extents for Q ordered by increasing position in the word sequence. Given a threshold score (*Threshold*) the algorithm in Figure 1 computes the shortest intervals that meet or exceed the threshold and lie within a single document.

The algorithm makes a single scan over the solution extents. The variables i and j represent the start and end positions of a range of solution extents. The first **while** loop advances j until the score of the interval between (p_i, q_i) and (p_j, q_j) exceeds the threshold value. The second **while** loop then advances i until any further advance would drop the score below the threshold. All intervals containing (p_i, q_j) meet or exceed the threshold; all intervals contained in (p_i, q_j) have scores lower than the threshold. If (p_i, q_j) is contained within a single document, it is presented to the user.

The solution extents (p_1, q_1), ..., (p_n, q_n) are generated in word sequence order by GCL and may be generated one at a time as needed for the the threshold passage retrieval algorithm. In turn, the algorithm can generate intervals exceeding the threshold without requiring the full set of solution extents. For interactive query refinement, the algorithm may be used to quickly generate a small number of high-scoring passages for presentation to the user, without a need to invoke the full ranking process.

```
i ← 1;
j ← 0;
Score ← 0.0;
loop
   while Score < Threshold do
      if j = m then stop; end if;
      j ← j + 1;
      Score ← Score + I(p_j, q_j);
   end while;
   while Score − I(p_i, q_i) ≥ Threshold do
      Score ← Score − I(p_i, q_i);
      i ← i + 1;
   end while;
   if {(p_i, q_j)} contained in "<doc>"..."</doc>" then
      Display (p_i, q_j);
   end if;
   if i = m then stop; end if;
   Score ← Score − I(p_i, q_i);
   i ← i + 1;
end loop;
```

Figure 1: Threshold Passage Retrieval

5 MultiText Experiments for TREC-5

The MultiText Project participated in both the routing task and the adhoc task. Queries were developed manually by the authors working in conjunction. Approximately 45 minutes were spent developing a query for each topic. For the routing task the queries were developed interactively over a collection consisting of the documents from TIPSTER disks #2 and #3. Relevance judgements were indexed as part of the document structure and could be directly referenced in queries. For example, the query

```
(("<doc>"..."</doc>") containing "<+78>") not containing "greenpeace"
```

specifies documents that have been judged relevant to topic 78 that do not contain the word “greenpeace”. The query

```
(("<doc>"..."</doc>") containing "<-78>") containing "greenpeace"
```

specifies documents that have been judged not relevant to topic 78 that contain the word “greenpeace”. For the adhoc task the queries were developed interactively over the collection consisting of TIPSTER disk #2 and TREC disk #4. For adhoc query development, interaction was used primarily for term expansion purposes, and the resulting queries are not intended to incorporate specific relevance judgements made during interaction. The final queries would be suitable for use in a future routing task. Besides the authors' personal knowledge of the topics, the only external resources used were Webster's on-line dictionary, the Unix `spell` program, and an on-line list of country, state and city names and state postal abbreviations.

Figure 2 shows our user interface for interactive query development, with the results of a query for topic 272 displayed. The first result is displayed with a small window of additional text surrounding the high-scoring passage and with terms from the query highlighted.

The final query developed for each topic was a compound query consisting of an ordered list of one or more sub-queries (1.79 sub-queries on average). Ranking for each sub-query was first determined separately, and the results were then combined into a final solution set according to the ordering of the sub-query list. The sub-query list was processed in order. If a document was given a non-zero score by a sub-query, it was eliminated from the results of subsequent sub-queries in the final ranking.

This approach reflects a trade-off between a desire for precision and the need to produce 1000 ranked documents. The query appearing at the beginning of the list is intended to be a precise expression of the requirements underlying the topic. Queries occurring later in the list are “weaker” and are intended to pick up a large number of possibly relevant documents. The effects of using these weaker, secondary sub-queries will be examined in the next section.

The number of terms used in the final queries varied from one to more than a hundred. For both tasks, queries contained an average of 44 terms. The large number of terms was partially due to the explicit listing of morphological variants and partially due to the repeated use of a pre-defined sub-query listing 150 geographical place names within the United States.

The official MultiText routing run (uwgcr0) and one of the official MultiText adhoc runs (uwgcx0) used the scoring method described in Section 3. The second official adhoc run (uwgcx1) used a variation of this scoring technique, described in Section 5.2 below, that attempts to normalize scores to account for document length.

The experiments were run on a dedicated DEC Alpha 2000/300 workstation running OSF V3.2 with 64MB of RAM.

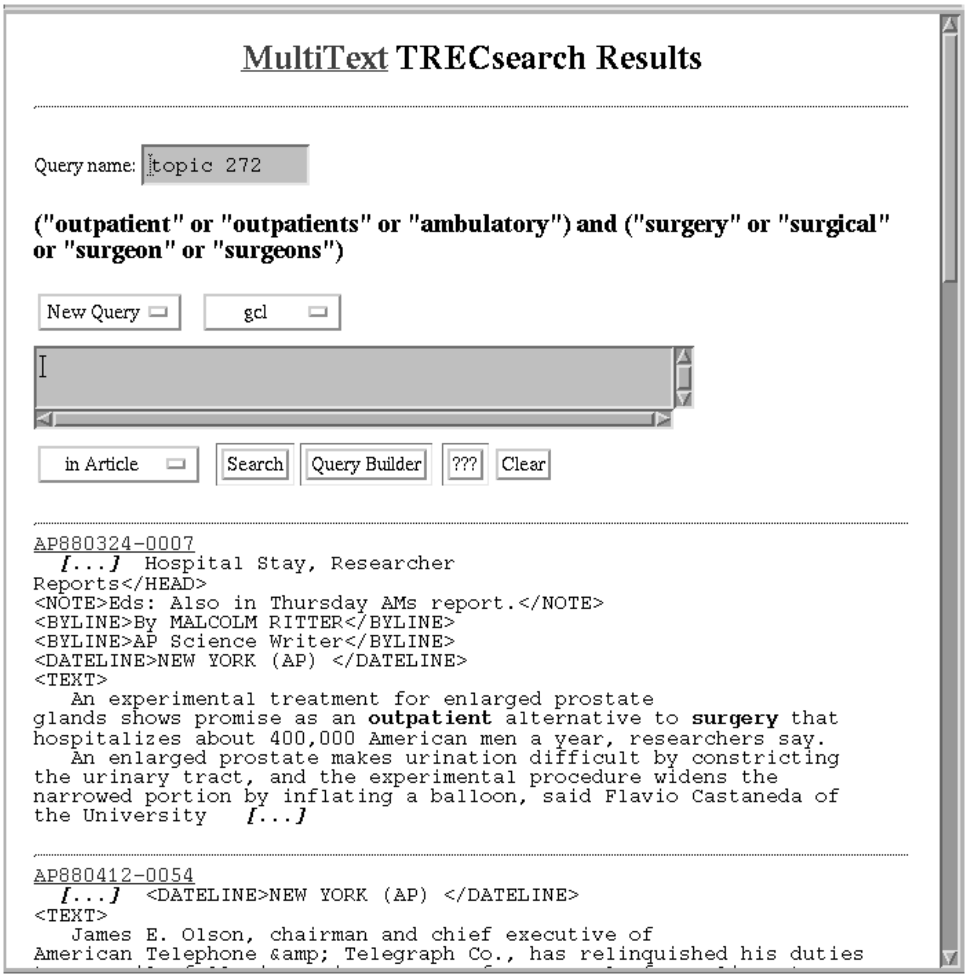

Figure 2: Interactive Retrieval Interface

5.1 Sub-Query Lists

Each query used in the MultiText runs consisted of an ordered list of one or more sub-queries. The first of these sub-queries was intended to be a "high-precision" query, an accurate expression of the user requirements embodied in the topic. The additional sub-queries were intended to increase overall recall. After the results of the first query were ranked, the remaining sub-queries were evaluated in order. New documents introduced by these sub-queries were ranked after the documents matching the first sub-query.

As part of our TREC-5 work, we performed three unofficial runs intended to assess the impact of these additional sub-queries. Recall-precision curves for the runs are plotted in figure 3, along with the curve for the corresponding official TREC-5 adhoc run (uwgcx0).

For the first unofficial run (plotted with "+") we eliminated the additional sub-queries, and used only the first sub-query for retrieval and ranking. As expected, overall recall was reduced. Our official run retrieved 3072 relevant documents, but the first sub-queries alone retrieved only 2781 relevant documents. The second unofficial run (plotted with "□") used all sub-queries but did not use shortest substring ranking. Documents were ranked according to the sub-query that first retrieved them, with documents first retrieved by earlier sub-queries given higher scores. All documents first retrieved by a particular sub-query were given the same score, and their relative ordering was determined only by their position in the database. The difference in retrieval effectiveness between this run and the official run is dramatic, showing substantially lower precision at all recall levels.

The third unofficial run (plotted with "×") used the first sub-query only, treating it as a traditional boolean query. For this run, documents were ordered only by their position in the database. The improvement seen in retrieval effectiveness between this run and the first unofficial run ("+") is entirely due to shortest substring ranking.

5.2 Document Length Normalization

Users of ranked retrieval in the MultiText system have noticed a slight bias for longer documents. Solely because of length, a long document may contain many more high-scoring solution extents than a short document with similar relevance, and receive an inappropriately higher score. It has been noted by others [15] that in the TREC collection a longer document is more likely to relevant than a shorter document. Nonetheless, we viewed the lack of *any* document length normalization as a possible deficiency in the scoring method.

In preliminary experiments, normalization of scores by dividing by document length appeared to excessively penalize long documents with short relevant passages. However, a weak normalization, dividing by the square root of the document length, appeared to improve precision among high ranking documents.

The high variance in document length found in the adhoc collection provided an opportunity to test this document normalization technique. For our second official adhoc run (uwgcx1) documents were scored using the formula

$$S'(D) = \frac{S(D)}{\sqrt{|D|}}$$

where $|D|$ is the length of document D. The other official run (uwgcx0) used the scoring procedure of Section 3.

Document length normalization had little impact on overall retrieval effectiveness. Figure 4 plots recall and precision for the two runs. The largest difference in precision occurs at the 10% recall level, where the normalized run achieves a slightly higher precision (0.6024 vs. 0.5833).

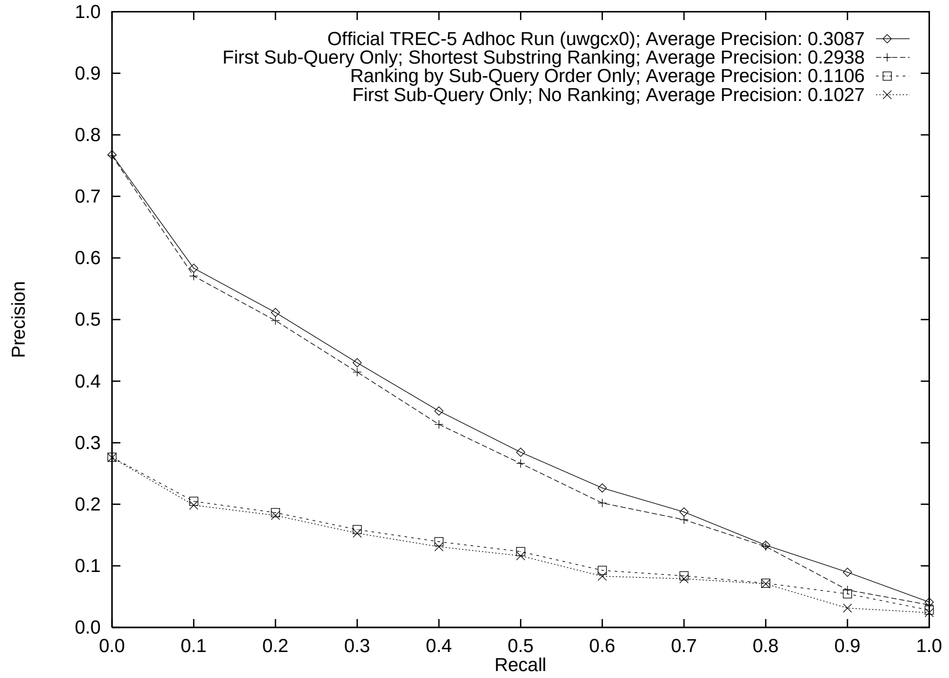

Figure 3: Effects of Sub-Query Lists

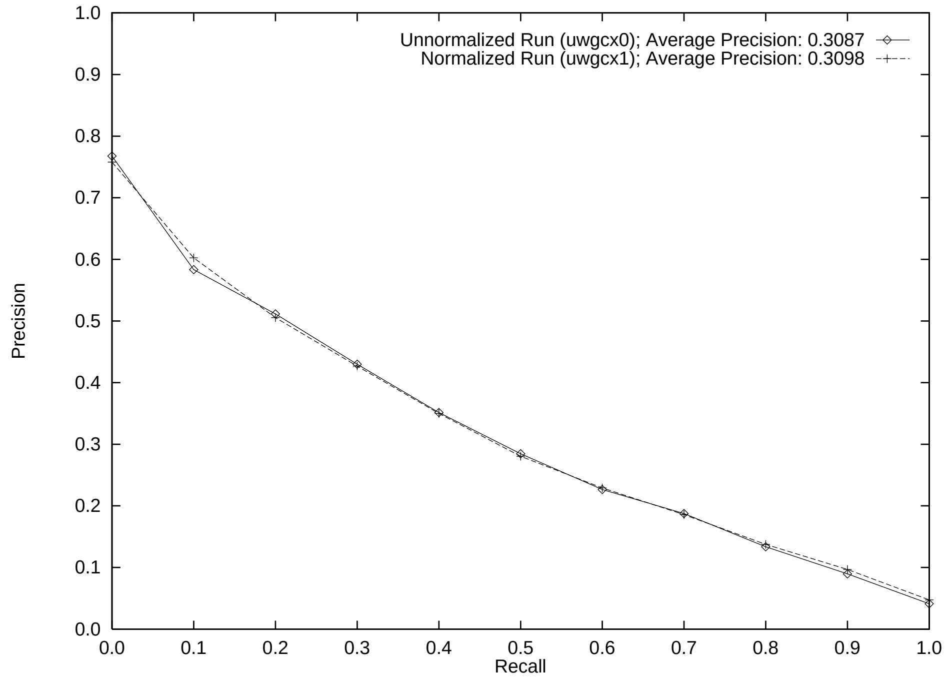

Figure 4: Effects of Document Length Normalization

6 Concluding Remarks

The work described in this paper was undertaken as part of the MultiText Project. The MultiText Project develops and prototypes new technologies associated with large-scale document database systems. Research issues include fault tolerance, distributed data management, dynamic update, and support for heterogeneous document structure. Prototype development is based around a federated architecture in which a group of small specialized servers under common administration present the unified services of a large, highly-reliable document database system to external clients. Emphasis is placed on flexibility. A large-scale system must store documents in wide variety of formats with multiple user interfaces. In addition, the federated architecture can provide an increase in efficiency and reliability if appropriate data organization and replication methods are used.

Part of the ongoing work of the MultiText Project is directed toward the creation of tools for interactive query refinement. We are particularly interested in query expansion techniques, including the work described in this paper and the use of stemming algorithms for interactive term expansion [16]. Our TREC-5 work provided preliminary experience with our approach. For TREC-6 we hope to perform a more substantial evaluation of our tools, particularly through participation in the Interactive and High-Precision tracks. In addition, given our overall focus on large-scale document database systems, we are planning to participate in the Very Large Collection track.

Acknowledgements

Rob Good provides technical and system administration support for the MultiText Project and gave some much-needed last-minute assistance with data indexing for our TREC-5 work. Other people who have contributed to the MultiText Project include Forbes Burkowski, Elizabeth Tudhope, Nikita Borisov and Chris Palmer. The first author would like to thank Rob Pitt of Canning & Pitt Associates, St. John's, Newfoundland for access to office space and computer facilities during an unexpected and lengthy stay in that city overlapping the TREC-5 data submission deadline.

The MultiText Project receives its primary funding from the Information Technology Research Centre, Ontario. Additional funding was provided by the Natural Sciences and Engineering Research Council of Canada.

References

[1] James Allan. Relevance feedback with too much data. In *18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'95)*, pages 337–343, Seattle, July 1995.

[2] James Callan. Passage-level evidence in document retrieval. In *17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'94)*, pages 302–310, Dublin, July 1994.

[3] Charles L. A. Clarke. *An Algebra for Structured Text Search.* PhD thesis, University of Waterloo, Waterloo, Ontario, 1996.

[4] Charles L. A. Clarke, G. V. Cormack, and F. J. Burkowski. An algebra for structured text search and a framework for its implementation. *The Computer Journal*, 38(1):43–56, 1995.

[5] 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 (SDAIR'95)*, pages 279–289, Las Vegas, April 1995.

[6] Charles L. A. Clarke, Gordon V. Cormack, and Forbes J. Burkowski. Shortest substring ranking (MultiText experiments for TREC-4). In *Fourth Text REtrieval Conference (TREC-4)*, Gaithersburg, Maryland, November 1995.

[7] Donna Harman. Towards interactive query expansion. In *11th International Conference on Research and Development in Information Retrieval (SIGIR'88)*, pages 321–331, Grenoble, France, June 1988.

[8] David Hawking and Paul Thistlewaite. Proximity operators — So near and yet so far. In *Fourth Text REtrieval Conference (TREC-4)*, Gaithersburg, Maryland, November 1995.

[9] David Hawking and Paul Thistlewaite. Relevance weighting using distance between term occurrences. Computer Science Technical Report TR-CS-96-08, Australian National University, August 1996.

[10] Marti A. Hearst. TileBars: Visualization of term distribution information in full text information access. In *ACM SIGCHI Conference on Human Factors in Computing Systems (CHI'95)*, pages 59–66, Denver, May 1995.

[11] Marti A. Hearst and Christian Plaunt. Subtopic structuring for full-length document access. In *16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'93)*, pages 59–68, Pittsburgh, July 1993.

[12] Daniel Knaus, Elke Mittendorf, Peter Schäuble, and Páraic Sheridan. Highlighting relevant passages for users of the interactive SPIDER retrieval system. In *Fourth Text REtrieval Conference (TREC-4)*, Gaithersburg, Maryland, November 1995.

[13] Jürgen Koenemann and Nicholas J. Belkin. A case for interaction: A study of interactive information retrieval behavior and effectiveness. In *Proceeding of the ACM SIGCHI Conference on Human Factors in Computing Systems*, pages 205–212, Vancouver, BC, April 1996.

[14] Gerard Salton, J. Allan, and C. Buckley. Approaches to passage retrieval in full text information systems. In *16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'93)*, pages 49–58, Pittsburgh, July 1993.

[15] Amit Singhal, Gerard Salton, Mandar Mitra, and Chris Buckley. Document length normalization. *Information Processing and Management*, 32(5):619–633, 1996.

[16] Elizabeth Tudhope. Query based stemming. Master's thesis, University of Waterloo, Waterloo, Ontario, 1996.

[17] Ross Wilkinson. Effective retrieval of structured documents. In *17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'94)*, pages 311–317, Dublin, July 1994.

[18] Ross Wilkinson and Justin Zobel. Comparison of fragmentation schemes for document retrieval. In *Third Text REtrieval Conference (TREC-3)*, pages 81–84, Gaithersburg, Maryland, November 1994.