Shortest Substring Ranking
(MultiText Experiments for TREC-4)

Charles L. A. Clarke   Gordon V. Cormack   Forbes J. Burkowski

MultiText Project
Department of Computer Science
University of Waterloo
Waterloo, Ontario, Canada
mt@plg.uwaterloo.ca

Abstract

To address the TREC-4 topics, we used a precise query language that yields and combines arbitrary intervals of text rather than pre-defined units like words and documents. Each solution was scored in inverse proportion to the length of the shortest interval containing it. Each document was scored by the sum of the scores of solutions within it. Whenever the above strategy yielded less than 1000 documents, documents satisfying successively weaker queries were added with lower rank. Our results for the ad-hoc topics compare favourably with the median average precision for all groups.

1 Introduction

The central concern of the MultiText project at the University of Waterloo is the management of data in large-scale distributed text database systems [10]. A major component of this work has been the development of a query language that is suitable for expressing queries over the heterogeneous data that is present in a very large text database. The query language developed for the MultiText project, called GCL, provides for the general expression of containment and ordering relationships between document components, phrase searching, and boolean searches solved independently of containment in documents or other predetermined components. A solution to a GCL query is an interval of text, which may or may not correspond to a document component.

Prior to TREC-4, development of GCL focused on the properties of the language that provide precise and simple query semantics and flexible retrieval from structured text. For our TREC-4 experiments we have addressed a different issue, focusing on how the properties of the language might be exploited to rank documents, or components of documents, when queries are expressed primarily in a boolean-flavored subset of GCL.

2 Overview of the GCL Query Language

In a traditional text database system, a simple boolean query

```
"future" AND ("vision" OR "prediction")
```

selects documents that satisfy a boolean expression, in this case by containing the term "future" and one of the terms "vision" or "prediction". In GCL the equivalent boolean expression is not solved with respect to any pre-defined component. Instead, the results of a query are simply the smallest intervals of text that satisfy the expression.

Within GCL, the results of a query may be used to select elements from the results of a second query using containment relationships, allowing queries to be solved in terms of arbitrary document components. For example, the GCL query:

```
P2 = "<paragraph>" ... "</paragraph>"  ... "</paragraph>"
```

specifies pairs of paragraphs and names the result "P2". The ordering operator "..." associates the start and end tags for the paragraphs that are presumed to exist in the text. Combining these queries with the GCL "`containing`" operator selects pairs of paragraphs that satisfy the boolean expression:

```
P2 containing ("future" AND ("vision" OR "prediction"))
```

Other GCL operators express other containment relationships: *contained in*, *not containing* and *not contained in*. All GCL operators are completely general and orthogonal; any query may be used as an operand to any operator. The query

```
("future" AND ("vision" OR "prediction")) contained in P2
```

returns intervals of text that satisfy the boolean expression and are contained within two paragraphs. GCL can search for phrases as well as single terms. The more unusual query

```
"luck of the draw" AND P2
```

finds intervals of text containing the phrase "luck of the draw" along with two paragraphs close-to or containing it, depending on whether the phrase appears in a paragraph.

A complete definition and discussion of GCL and its features appears elsewhere [4]. A precise formal semantics and implementation framework for GCL is also available [3].

3 Shortest Substring Solution Model

We treat the text in the database as a continuous sequence of *terms*, or *tokens*, each corresponding to a word or number in the text. Each term is assigned an integral position in this sequence.

Figure 1 represents a simple database containing documents related to the food service industry. The data was obtained through our local commercial contacts in the Waterloo area. The text is marked up using a variant of SGML[1]. Figure 2 shows the mapping of the documents into the

[1] Smilin' Guy Markup Language

database. Markup indicating interdocument boundaries ("☺") is indexed between words at positions $\frac{1}{2}$, $10\frac{1}{2}$, $17\frac{1}{2}$, $26\frac{1}{2}$, $37\frac{1}{2}$, and $45\frac{1}{2}$. We will use this database as an on-going example throughout the next several sections of the paper.

Given a query Q, an *extent* satisfying Q is a pair (p, q) of positions in the text such that the substring of the text database beginning at position p and ending at q satisfies the query. For simple boolean queries:

1. An extent (p, q) satisfies a query Q_1 `AND` Q_2 if the extent satisfies Q_1 and satisfies Q_2.
2. An extent (p, q) satisfies a query Q_1 `OR` Q_2 if the extent satisfies Q_1 or satisfies Q_2.
3. An extent (p, q) satisfies a term T if the term occurs in the interval of text represented by the extent.

From these definitions it is clear that a great many extents may satisfy a particular query. For example, if there is any extent that satisfies a query then the extent corresponding to the entire database also satisfies the query. For a query consisting of a single term, any extent in the database that overlaps an occurrence of the term satisfies the query. For the database of figure 2, the query

```
"you" AND "will"
```

is satisfied by the extents $(1, 12)$, $(2, 12)$, $(3, 12)$, $(12, 45)$ and $(34, 45)$ among others — there are hundreds — but not by $(1, 11)$, $(12, 17)$ or $(35, 45)$.

From the large number of extents that may satisfy a query, we take as solutions only those that have no other satisfying extents contained within them. That is, from a set of extents S satisfying a query we accept as solutions the set of extents $\mathcal{G}(S)$, where

$$\mathcal{G}(S) = \{(p, q) \mid (p, q) \in S \text{ and } \nexists\ (p', q') \in S \text{ such that } (p, q) \neq (p', q'),\ p \leq p' \text{ and } q \geq q'\}.$$

In the case of our example database the solutions are $(11, 12)$, $(12, 18)$, $(18, 19)$, $(19, 27)$, $(27, 28)$, and $(34, 36)$.

4 Ranking by Solution Density

When our work began for TREC-4 we had no experience with ranking solutions to GCL queries. Although adapting traditional ranking techniques to GCL was one possible route, we felt that the unique properties of GCL, particularly the shortest substring search model, might lead to the successful development of a more novel method. Although we are quite happy with the results of our TREC experiments, we wish to emphasize that the techniques we used, described in this section, are highly experimental and were invented within the timeframe of TREC-4, given impetus by the necessity that our participation created.

After some preliminary work we decided to base our ranking on two assumptions:

- *Assumption A*
  The smaller a solution extent, the more likely that the corresponding text is relevant.

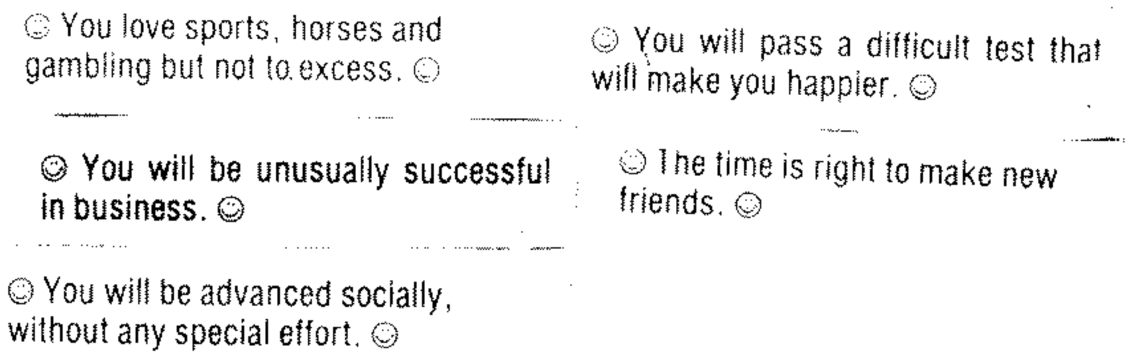

Figure 1: Example Source Text

1	2	3	4	5	6	7	8	9
you	love	sports	horses	and	gambling	but	not	to
10	11	12	13	14	15	16	17	18
excess	you	will	be	unusually	successful	in	business	you
19	20	21	22	23	24	25	26	27
will	be	advanced	socially	without	any	special	effort	you
28	29	30	31	32	33	34	35	36
will	pass	a	difficult	test	that	will	make	you
37	38	39	40	41	42	43	44	45
happier	the	time	is	right	to	make	new	friends

Figure 2: Example Text Database

- *Assumption B*
  The more solution extents contained in a document, the more likely that the document is relevant.

The first assumption provides a ranking for individual solutions extents; the second suggests a ranking technique for particular documents in terms of solution extents. Both assumptions are superficially reasonable, and preliminary trials with the TREC data appeared to bear out the assumptions. However, a problem remained in combining these assumptions to produce a single score for a document.

Assumption B suggests that a document might be ranked by summing individual scores of solution extents contained within it. A natural value to use as the score of a particular extent (p, q) is its length $|(p,q)| = (q - p + 1)$. Unfortunately this approach assigns a higher score to less relevant documents. Summing individual score is reasonable only if a higher score indicates a more relevant document. To rectify the problem we considered an inverse relationship

$$\text{Score of } (p, q) \propto \frac{1}{|(p,q)|}$$

or

$$\text{Score of } (p, q) = S(p, q) = \frac{A}{|(p,q)|}$$

During our preliminary trials it was then quickly observed that if the length of an extent was below a threshold of a dozen or so words, assumption A no longer appeared to hold and all extents appeared equally relevant — and certainly not varying at the level indicated by an inverse relationship. Therefore, we took the score for a particular extent as:

$$S(p, q) = \begin{cases} \frac{A}{|(p,q)|} & \text{if } |(p,q)| \geq A \\ 1 & \text{if } |(p,q)| \leq A \end{cases}$$

For any extent (p, q), we have $0 < S(p, q) \leq 1$. For our Trec experiments, a value of 16 was used for the constant A.

For our example database, arbitrarily taking A to be 2, we get scores $S(11, 12) = 1$, $S(12, 18) = 0.29$, $S(18, 19) = 1$, $S(19, 27) = 0.22$, $S(27, 28) = 1$, and $S(34, 36) = 0.67$ for solution extents to the query

```
"you" AND "will"
```

If solution extents (p_1, q_1)...,(p_N, q_N) are contained in a particular document, the score for the document is

$$\sum_{i=0}^{N} S(p_i, q_i)$$

In determining the score for each fortune in figure 1 we take only the solution extents contained entirely in a single fortune

```
("you" AND "will") contained in ("☺" ... "☺")
```

leaving extents $(11, 12)$, $(18, 19)$, $(27, 28)$ and $(34, 36)$. The score for the fourth fortune is highest ($S(27, 28) + S(34, 36) = 1 + 0.67 = 1.67$); the second and third fortunes both receive a score of 1; the first and fifth fortunes receive a score of 0.

5 TREC Experiment

5.1 Procedure

The MultiText project participated in both the routing task and the ad-hoc task. Queries were developed manually, the procedure differed only slightly for the two tasks. The queries were created manually by two of the investigators (Clarke and Cormack) working in conjunction. Approximately 15 to 45 minutes was spent developing a query for each topic. During creation of the routing queries relevant documents were sometimes pulled and used as a source of possible terms, but this practice was not uniformly followed. Besides the personal knowledge of the investigators, the only external resources used were an on-line dictionary (Webster's); the Unix spell program; an on-line list of country, state and city names and state postal abbreviations; and, in some few cases, current issues of newspapers.

The final query developed for each topic was a compound query consisting of an ordered list of one or more sub-queries (2.05 sub-queries on average). Results for each sub-query were determined separately using the ranking techniques described in the previous section. These results were then combined into a final solution set according to the ordering of the sub-query list, with results of a particular sub-query ranked before the results of subsequent sub-queries. Documents given a non-zero score by one sub-query were eliminated from the results of subsequent sub-queries before this final ranking.

This approach reflects a trade-off between a desire for precision and an artificial 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.

Figure 3 shows topic 246 and figure 4 gives our query in the internal format presented to the system and forwarded to NIST. In this format a terse syntax is used for operators, and phrases are expanded into term queries. Some explanation of the syntax is required: "^" and "+" are equivalent to "AND" and "OR" respectively, "<>" is equivalent to the ordering operator "...", the expression "[2]" is a query representing all two-word intervals in the text. The "@output" command sets the output file name for the query. The "@rank" command takes a topic number and a compound query as arguments and executes the ranking procedure. The topic number is used by the "@rank" command only for formatting the output. In this case the compound query list has only a single sub-query, which has been assigned the name "q".

The query of figure 4 is essentially a boolean expression in conjunctive normal form, consisting of three "facets", each built from several named pieces for convenience. The first facet ("arms") is a disjunction of terms and phrases related to military weapons. The second facet ("export") is a disjunction of terms related to trade. The final facet ("USbroad") is a disjunction of 150 geographical place names and abbreviations related to the United States. The definition of this last facet is not included in figure 4; its definition is global in scope and it is used whenever a topic concerns only the U.S.

Several other global definitions of this type were used in developing the queries and these definitions contributed significantly to the size of the queries. For the ad-hoc task, queries contained an average of 67 terms. For the routing task, queries contained an average of 53 terms. The variance was fairly high, for some topics the query consisted of hundreds of terms, for other topics the query

```
<top>

<num> Number:  246

<desc> Description:

What is the extent of U.S. arms exports?

</top>
```

Figure 3: Topic 246

```
@output "246.output"

arms0 = "arms" + "gun" + "guns" + "tanks"
arms1 = "firearm" + "firearms" + "weapon" + "weapons" + "rifle" + "rifles"
arms2 = (("fighter" <> ("jet" + "jets")) <  [2]) + "bomber" + "bombers"
arms = arms0 + arms1 + arms2

export0 = "export" + "exports" + "trade" + "sale" + "sales"
export1 = "tariff" + "tariffs"
export = export0 + export1

q = arms^export^USbroad

@rank 246 q
```

Figure 4: MultiText Query for Topic 246

consisted of a single two-term phrase. Overall, about half of the query terms resulted from the expansion of global definitions, overwhelmingly from the expansion of the "`USbroad`" definition. Expansion of phrases into terms and the manual construction of morphological term variants also contributed to the large number of terms per query.

5.2 Results and Discussion

Our results for the ad-hoc task are quite reasonable (average precision: 0.2994; R-precision: 0.3347). For over 65% of the topics our average precision is above the median average precision for all groups. Our results for the routing task are relatively poor (average precision: 0.1188; R-precision: 0.1649). In most cases our average precision is below the median average precision.

Given the similarity in methodology these results are surprising. After attempting to explain the difference, we discovered that the Ziff data from disk 3 had been omitted inadvertently from the routing run. In our final results, no Ziff documents were reported, and the overall recall results are commensurately lower.

The ranking technique is reasonably efficient. For the ad-hoc task, system search time required an average of 40 seconds per query (an average of 18 seconds per sub-query). For the routing task, system search time required an average of 10 seconds per query (an average of 5 seconds per sub-query). Nonetheless, the system is a research prototype and performance tuning and known algorithmic enhancements should reduce these numbers significantly. Since the results were submitted, some simple performance tuning, requiring less than an evening's work, have made the runs approximately 40% faster.

The scoring procedure described in section 4 produces scores that are independent of the characteristics of other documents in the collection — inverted document frequency, for example. This property allows a collection to be split arbitrarily into a number of sub-collections to be searched in parallel by separate search engines, with the results merged for the final ranking. This approach would produce essentially a linear speed-up in search time with the number of engines used.

6 Related Work

GCL owes some of its intellectual and cultural heritage to two earlier structured-text retrieval languages developed at the University of Waterloo. The language of Burkowski [1] is the direct ancestor of GCL, but primarily focused on structural queries over pre-defined hierarchical document components. The capabilities of that language are extended substantially by GCL.

The Pat text searching system [5, 11] was developed for use with the New Oxford English Dictionary. Queries expressible in Pat (with a few exceptions) are a subset of those expressible in GCL. Boolean queries in Pat must be solved with respect to a particular document component or fixed proximity, making the results of this paper inapplicable. Semantic limitations in Pat make the solving of certain types of structural queries, such as the earlier examples involving pairs of paragraphs, impossible.

Because of the prevalence of boolean queries in commercial text retrieval systems, the ranking of boolean queries has been the subject of extensive research. Fox et al. [6] provides an overview of several methods; a special issue of *Information Processing and Management* [12] has been devoted to the subject. In the previous TREC conference [7], Charoenkitkarn, Chignell and Golovchinsky [2] produced reasonable preformance using a simple boolean ranking technique along with highly interactive query development. For the same conference, Hawking and Thistlewaite [8] described experiments using the PADRE parallel free-text scanning system. The language used by the PADRE system is very similar to Pat. For ranking, they used a weighed sum based on term frequency and document length. In the current TREC conference, Hawking and Thistlewaite [9] continue their work with the PADRE system using a proximity scoring technique similar to ours.

7 Conclusions

For TREC-4, the MultiText project explored how the unique properties of our query language, GCL, could be exploited for document ranking purposes. Queries were primarily expressed in a boolean subset of the language, and the shortest substring property of the language was used as the basis for developing a scoring method. Based on our results, the techniques we developed appear to provide a simple, efficient and effective method for ranking GCL queries.

Since our ranking technique is relatively new, many aspects have not yet been explored. The scoring technique described in section 4 should be investigated in more depth. The sensitivity of ranking to changes in the parameter A is of interest. Other scoring formula could be developed. The query terms we used for the TREC experiments could be used with an entirely different ranking method to provide a more direct comparison than that provided by TREC, where the terms vary heavily from group to group.

For a future TREC conference we would concentrate our participation on the ad-hoc task and extend our participation to the interactive task. We are presently undertaking user interface and query construction research targeted toward the capabilities of GCL. This research should directly benefit our participation in an interactive task.

Acknowledgements

Rob Good provided system administration support and helpful feedback as the experimental work progressed. Bryan West, our Undergraduate Research Assistant, wrote a number of scripts to assist with data loading and result processing. We thank Sunshine Express for providing the example data used this paper.

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

References

[1] Forbes J. Burkowski. An algebra for hierarchically organized text-dominated databases. *Information Processing and Management*, 28(3):333–348, 1992.

[2] N. Charoenkitkarn, M. Chignell, and G. Golovchinsky. Interactive exploration as a formal text retrieval method: How well can interactivity compensate for unsophisticated retrieval

algorithms. In *Overview of the Third Text REtrieval Conference (TREC-3)*, pages 179–199, 1994.

[3] 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.

[4] 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*, Las Vegas, Nevada, April 1995.

[5] Heather Fawcett. *A Text Searching System — PAT 3.3 User's Guide*. Centre for the New Oxford English Dictionary, University of Waterloo, 1989.

[6] E. Fox, S. Betrabet, M. Koushik, and W. Lee. Extended boolean models. In William B. Frakes and Ricardo Baeza-Yates, editors, *Information Retrieval — Data Structures and Algorithms*, chapter 15, pages 393–418. Prentice Hall, Englewood Cliffs, NJ, 1992.

[7] D. K. Harman, editor. *Overview of the Third Text REtrieval Conference (TREC-3)*. National Institute of Standards and Technology, U. S. Department of Commerce, 1994. NIST Special Publication 500-225.

[8] David Hawking and Paul Thistlewaite. Searching for meaning with the help of a PADRE. In *Overview of the Third Text REtrieval Conference (TREC-3)*, pages 257–267, 1994.

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

[10] The MultiText Project. More information may be found in the project repository: `ftp://plg.uwaterloo.ca/pub/mt`.

[11] Airi Salminen and Frank Wm. Tompa. Pat expressions — An algebra for text search. *Acta Linguistica Hungarica*, 41:277–306, 1994. A version of this paper is available as: Technical Report OED-92-02, UW Centre for the New Oxford English Dictionary, Waterloo, Ontario, Canada, 1992.

[12] Tadeusz Radecki, editor. Special issue on the potential for improvements in commerical document retrieval systems. *Information Processing and Management*, 24(3), 1988.