PERGAMON Information Processing and Management 36 (2000) 291–311

INFORMATION PROCESSING & MANAGEMENT

www.elsevier.com/locate/infoproman

Relevance ranking for one to three term queries[☆]

Charles L.A. Clarke[a,*], Gordon V. Cormack[b], Elizabeth A. Tudhope[b]

[a]*Department of Electrical and Computer Engineering, University of Toronto, 10 King's College Road, Toronto, Ont., Canada M5S 3G4*

[b]*Computer Science, University of Waterloo, Waterloo, Ont., Canada N2L 3G1*

Received 10 July 1998; accepted 2 April 1999

Abstract

We investigate the application of a novel relevance ranking technique, *cover density ranking*, to the requirements of Web-based information retrieval, where a typical query consists of a few search terms and a typical result consists of a page indicating several potentially relevant documents. Traditional ranking methods for information retrieval, based on term and inverse document frequencies, have been found to work poorly in this context. Under the cover density measure, ranking is based on term proximity and cooccurrence. Experimental comparisons show performance that compares favorably with previous work. © 2000 Elsevier Science Ltd. All rights reserved.

1. Introduction

In interactive settings, queries to text retrieval systems are often extremely short, perhaps consisting of two or three terms, a short phrase or even a single word. The tendency toward very short queries appears to be particularly strong for Web-based retrieval systems. A trace of nearly 4000 queries taken from our MultiText Netnews searcher contained an average of 2.9 terms per query. The observations of others confirm our experience. Rose and Stevens (1996) report query length statistics collected from three Web-based information retrieval systems: THOMAS (Croft, Cook, & Wilder, 1995), Excite, and their own V-Twin system. For all three

☆ A previous version of this work was presented at the RIAO'97 conference in Montreal, Que. and included in the RIAO'97 proceedings (Clarke, Cormack, & Tudhope, 1997).

* Corresponding author.

E-mail addresses: clclarke@eecg.toronto.edu (C.L.A. Clarke), gvcormac@plg.uwaterloo.ca (G.V. Cormack), eatudhop@wheat.uwaterloo.ca (E.A. Tudhope).

systems, more than 85% of the queries consisted of three terms or less. In the query trace collected from one of the systems, involving more than 10,000 queries issued by more than 4000 users, 53% of the queries consisted of a single term. More recently, Jansen, Spink, Bateman and Saracevic (1998) studied a set of 51,473 queries collected from 18,113 users of the Excite Internet search system. They report an average query length of 2.35 terms, with 80% of the queries consisting of three terms or less.

In interactive retrieval environments, a predominance of extremely short queries is not surprising. Quickly issuing a short query provides immediate feedback, and an unsuccessful query can be easily reformatted and reissued.

Studies of user behavior in more traditional interactive retrieval environments do not always show the same extreme tendency for very short queries. For example, Koenemann and Belkin (1996) report an average length of approximately six terms for queries developed interactively on their baseline system. However, users in that study were provided with a training tutorial, were presented with a relatively elaborate interface and were specifically instructed to develop routing queries, queries that could be used to filter future additions to the corpus. In other words, the result of a search session was a query, not a list of relevant documents.

In contrast, the user of a Web-based retrieval system is usually engaged in an information discovery task, seeking the answer to a specific question or the location of a single relevant document. The user interface of a typical Web-based retrieval system often consists of little more than a label ('Search Me:') and tiny text box in which to type. This form of interface is not unreasonable, given that Web-based retrieval systems cannot provide user training, and as a result cannot depend on structured queries, detailed natural language descriptions or elaborate user interfaces.

Development of effective ranking techniques for short queries can be challenging. For example, Wilkinson, Zobel and Sacks-Davis (1995) report that users are often dissatisfied with standard similarity measures when applied to very short queries. They studied queries of 2–10 terms in length and concluded that the relative performance of similarity measures can vary considerably with average query length, noting particularly that the standard cosine measure performs poorly for very short queries.

Both Rose and Stevens (1996) and Wilkinson et al. (1995) report that for very short queries users generally expect a document containing most or all of the query terms to be ranked before a document containing fewer terms, regardless of the frequency of term occurrence. Indeed, Wilkinson et al. (1995) report that this preference apparently holds even when a document with fewer terms is relevant but a document with more terms is not.

As a result of their experiences, both groups identify coordination level, the number of distinct query terms contained in the document, as being of major importance for ranking the results of very short queries. Both groups support their anecdotal reports of user preferences by showing superior retrieval effectiveness for similarity measures that emphasize coordination level when queries are short. Wilkinson et al. (1995) studied a number of similarity measures and report the best performance for a variant of the Okapi measure (Roberston & Walker, 1994), which they regard as a form of coordination level measure. Rose and Stevens introduced a measure that blends the cosine measure together with coordination level, giving more weight to the cosine measure as the query length is increased, and showed an improvement over an unblended cosine measure for short queries.

The relatively poor performance of the traditional cosine measure and the relative importance of coordination level suggests that a direct focus on modifications to coordination level ranking may prove more successful than adaptations of standard similarity measures. For queries of two or three terms, coordination level is likely to be of paramount importance, but the small number of terms makes it likely that many documents will match at the highest coordination level. For queries of a single term, ranking by coordination level is meaningless. Our solution is to use coordination level as the basis for a primary or first-level ranking and to focus further attention on ranking within coordination levels.

In this paper, we introduce a novel ranking method for very short queries, those of one to three terms in length, that augments coordination ranking. The ranking method is based on an unusual measure of term cooccurrence, which we call cover density.

2. Cover density ranking

Documents containing one or more query terms are first ranked by coordination level. The greater the number of distinct query terms contained in a document, the higher it is ranked. The documents are thus grouped into sets according to the number of distinct query terms each contains, with the initial ranking of a document based on the set in which it appears. Documents that do not contain any of the query terms are not considered by the ranking procedure and are not reported as part of the query result set. A secondary ranking procedure is applied to the set of documents at each coordination level to produce the overall ranking.

Given a query Q, which we treat as an unordered set of terms, the initial ranking produces $|Q|$ document sets, $D_1, \ldots, D_{|Q|}$, where

$$d \in D_i \Longrightarrow |\{t | t \in d \text{ and } t \in Q\}| = i.$$

Documents in D_i are ranked ahead of the documents in D_j if $i > j$. Ranking of documents within a coordination level will be based on the proximity and density of query terms within the documents.

Each document $d \in D_i$ is treated as an ordered sequence of terms

$$t_1, t_2, \ldots, t_{|d|}.$$

An extent over the document is an ordered paired (p, q), with $1 \le p \le q \le |d|$, specifying the interval of text beginning at t_p and continuing to t_q.

Since $d \in D_i$, the document will contain a subset of the query terms $T \subseteq Q$, with $|T| = i$. An extent (p, q) *satisfies* a term set T if all of the terms in T appear in the interval of text specified by the extent. Let $\mathscr{E}$ be the set of all extents over d satisfying the term set T:

$$\mathscr{E} = \{(p, q) | 1 \le p \le q \le |d| \text{ and } T \bigcap \{t_p, \ldots, t_q\} = T\}.$$

Included in the set $\mathscr{E}$ is the extent $(1, |d|)$, representing the entire document. An extent (p, q) is a *cover* for T if and only if it satisfies T and it does not contain a shorter extent satisfying T, that is, if there does not exist an extent (p', q') over d satisfying T such that $p < p' \le q' \le q$ or $p \le p' \le q' < q$. Finally, we define the set $\mathscr{C}$ as the set of all covers for T in d. The cover set

$\mathscr{C}$ is a compact representation of the set of all extents that satisfy T — if an extent satisfies T then it is either an element of $\mathscr{C}$ or it contains an element of $\mathscr{C}$. Our ranking formula will be based on the length and number of covers in $\mathscr{C}$. Cover sets can be efficiently generated; details are given in the appendix.

To illustrate the concept of a cover set we will use the short document (Pratt, 1989):

Erosion^{1}

It^{2} took^{3} the^{4} sea^{5} a^{6} thousand^{7} years,^{8}
A^{9} thousand^{10} years^{11} to^{12} trace^{13}
The^{14} granite^{15} features^{16} of^{17} this^{18} cliff,^{19}
In^{20} crag^{21} and^{22} scarp^{23} and^{24} base.^{25}

It^{26} took^{27} the^{28} sea^{29} an^{30} hour^{31} one^{32} night,^{33}
An^{34} hour^{35} of^{36} storm^{37} to^{38} place^{39}
The^{40} sculpture^{41} of^{42} these^{43} granite^{44} seams,^{45}
Upon^{46} a^{47} woman^{48}'s^{49} face.^{50}

—*E.*^{51} *J.*^{52} *Pratt*^{53} *(1882*^{54}*–1964)*^{55}

Superscripts indicate term positions. The term set

$$T' = \{\text{“}sea\text{”}, \text{“}thousand\text{”}, \text{“}years\text{”}\}$$

has the cover set

$$\mathscr{C}' = \{(5, 8), (10, 29)\}.$$

The extents (5, 11), (8, 29) and (1, 55) all satisfy T', but are not included in the cover set since they contain shorter extents that satisfy T'. Similarly, the term set

$$T'' = \{\text{“}granite\text{”}, \text{“}sea\text{”}\}$$

has the cover set

$$\mathscr{C}'' = \{(5, 15), (15, 29), (29, 44)\};$$

and the term set

$$T''' = \{\text{“}sea\text{”}\}$$

has the cover set

$$\mathscr{C}''' = \{(5, 5), (29, 29)\}.$$

Scoring of cover sets is based on two assumptions: (1) the shorter the cover, the more likely the corresponding text is relevant; and (2) the more covers contained in a document, the more likely the document is relevant. The first assumption suggests that a score for an individual extent might be based on its length. The second assumption suggests that a document may be scored by summing the individual scores of the covers in its cover set. Combining these ideas, we score the cover set $\mathscr{C} = \{(p_1, q_1), (p_2, q_2), \dots, (p_n, q_n)\}$ using the formula

$$S(\mathscr{C}) = \sum_{j=1}^{n} I(p_j, q_j), \tag{1}$$

where

$$I(p, q) = \begin{cases} \dfrac{\mathscr{K}}{q-p+1} & \text{if } q-p+1 > \mathscr{K}, \\ 1 & \text{otherwise.} \end{cases} \tag{2}$$

Covers of length $\mathscr{K}$ or shorter are assigned a score of 1, and longer covers are assigned scores < 1 in proportion to the inverse of their length. If $\mathscr{K} = 4$ then the example cover sets for *Erosion* have scores $S(\mathscr{C}') = 1.20$, $S(\mathscr{C}'') \approx 0.88$ and $S(\mathscr{C}''') = 2.00$. For the experiments reported in the remainder of this paper we use $\mathscr{K} = 16$, which has been shown to produce good results for a related measure applied to the results of Boolean queries (Clarke, Cormack, & Burkowski, 1995).

Cover sets have properties that can be exploited to permit their efficient computation. In a cover set, no two covers can start at the same position, since one would then be contained in the other. As a result, the covers are totally ordered by their start positions. Similarly, no two covers can end at the same position and are totally ordered by their end positions. The total orders are the same. Otherwise, there would exist covers (p, q) and (p', q') such that $p > p'$ and $q < q'$ and (p, q) would be contained in (p', q').

It is not the frequency of individual query terms that is measured by cover density, but the frequency and proximity of their cooccurrence. It is conceivable that a document could contain many occurrences of all the query terms but contain only a single cover, with the query terms clustered into discrete groups. In some sense, a term cover extends the notion of coordination level below the granularity of a document, down to the word level.

A similar method has been successfully used to rank the results of Boolean queries in our previous work (Clarke et al., 1995; Clarke & Cormack, 1996) and in independent work by researchers at the Australia National University (Hawking & Thistlewaite, 1995, 1996; Hawking, Thistlewaite, & Bailey, 1996). The work described in this paper represents our first attempt to extend the method to unstructured queries. Other related work includes that of Keen (1991, 1992a, 1992b), who evaluated the benefits of proximity operators in Boolean systems and proposed several non-Boolean ranking methods based on term proximity within sentences.

Cover density ranking can be efficiently implemented. An appendix provides the details.

3. Evaluation

In this section, we evaluate our ranking procedure using queries of fixed lengths from one to three terms to determine what improvement, if any, a secondary ranking by cover density makes over ranking by coordination level alone.

The evaluation is based on the test collection developed by the US National Institute of Standards and Technology (NIST) as part of the TREC series of conferences (Harman, 1993; Voorhees & Harman, 1997). The collection presently consists of approximately 5 gigabytes of text, distributed to conference participants on five CDROM disks. The collection includes large subcollections of newspaper articles from several different sources, along with US patents and issues of the US Federal Register. Associated with the collection is a set of 350 descriptions of user requirements, called *topics*, from which queries are developed. For each topic, relevance judgements are available for large portions of the text collection. Recent conference proceedings should be consulted for a more detailed overview of the TREC effort (Harman, 1995; Voorhees & Harman, 1996, 1997).

For our initial evaluation of the ranking technique, we selected a subset of the collection consisting of *AP Newswire* articles from 1988, articles taken from the *Wall Street Journal* over the period 1990–1992 and issues of the 1988 *Federal Register* (the AP, WSJ and FR documents from disk 2). In total, the subcollection contained 699 megabytes of text divided into 174,299 documents. As the basis for our queries, we selected 100 of the topics for which relevance judgements were available over this portion of the collection (topics 101–200). Since these topics are intended to be descriptions of user requirements they are quite long, containing an average of 145 terms including stopwords and are structured into several fields, including a title, a short description, a longer narrative section and in some cases, a list of concepts.

For each of these topics we created a one term query, a two term query and a three term query by selecting terms from the topic text. Using these queries we ranked the documents using three different variants on ranking within coordination levels (see Table 1). The first variant uses coordination level alone. Within a coordination level documents are ordered only by their position in the database. The second variant (labeled 'CL > TF') orders documents at each coordination level according to a count of the number of times a query term appears in the document. The term frequencies of each query term are added together to produce a

Table 1
One to three terms — precision at five document retrieval levels averaged over 100 queries at each of three query lengths, executed over a selected subset of the TREC test collection

Query Terms	Ranking method	Number of documents retrieved				
		5	10	15	20	100
1	coordination level only	0.040	0.035	0.034	0.034	0.033
	CL > CD	0.170	0.156	0.158	0.154	0.126
2	coordination level only	0.080	0.080	0.073	0.075	0.076
	CL > TF	0.168	0.195	0.208	0.213	0.157
	CL > CD	0.336	0.325	0.309	0.294	0.183
3	coordination level only	0.132	0.137	0.132	0.138	0.108
	CL > TF	0.180	0.199	0.213	0.214	0.164
	CL > CD	0.360	0.347	0.331	0.312	0.193

document score, with higher scoring documents given higher ranks. We consider this second variant to be intermediate between pure coordination level ranking and the cover density measure described in the previous section. The third variant (labeled 'CL > CD') uses the cover density measure. For all runs, the query and document terms were stemmed using the Porter stemmer (Frakes, 1992; Porter, 1980).

The table reports precision at five different document retrieval levels averaged over the 100 topics. We consider the results at 5 and 10 documents to be of the greatest interest since the user of an interactive system should rarely be expected to view a large number of documents in response to a very short query. Note that for single term queries, the second and third variants are equivalent and figures for the second variant are not reported. In all cases, cover density ranking improves performance substantially.

4. Comparison with other methods

This section compares cover density ranking with standard ranking methods.

Wilkinson et al. (1995) evaluated the performance of a number of similarity measures using a set of 2–8 term queries based on 49 topics taken from the TREC test collection (topics 202–250). These topics are somewhat shorter than those used in the previous section, averaging 19 terms in length, including stopwords. The queries were created by manually selecting terms from the topics and have an average length of 3.7 terms. The queries were executed over the collection of *Wall Street Journal* articles used in the evaluation of the previous section (WSJ articles from disk 2).

We obtained the queries used by Wilkinson et al. and executed them against the same text collection using ranking both by coordination level alone and by coordination level and cover density. The results appear in Table 2. The last four columns of the table are reproduced from Wilkinson et al. (1995). In the remainder of this section we refer to these queries as 'query set A'.

The method labeled 'CDM' is a ranking by coordination level only. In principle, this column should be identical with the 'CL Only' column. The slight differences may be related to the ordering of documents within coordination levels, or to stemming and other lexical factors. The method labeled 'WCM' uses a weighted coordination level ranking, with the weights based on inverse document frequency. The label 'COS' represents the cosine measure, and the label 'OKA' represents a variant of the Okapi measure (Robertson & Walker, 1994).

The table reports interpolated precision at eleven recall levels and (noninterpolated) average precision, computed over the top 1000 documents returned for each query and averaged over the 49 queries. Interpolated precision values are calculated as the maximum precision achieved at any recall level greater than or equal to the specified recall level.

With an average length of 3.7 terms and a maximum length of 8 terms, the queries used by Wilkinson et al. for their study are still quite long. We created a second set of shorter queries ('query set B') for the same topics, following the same procedure but selecting only one to three terms from the topic statements. The queries were created independently of the queries created by Wilkinson et al., but were subsets of those queries in all but six cases. The queries

Table 2
Two to eight terms — interpolated precision at 11 recall levels and average precision for 49 short queries (query set A) evaluated over 90,257 *Wall Street Journal* articles taken from the TREC test collection. The queries and the results in the last four columns are reproduced from Wilkinson et al. (1995)

Recall level (%)	Method					
	CL only	CL > CD	CDM	WCM	COS	OKA
0	0.343	0.504	0.374	0.386	0.226	0.497
10	0.258	0.406	0.242	0.244	0.177	0.337
20	0.223	0.322	0.178	0.182	0.127	0.242
30	0.173	0.257	0.132	0.143	0.089	0.196
40	0.129	0.210	0.102	0.132	0.068	0.146
50	0.087	0.152	0.087	0.101	0.055	0.116
60	0.074	0.128	0.074	0.094	0.044	0.094
70	0.041	0.103	0.020	0.032	0.013	0.033
80	0.031	0.078	0.020	0.032	0.013	0.033
90	0.014	0.046	0.008	0.016	0.008	0.019
100	0.009	0.038	0.006	0.011	0.004	0.013
Average precision	0.111	0.184	0.096	0.107	0.066	0.139

were executed over the same collection of *Wall Street Journal* articles. The results appear in Table 3.

Comparing these results with the equivalent values in Table 2, we see that use of the shorter queries produces an apparent degradation in retrieval effectiveness when ranking is by

Table 3
One to three terms — interpolated precision at 11 recall levels and average precision for 49 very short queries (query set B) evaluated over 90,257 *Wall Street Journal* articles taken from the TREC test collection. The results may be compared with the corresponding columns of Table 2

Recall level (%)	Method	
	CL Only	CL > CD
0	0.280	0.556
10	0.227	0.465
20	0.190	0.383
30	0.149	0.311
40	0.121	0.269
50	0.088	0.218
60	0.075	0.181
70	0.055	0.140
80	0.036	0.102
90	0.022	0.068
100	0.014	0.045
Average precision	0.097	0.230

coordination level alone, but produces an apparent improvement in retrieval effectiveness when a secondary ranking by cover density is applied.

One of the main experimental tasks (the 'adhoc' task) of the TREC-4 conference (Harman, 1995) worked with the TREC topics (202–250) used for our comparison. Groups participating in this task executed queries derived (either manually or automatically) from these topics over a subset consisting of approximately half of the TREC text collection (disks 2 and 3). This subset consists of 2 gigabytes of text comprising 567,529 documents.

Using our ranking method, we executed both short query sets over this subset. The results appear as the second and third columns ('CL > CD') in Table 4.

The last two columns of the table are reproduced from the TREC-4 proceedings (Harman, 1995). The column labeled 'uwgcl1' reproduces the official TREC-4 run submitted by our research group (Clarke et al., 1995). The queries used for this run are manually constructed and are quite large, with an average of 67 terms each. Manual construction of each of these queries required approximately 15–45 min and drew upon the general knowledge of their creators for term expansion purposes. In contrast, the construction of query set B required less than 10 min in total (an average of less than 13 s per query) and used only the terms existing in the topics.

The last column of the table, labeled 'CITRI1' reproduces one of two official TREC-4 runs submitted by Wilkinson et al. (1995). Here, the queries were constructed automatically from the topic text by deleting stopwords, and comparisons with the other runs in the table should be viewed in this light. We include this run since its ranking is based on the same variant of the Okapi measure used for the last run in the Table 2 ('OKA').

While the performance of the short query runs is inferior to the performance reported in the

Table 4
TREC-4 comparison — interpolated precision at 11 recall levels and average precision for runs over the TREC-4 'adhoc' collection. The first run uses the short, two to eight term, queries from Wilkinson et al. (1995)(query set A). The second run uses the very short, one to three terms, queries developed for this paper using the same topics (query set B). The last two runs are reproduced from the TREC-4 proceedings (Harman, 1995) and are intended to provide context for the first two runs. The CITRI1 run is one of two official runs submitted by Wilkinson et al. (1995) and is based on the same Okapi variant (OKA) as the results reported in the last column of Table 2

Recall level (%)	CL > CD (A)	CL > CD (B)	uwgcl1	CITRI1
0	0.626	0.656	0.724	0.595
10	0.369	0.420	0.573	0.384
20	0.283	0.339	0.509	0.320
30	0.206	0.285	0.414	0.261
40	0.161	0.248	0.350	0.213
50	0.109	0.190	0.297	0.171
60	0.080	0.145	0.236	0.119
70	0.054	0.095	0.181	0.070
80	0.035	0.053	0.118	0.031
90	0.016	0.025	0.070	0.010
100	0.006	0.001	0.011	0.003
Average precision	0.150	0.198	0.299	0.178

Table 5
TREC-4 comparison — precision at five document retrieval levels for the runs in Table 4

Run	Number of documents retrieved				
	5	10	15	20	100
CL > CD (A)	0.425	0.380	0.347	0.317	0.196
CL > CD (B)	0.437	0.410	0.395	0.366	0.232
uwgcl1	0.551	0.504	0.487	0.477	0.341
CITRI1	0.380	0.376	0.342	0.317	0.207

uwgcl1 run, the difference in user effort represented by the two runs is extreme, differing by a factor of at least 100. Table 5 reports precision at five document retrieval levels for these same runs. Over the first 10 documents the difference in precision for short query set B and uwgcl1 is 0.094, an average difference of less than a single document.

Since queries were generated automatically from the topic statements, query creation for the 'CITRI1' run required no user effort at all, except for the effort required to formulate the original topic, which has been discounted in all cases.

5. TREC-6 comparison

The experiments described in Sections 3 and 4 were conducted in early 1997 and were reported at the Fifth RIAO conference in June 1997 (Clarke et al., 1997). While the results of these experiments demonstrate the potential of cover density ranking, they should be viewed with some caution, since the very short queries used in the experiments were created manually by the individuals conducting the research. The Sixth Text Retrieval Conference (TREC-6), held in November 1997, provided an opportunity to test cover density ranking using an independently generated query set (Voorhees & Harman, 1997). The *title* fields of the new topics created by NIST for TREC-6 (topics 301–350) were specifically intended to be used as very short queries. Participants could submit experimental runs using the topic titles only, with the results of these runs reported separately in the conference proceedings. The MultiText project participated in many of the experimental tasks and tracks associated with TREC-6; the details are given our TREC-6 paper (Cormack, Clarke, Palmer, & To, 1997). Our official TREC-6 experimental run for the very short, title-only queries extended cover density ranking with automatic query expansion and combined the results with a ranking based on a variant of the Okapi measure. In addition to our official run, we performed several unofficial runs to validate cover density ranking; this section summarizes the results.

After stopword elimination, the TREC-6 very short queries are ideally suited to cover density ranking. Of the 50 queries, 45 consist of two or three terms. Two consist of four terms; three consist of a single term. The average length is 2.48 terms.

As part of our TREC-6 work, we implemented a version of the Okapi measure as an extension of the MultiText system. For a document D and a query Q we compute

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

where: N is the number of documents in the database, n_t the number of documents containing term t, $f_{D,t}$ the frequency of occurrence of term t in document D, l_D the length of document D and l_{avg} the average document length.

This formula is equivalent to the standard Okapi BM11 measure (Robertson & Walker, 1994) with reasonable choices for the parameter values ($k_1=1$ and $k_2=k_3=R=r=0$) and is similar to the version of the Okapi measure used by Wilkinson et al. (1995) for their TREC-4 experiments.

The TREC-6 adhoc collection consists of 2 gigabytes of text comprising 556,077 documents. We executed the TREC-6 very short queries over this collection using three different ranking methods, including the MultiText version of the Okapi measure. The results are shown as the first three rows of Table 6.

The last two rows are reproduced from the TREC-6 proceedings and are included for context. Most of the groups submitting official TREC-6 very short query runs, including MultiText, augmented their basic ranking method with automatic query expansion and other techniques. While these techniques can be computationally expensive, they can often increase retrieval performance by 10–20%. In Table 6, the row labeled 'uwmt6a1' reproduces the official TREC-6 run submitted by our research group and shows the effects of our query expansion and evidence combination techniques. This run had the fourth-highest average precision of the officially submitted TREC-6 very short query runs. The row labeled 'city6at' reproduces the run submitted by Walker, Robertson, Boughanem, Jones and Jones (1997), the official run with the best average precision. This run incorporates automatic query expansion and passage weighting techniques.

6. Lexical issues

For very short queries, lexical issues related to stemming, capitalization, stopwords and

Table 6
TREC-6 comparison — precision at five document retrieval levels for runs over the TREC-6 'adhoc' collection using the TREC-6 very short, title-only, queries. The last two runs, reproduced from the TREC-6 proceedings (Voorhees and Harman, 1997) incorporate query expansion and are intended to provide context

Ranking method	Number of documents retrieved				
	5	10	15	20	100
CL only	0.240	0.204	0.192	0.193	0.137
CL > CD	0.456	0.402	0.360	0.320	0.180
MultiText Okapi	0.400	0.386	0.348	0.329	0.178
uwmt6a1	0.464	0.422	0.365	0.337	0.193
City6at	0.480	0.438	0.393	0.367	0.220

phrases are of increased importance. Illustrative examples of the problems arising from the treatment of lexical issues are numerous and legendary: 'The Who' vs. 'The WHO', 'start' vs. 'START', 'oriental' vs. 'orienteering', 'The Limited', 'to be or not to be that is the...'

When queries are long, redundancy within the query should mitigate the effects of choices made at the lexical level. A common practice has evolved for handling these issues: punctuation is ignored, characters are converted to a single case, stopwords are stripped from the text and the remaining terms are stemmed. This practice optimizations factors such as index size, execution speed, and overall recall in the face of indifferent effects on retrieval effectiveness.

Stemming, in particular, represents a profound change at the lexical level, but has not been shown to consistently cause either benefit or harm to retrieval effectiveness (Harman, 1991; Tudhope, 1996). As part of a larger study on stemming Harman (1991) investigated the impact of restricting stemming to short queries (those with <10 terms) and concluded that query length does not predict whether stemming will improve query performance. We were not confident that this conclusion could be extended to queries of one to three terms. Given the lack of redundancy in very short queries, we expected that stemming would have a negative influence on retrieval effectiveness, particularly for single term queries.

We repeated the very short query runs from Section 3 using cover density ranking and unstemmed terms. The results appear in Table 7.

Although slightly better performance is seen at low document retrieval levels for the unstemmed single term queries, these results cannot be interpreted as confirming our expectations. Examining the results for individual queries reveals that for many queries retrieval performance is essentially unaffected by stemming. Stemming helps other queries and harms a similar number of queries. In view of this behavioral mixture, it is possible that blending the scores for stemmed and unstemmed queries based on cover density measures between morphological variants would result in a balance between the two extremes. Our ongoing work on this approach is partially inspired by the related research of Church (1995).

Given the discussion in the introduction, the results of Table 7 may be interpreted slightly differently. The reported user preference for coordination level ranking of very short queries

Table 7
Stemmed vs. unstemmed queries — precision at five document retrieval levels averaged over 100 queries at each of three query lengths, executed over a selected subset of the TREC test collection. In all cases ranking is by coordination level followed by cover density (CL > CD). Values for stemmed terms are reproduced from Table 1

Query terms	Stemming	Number of documents retrieved				
		5	10	15	20	100
1	stemmed	0.170	0.156	0.158	0.154	0.126
	unstemmed	0.182	0.163	0.161	0.163	0.124
2	stemmed	0.336	0.325	0.309	0.294	0.183
	unstemmed	0.344	0.314	0.307	0.289	0.176
3	stemmed	0.360	0.347	0.331	0.312	0.193
	unstemmed	0.362	0.324	0.306	0.290	0.189

may be generalized to a preference for documents that accurately match the user's query. Since no major impact is seen on retrieval performance, at least in the case of stemming, no harm is caused by catering to this (assumed) preference.

This policy may be extended to other lexical issues. For our TREC experiments (Clarke & Cormack, 1996; Clarke et al., 1995) we have found it useful to provide special handling for tokens consisting entirely of upper-case characters. By default these tokens are mapped to lower case for indexing, but in addition they are indexed in their original upper-case forms. This special handling preserves the distinction between 'salt'/'SALT', and 'pal'/'PAL'. If a query term appears entirely in upper case it is satisfied only by the uppercase form of the term.

Our experience indicates that very short queries are often phrases or proper names. 'Bill Gates' is an typical example of a query better treated as a phrase than as a pair of terms. Since term proximity is a major factor contributing to cover density, phrases are generally handled correctly. However, we anticipate that special consideration for phrases could be integrated into the method, perhaps by favorably weighing term covers in which all or part of the query appears as a phrase. Some of the special lexical issues related to proper names have been studied by Gross (1991) and Pfeifer, Poersch and Fuhr (1996). The importance of stopwords in phrases also cannot be ignored: a 'bill for gates' is very different from 'Bill Gates'.

The treatment of lexical issues for very short queries is a focus of our current research.

7. Concluding discussion

We do not believe that a primary ranking based on coordination level is necessarily suited to queries longer than three terms. However, given the frequency with which very short queries are submitted to Web-based retrieval engines, these queries are reasonably treated as a special case. Extension of cover density ranking to longer queries could be effected by a gradual blending with a more traditional similarity measure as query length is increased (Rose & Stevens, 1996).

Cover density ranking is fast enough for interactive use. Our system executed the TREC-6 very short queries over the 2.1GB TREC-6 adhoc collection at an average rate of 0.7 s per query, reporting the top 20 solutions in each case. Reporting the top 1000 solutions required an average of 2.0 s per query. The machine used for these performance measurements ran Linux on a Cyrix P200+ (Pentium clone) processor, with 64 MB of memory and two 4.1 GB EIDE drives, and cost less than US$2000 in mid-1997.

Cover density ranking has several attractive properties in addition to being an efficient and effective method of relevance ranking for one to three term queries. As a form of triage, text passages corresponding to high-scoring covers may be presented to users in lieu of full documents, potentially reducing the time required to make relevance judgements or to determine which documents warrant full examination. These passages may also be used as a source of terms for automatic or interactive query expansion. Exploration of these ideas has been undertaken as part of our TREC-6 and TREC-7 work (Cormack et al., 1997; Cormack, Palmer, Van Biesbrouck, & Clarke, 1998). Apart from few exceptions (Kaszkiel & Zobel, 1997; Knaus, Mittendorf, Schäuble, & Sheridan, 1995), most passage retrieval methods divide a document into passages at the time it is added to the database (Allan, 1995; Callan, 1994;

Hearst & Plaunt, 1993; Salton, Allan, & Buckley, 1993; Wilkinson, 1994; Wilkinson & Zobel, 1994;). As a result, this division may not be appropriate to the query at hand. In contrast, passages identified through cover density ranking are based on the query itself.

An unusual feature of cover density ranking is its dependence on term frequency and proximity. Collection-wide statistics, such as inverse document frequency and average document length, are not used. Since the scoring formula does not use collection-wide statistics, cover density ranking is particularly suited to distributed and parallel environments. Collections may be partitioned in any suitable way. During query processing, documents are scored independently and the results merged as a final step.

In the interests of promoting experimental comparison, the queries developed for this paper are publically available at the MultiText repository site: ftp://plg.uwaterloo.ca/pub/mt/shortq.

Acknowledgements

We are grateful to Ross Wilkinson of the Royal Melbourne Institute of Technology for providing us with the short queries used in their TREC-4 experiments. This research received its primary funding from Communications and Information Technology Ontario (CITO). Additional funding was provided by the Natural Sciences and Engineering Research Council of Canada and by the University of Toronto.

Appendix A. Implementation issues

Implementation issues are ignored in the high-level description of Section 2. This appendix addresses these issues, detailing an efficient algorithm for cover density ranking. The description proceeds top down. At the lowest level, the algorithm is founded on simple inverted-list file structures that record term positions within documents.

A.1. Combining coordination levels

For many applications, a limit (r) on the number of documents to be returned by the ranking procedure can be predefined. The job of the ranking procedure is then to return the top r documents in ranked order. For Web-based retrieval systems this limit is typically 10–20. For experiments conducted as part of the TREC conference series the limit is 1000, a limit that would be sufficient for most purposes. The algorithm takes advantage of this limit to improve its efficiency.

Given a query Q, containing $|Q|$ terms, the cover density ranking algorithm first generates the document set $D_{|Q|}$ in ranked order. If the size of this document set $|D_{|Q|}|$ is greater than or equal to the predefined limit r on the number of ranked documents to be generated, the algorithm returns the top r documents from $D_{|Q|}$ and terminates. If $|D_{|Q|}|$ is less than r, the algorithm proceeds to generate $D_{|Q|-1}$, $D_{|Q|-2}$, etc. until r ranked documents have been generated or until D_1 has been generated, at which point all documents that contain at least one of the query terms have been generated and ranked.

Each of the document sets D_i, $1 \leq i \leq |Q|$, consists of those documents that contain exactly i

terms from Q. The algorithm generates these document sets from document sets X_i, $1 \leq i \leq |Q|$, where each X_i consists of those documents that contain i or more terms from Q. That is,

$$X_i = \bigcup_{i \leq j \leq |Q|} D_j.$$

Starting with $D_{|Q|}$, the document sets D_i, $1 \leq i \leq |Q|$, can be generated recursively as follows:

$$D_{|Q|} = X_{|Q|}$$

$$D_i = X_i - X_{i+1} = X_i - \bigcup_{i<j \leq |Q|} D_j, 1 \leq i < |Q|.$$

These equations generate the document sets D_i, $1 \leq i \leq |Q|$, in the order required by the cover density ranking algorithm. If D_i is required by the algorithm, then $D_{i+1}, \ldots, D|Q|$ have already been generated and X_{i+1} is available; X_i can be generated and filtered against X_{i+1} to produce D_i. The algorithm for generating the document set X_i will be presented shortly.

The algorithm for generating X_i also scores and ranks the documents in X_i. The scoring method is an extension of the scoring method described in Section 2. Given a query Q and a document d, treated as an ordered sequence of terms $t_1, \ldots, t_{|d|}$, an extent (p, q) is said to *i-satisfy* Q if the associated text interval contains exactly i distinct terms from Q. That is,

$$|\{t| \in \{t_p, \ldots, t_q\} \text{ and } t \in Q\}| = i.$$

An extent (p, q) is an *i-cover* for Q if and only if it *i*-satisfies Q and does not contain a shorter extent that *i*-satisfies Q. As before, if $\mathscr{C}_i = \{(p_1, q_1), (p_2, q_2), \ldots (p_n, q_n)\}$ is the set of all *i*-covers for Q in d, we score the document using Eqs. (1) and (2) from Section 2, assigning the score $S(C_i)$. The query

$$Q = \{\text{``sea''}, \text{``thousand''}, \text{``years''}\}$$

has the 2-cover set

$$\mathscr{C}_2 = \{(5, 7), (7, 8), (8, 10), (10, 11), (11, 29)\}.$$

over the poem *Erosion* from Section 2 and has score $S(C_2) = 4.21$ if $\mathscr{K} = 4$.

The score for a document $d \in X_i$ is based on its set of *i*-covers and is computed using Eqs. (1) and (2) from Section 2. If d is also an element of D_i, its set of *i*-covers is the same as the set of covers for the terms from Q appearing in d, since d contains exactly i terms from Q, and the scores are identical. As a consequence, the ranking of the documents from X_i using *i*-covers is consistent with the ranking required for D_i. The result of filtering the ranked documents from X_i against the documents in X_{i+1} is a ranked D_i.

The details of the algorithm are presented in Fig. 1. The *CoverDensityRank* procedure takes as its arguments a set of query terms Q and a limit on the number of documents to be returned r. The procedure constructs an indexed set of documents $\mathscr{Z}$, and returns this set as its

```
$CoverDensityRank(Q, r) \equiv$
    $\mathcal{Z} \leftarrow \emptyset$;
    $documents \leftarrow 0$;
    $i \leftarrow |Q|$;
    while $i \geq 1$ do
        $\mathcal{Y} \leftarrow LevelRank(Q, i)$;
        for $j \leftarrow 1$ to $|\mathcal{Y}|$ do
            if $\mathcal{Y}[j] \notin \mathcal{Z}$ then
                $documents \leftarrow documents + 1$;
                $\mathcal{Z}[documents] \leftarrow \mathcal{Y}[j]$;
                if $documents = r$ then
                    return $\mathcal{Z}$;
                end if;
            end if;
        end for;
        $i \leftarrow i - 1$;
    end while;
    return $\mathcal{Z}$;
```

Fig. 1. Cover density ranking procedure — given a set of query terms Q and a limit r on the number of documents to be returned, the procedure performs cover density ranking and returns an indexed set of ranked documents. The *LevelRank* procedure of Fig. 2 is called to generate and rank the document sets Xi, $1 \leq i \leq |Q|$.

result. The set $\mathcal{Z}$ is indexed by rank; $\mathcal{Z}[i]$ is the document at rank i; $|\mathcal{Z}|$ is the size of $\mathcal{Z}$. Like $\mathcal{Z}$, $\mathcal{Y}$ is an indexed set of ranked documents; $\mathcal{Y}$ is used to hold X_i for various values of i. For clarity, the integer variable *documents* is used to maintain the current size of the result set ($|\mathcal{Z}|$); the integer variables i and j are used as counters and indices.

After initialization (lines 1–3), each iteration of the loop at lines 4–16 generates and processes a document set X_i, from $X_{|Q|}$ down to X_1. At the start of each iteration, $\mathcal{Z}$ contains the documents in X_{i+1}. At line 5, the *LevelRank* procedure, detailed in Fig. 2, is called to generate the ranked documents in X_i, which are stored in $\mathcal{Y}$ for processing. The loop at lines 6–14 processes each document in $\mathcal{Y}$ in ranked order. Documents not already appearing in $\mathcal{Z}$ are added to $\mathcal{Z}$ (lines 8–9). Once $\mathcal{Z}$ contains r documents, the algorithm is terminated and $\mathcal{Z}$ is returned (lines 10–12). If all X_i, $1 \leq i \leq |Q|$, are processed without generating r documents, the loop at lines 4–16 terminates and $\mathcal{Z}$, containing less than r documents, is returned at line 17.

B.1. Generating X_i

To facilitate the generation of i-covers we initially ignore document boundaries and treat the documents in the target database as a single term sequence, essentially as a single large document. Together, the documents in the database are treated as an ordered sequence of terms

$$t_1, t_2, \ldots, t_N,$$

```
LevelRank(Q, i) ≡
    $\mathcal{S} \leftarrow \emptyset$; $\mathcal{Y} \leftarrow \emptyset$;
    $documents \leftarrow 0$;
    $(p, q) \leftarrow Cover(Q, i, 1)$;
    while $q \le N$ do
        $(p', q') \leftarrow DocumentBoundary(p)$;
        $score \leftarrow 0$;
        while $q \le q'$ do
            $score \leftarrow score + I(p, q)$;
            $(p, q) \leftarrow Cover(Q, i, p + 1)$;
        end while;
        if $score > 0$ then
            $documents \leftarrow documents + 1$;
            $\mathcal{S}[documents] \leftarrow score$;
            $\mathcal{Y}[documents] \leftarrow DocumentIdentifier(p', q')$;
        else
            $(p, q) \leftarrow Cover(Q, i, q')$;
        end if;
    end while;
    sort $\mathcal{Y}$ by score, $\mathcal{S}$;
    return $\mathcal{Y}$;
```

Fig. 2. Level ranking procedure — Given a query Q and a ranking level i, the procedure generates and ranks the documents in X_i. The procedure calls the *DocumentBoundary* procedure to generate the start and end positions of a document given a position in the document, and calls the *DocumentIdentifier* procedure to generate a document reference tag given the boundaries of the document. The Cover procedure of Fig. 3 is called to generate elements of the i-cover set.

where N is the sum of the lengths of the documents in the database. After generation, the i-covers are filtered against the document boundaries to produce and rank the elements of X_i.

Over this term sequence, a call $Cover(Q, i, k)$ to the procedure detailed in Fig. 3 generates the first i-cover for Q that starts at or after position k. If no i-cover starts at or after k, the value $(N + 1, N + 1)$ is returned instead. However, the text interval corresponding to the i-cover is not necessarily contained in a single document. The positions of document boundaries within this term sequence are recorded in an index, and this index is used to identify and eliminate i-covers that overlap document boundaries. This index is accessed using a procedure call $DocumentBoundary(p)$, which returns a pair (p', q') indicating the boundaries of the document that includes position p. The *DocumentBoundary* procedure can be implemented using standard index file structures, and details of the implementation will be not be discussed.

We now examine the details of the *LevelRank* procedure (Fig. 2), which generates a ranked set consisting of the documents in X_i. The procedure takes a set of query terms Q and a ranking level i as its arguments. As elements of X_i are generated they are stored in the indexed set $\mathcal{Y}$, with their scores stored in the array $\mathcal{S}$, where $\mathcal{S}[i]$ holds the score for document $\mathcal{Y}[i]$. The integer variable documents maintains the current size of $\mathcal{Y}$ and $\mathcal{S}$. The integer variables p,

```
$Cover(Q, i, k) \equiv$
    for $j \leftarrow 1$ to $|Q|$ do
        $\mathcal{R}[j] \leftarrow r(t_j, k)$;                    // $t_1, \ldots, t_{|Q|}$ elements of $Q$
    end for;
    $q \leftarrow i$th largest element of $\mathcal{R}$;
    $Q' \leftarrow \emptyset$;
    for $j \leftarrow 1$ to $|Q|$ do
        if $\mathcal{R}[j] \leq q$ then
            $Q' \leftarrow Q' \cup \{t_j\}$;
        end if;
    end for;
    for $j \leftarrow 1$ to $|Q'|$ do
        $\mathcal{L}[j] \leftarrow l(t'_j, q)$;                    // $t'_1, \ldots, t'_{|Q'|}$ elements of $Q'$
    end for;
    $p \leftarrow$ smallest element of $\mathcal{L}$;
    return $(p, q)$;
```

Fig. 3. Cover generation procedure — Given a query Q, a ranking level i, and a database position k, the procedure generates the first i-cover for Q starting at or after k. The procedure calls the l and r access functions to generate term positions.

q, p', q' maintain positions in the term sequence. The variable score holds intermediate and final values for the computation of a document's score.

Lines 1–2 initialize $\mathcal{S}$, $\mathcal{Y}$ and documents. The i-covers for Q are generated and processed from left to right over the term sequence. After processing, the score of an i-cover (p, q) either has contributed to the score of a document or the i-cover has been eliminated because it overlaps a document boundary. At the start of each iteration of the main loop (lines 4–18), the pair (p, q) contains the next i-cover to be processed. This invariant is established at line 3, where the *Cover* procedure is called to generate the first i-cover for Q and the result is assigned to (p, q). Once all i-covers have been processed, the *Cover* procedure returns the value $(N + 1, N + 1)$ and the loop terminates.

At line 5, the *DocumentBoundary* procedure is called to determine the start and end positions of the document containing position p, and the result is assigned to the pair (p', q'). Lines 6–10 compute the score for the document. At line 6, score is initialized to 0. At line 8, the score for the i-cover (p, q) is determined using Eq. (2) to compute $I(p, q)$, and the result is added to *score*. At line 9, the *Cover* procedure is called to generate the next i-cover, the first one starting after p. If the i-cover (p, q) ends after the boundary of the current document (i.e. $q > q'$), the loop over lines 7–10 terminates. If (p, q) overlaps a document boundary at the start of the main loop, before line 5, then $q > q'$ after line 6. In this situation, the body of the loop at lines 7–10 is never executed, and $score = 0$ after line 10. Otherwise, at least one i-cover is contained in the document with boundaries (p', q') and $score > 0$ after line 10.

At line 11, if $score > 0$ then the document and its score are recorded in $\mathcal{Y}$ and $\mathcal{S}$ at lines 12–14. The *DocumentIdentifier* procedure, called at line 14, is assumed to return a reference tag that can be used to represent the document in document sets. If *score* > 0, then (p, q) was generated at line 9 during the last iteration of the loop at lines 7–10 and has not yet been processed. If *score* $= 0$, then (p, q) overlaps q' and can be eliminated. In this case, the call to the *Cover* procedure at line 16 generates the next *i*cover to be processed, the first one starting at or after q'. Note that *i*-covers starting after p but before q' will overlap q' and need not be generated at all.

After the loop at lines 4–18 terminates, $\mathcal{Y}$ contains the documents in X_i, in the order they appear in the database, with their associated scores stored in $\mathcal{S}$. $\mathcal{Y}$ is sorted by score at line 19 and returned to the caller at line 20.

C.1. Generating i-covers

The procedure *Cover* (Fig. 3) takes as its arguments a set of query terms Q, a ranking level i and a term sequence position k and generates the first *i*-cover for Q that starts at or after k. The procedure depends on two *access functions* r and l that return positions in the term sequence $t_1, \ldots, t_N$. Both take a term t and a position in the term sequence k as arguments and return results as follows:

$$r(t, k) = \begin{cases} v & \text{if } \exists t_v = t \text{ such that } k \le v \\ & \text{and } \nexists t_{v'} = t \text{ such that } k \le v' < v\,. \\ N+1 & \text{otherwise} \end{cases}$$

and

$$l(t, k) = \begin{cases} u & \text{if } \exists t_u = t \text{ such that } k \ge u \\ & \text{and } \nexists t_{u'} = t \text{ such that } k \ge u' > u\,. \\ 0 & \text{otherwise} \end{cases}$$

Informally, the access function $r(t, k)$ returns the position of the first occurrence of the term t located at or after position k in the term sequence. If there is no occurrence of t at or after position k, then $r(t, k)$ returns $N + 1$. Similarly, the access function $l(t, k)$ returns the position of the last occurrence of the term t located at or before position k in the term sequence. If there is no occurrence of t at or before position k, then $l(t, k)$ returns 0. For example, over the poem *Erosion* from Section 2, r(‘sea’, 10) = 29, l(‘sea’, 10) = 5 and l(‘sea’}, 4) = 0.

Like the *DocumentBoundary* procedure, the r and l access functions may be implemented using standard index file structures. The MultiText implementation of cover density ranking uses extended inverted-list file structures that store a sorted list of positions for each term. This list may be efficiently addressed by position, directly supporting the r and l access functions and allowing portions of the list to be skipped when possible. The details of the file structures used in the MultiText implementation are given elsewhere (Clarke, Cormack, & Burkowski, 1994). Other file structures that permit portions of the term list to be skipped, such as the self-indexing structures of Moffat and Zobel (1996), are also appropriate.

The loop over lines 1–3 of Fig. 3 calls the r access function for each term in Q. For each term $t_j \in Q$, $1 \le j \le |Q|$, the position of its first occurrence at or after k is assigned to $\mathcal{R}[j]$. At line 4, the integer variable q is assigned the ith largest element of $\mathcal{R}$. That is, if the elements of $\mathcal{R}$ were sorted in increasing order q would be assigned $\mathcal{R}[i]$. From the definition of r, the extent (k, q) i-satisfies Q, and any extent (k,q') with $k \leq q' < q$ will not i-satisfy Q. Therefore, the first i-cover for Q starting at or after position k ends at position q. Lines 5–10 construct the set Q' consisting of the i terms from Q that appear in the text interval associated with (k, q). The loop over lines 11–13 calls the l access function for each term in Q'. For each term $t'_j \in Q'$, $1 \leq j \leq i$, the position of its last occurrence at or before q is assigned to $\mathcal{L}[j]$. At line 14, the integer variable p is assigned the smallest element of $\mathcal{L}$. From the definition of l, the extent (p, q) i satisfies Q, and any extent (p', q) with $p < p' \leq q$ will not i-satisfy Q. Therefore, (p, q) is an i-cover for Q. At line 15, (p, q) is returned to the caller.

References

Allan, J. (1995). Relevance feedback with too much data. In *18th annual international ACM SIGIR conference on research and development in information retrieval, Seattle* (pp. 337–343).

Callan, J. (1994). Passage-level evidence in document retrieval. In *17th annual international ACM SIGIR conference on research and development in information retrieval, Dublin* (pp. 302–310).

Church, K. W. (1995). One term or two? In *18th annual international ACM SIGIR conference on research and development in information retrieval (SIGIR'95) Seattle* (pp. 310–318).

Clarke, C. L. A., & Cormack, G. V. (1996). Interactive substring retrieval. In *Fifth Text Retrieval Conference (TREC-5) Gaithersburg, MD* (pp. 267–277).

Clarke, C. L. A., Cormack, G. V. & Burkowski, F. J. (1994). *Fast inverted indexes with on-line update*. Technical Report No. CS-94-40. University of Waterloo Computer Science Department.

Clarke, C. L. A., Cormack, G. V., & Burkowski, F. J. (1995). Shortest substring ranking. In *Fourth Text Retrieval Conference (TREC-4) Gaithersburg, MD* (pp. 295–304).

Clarke, C. L. A., Cormack, G. V., & Tudhope, E. A. (1997). Relevance ranking for one to three term queries. In *Fifth RIAO conference (Montreal)* (pp. 388–400).

Cormack, G.V., Clarke, C.L.A., Palmer, C.R., & To, S.S.-L. (1997). Passage based refinement. In *Sixth Text Retrieval Conference (TREC-6) Gaithersburg, MD*.

Cormack, G.V., Palmer, C.R., Van Biesbrouck, M., & Clarke, C.L.A. (1998). Deriving very short queries for high precision and recall. In *Seventh Text Retrieval Conference (TREC-7) Gaithersburg, MD*.

Croft, W. B., Cook, R., & Wilder, D. (1995). Providing government information on the Internet: experiences with THOMAS. In *Digital libraries conference (DL'95) Austin, TX* (pp. 19–24).

Frakes, W. B. (1992). Stemming algorithms. In W. B. Frakes, & R. Baeza-Yates, *Information retrieval — data structures and algorithms*. Englewood Cliffs, NJ: Prentice Hall.

Gross, A. D. (1991). Getty synoname: the development of software for personal names. In *Intelligent text and image handling (RIAO'91) Barcelona* (pp. 754–763).

Harman, D. (1991). How effective is suffixing? *Journal of the American Society for Information Science*, *42*(1), 7–15.

Harman, D. (1993). Overview of the first TREC conference. In *16th annual international ACM SIGIR conference on research and development in information retrieval Pittsburgh* (pp. 36–47).

Harman, D. K. (Ed.) (1995). *The fourth Text Retrieval Conference (TREC-4). Gaithersburg, MD*: National Institute of Standards and Technology (NIST), United States Department of Commerce. NIST Special Publication 500-236. (Available electronically at http://trec.nist.gov).

Hawking, D., & Thistlewaite, P. (1995). Proximity operators — so near and yet so far. In *Fourth Text Retrieval Conference (TREC-4), Gaithersburg, MD* (pp. 131–143).

Hawking, D., & Thistlewaite, P., (1996). Relevance weighting using distance between term occurrences. *Computer Science Technical Report No. TR-CS-96-08*. Australian National University.

Hawking, D., Thistlewaite, P., & Bailey, P. (1996). ANU's TREC-5 experiments. In *Fifth Text Retrieval Conference (TREC-5) Gaithersburg, MD* (pp. 359–375).

Hearst, M. A., & Plaunt, C. (1993). Subtopic structuring for full-length document access. In *16th annual international ACM SIGIR conference on research and development in information retrieval (Pittsburgh)* (pp. 59–68).

Jansen, B. J., Spink, A., Bateman, J., & Saracevic, T. (1998). Real life information retrieval: a study of user queries on the web. *SIGIR Forum*, *32*(1), 5–17.

Kaszkiel, M., & Zobel, J. (1997). Passage retrieval revisited. In *20th annual international ACM SIGIR conference on research and development in information retrieval (Philadelphia)* (pp. 178–185).

Keen, E. M. (1991). The use of term position devices in ranked output experiments. *Journal of Documentation*, *41*(1), 1–22.

Keen, E. M. (1992a). Some aspects of proximity searching in text retrieval systems. *Journal of Information Science*, *18*, 89–98.

Keen, E. M. (1992b). Term position ranking: some new test results. In *15th annual international ACM SIGIR conference on research and development in information retrieval (Copenhagen)* (pp. 66–76).

Knaus, D., Mittendorf, E., Schäuble, P., & Sheridan, P., (1995). Highlighting relevant passages for users of the interactive SPIDER retrieval system. *Fourth Text Retrieval Conference (TREC-4), Gaithersburg, MD*.

Koenemann, J., & Belkin, N. J. (1996). A case for interaction: a study of interactive information retrieval behavior and effectiveness. In *ACM SIGCHI conference on human factors in computing systems (Vancouver)* (pp. 205–212).

Moffat, A., & Zobel, J. (1996). Self-indexing inverted files for fast text retrieval. *ACM Transactions on Information Systems*, *14*(4), 349–379.

Pfeifer, U., Poersch, T., & Fuhr, N. (1996). Retrieval effectiveness of proper name search methods. *Information Processing and Management*, *32*(6), 667–679.

Porter, M. F. (1980). An algorithm for suffix stripping. *Program*, *14*(3), 130–137.

Pratt, E. J. (1989). *Complete poems*. University of Toronto Press.

Robertson, S. E., & Walker, S. (1994). 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) Dublin* (pp. 311–317).

Rose, D. E., & Stevens, C. (1996). V-Twin: a lightweight engine for interactive use. In *Fifth Text Retrieval Conference (TREC-5) Gaithersburg, MD* (pp. 279–290).

Salton, G., Allan, J., & Buckley, C. (1993). Approaches to passage retrieval in full text information systems. In *16th annual international ACM SIGIR conference on research and development in information retrieval (Pittsburgh)* (pp. 49–58).

Tudhope, E. (1996). *Query based stemming*. Unpublished master's thesis, University of Waterloo, Waterloo, Ont.

Voorhees, E. M. & Harman, D. K. (Eds.) (1996). *Information technology: the fifth Text Retrieval Conference (TREC-5). Gaithersburg, MD:* National Institute of Standards and Technology (NIST), United States Department of Commerce. NIST Special Publication 500-238. (Available electronically at http://trec.nist.gov).

Voorhees, E. M. & Harman, D. K. (Eds.) (1997). *The sixth Text Retrieval Conference (TREC-6). Gaithersburg, MD:* National Institute of Standards and Technology (NIST), United States Department of Commerce. NIST Special Publication 500-240. (Available electronically at http://trec.nist.gov).

Walker S., Robertson, S.E., Boughanem, M., Jones, G.J.F., & Jones, K.S. (1997). Okapi at TREC-6. *Sixth Text Retrieval Conference (TREC-6). Gaithersburg, MD*.

Wilkinson, R. (1994). Effective retrieval of structured documents. In *17th annual international ACM SIGIR conference on research and development in information retrieval (Dublin)* (pp. 311–317).

Wilkinson, R., & Zobel, J. (1994). Comparison of fragmentation schemes for document retrieval. In *Third Text Retrieval Conference (TREC-3) Gaithersburg, MD* (pp. 81–84).

Wilkinson, R., Zobel, J., & Sacks-Davis, R. (1995). Similarity measures for short queries. In *Fourth Text Retrieval Conference (TREC-4) Gaithersburg, MD* (pp. 277–285).