Available online at www.sciencedirect.com
SCIENCE DIRECT®
Information Systems 31 (2006) 573–584
www.elsevier.com/locate/infosys

Efficient query expansion with auxiliary data structures

Bodo Billerbeck*, Justin Zobel

School of Computer Science and Information Technology, RMIT University, Melbourne, Australia

Abstract

Query expansion is a well-known method for improving average effectiveness in information retrieval. The most effective query expansion methods rely on retrieving documents which are used as a source of expansion terms. Retrieving those documents is costly. We examine the bottlenecks of a conventional approach and investigate alternative methods aimed at reducing query evaluation time. We propose a new method that draws candidate terms from brief document summaries that are held in memory for each document. While approximately maintaining the effectiveness of the conventional approach, this method significantly reduces the time required for query expansion by a factor of 5–10.

Keywords: Query expansion; Efficiency; Document surrogates; Query associations

1. Introduction

Standard ranking techniques in information retrieval return documents that contain the same terms as the query. While the insistence on exact vocabulary matching is often effective, identification of some relevant documents involves finding alternative query terms. Previous work has shown that query expansion (QE) often significantly improves retrieval effectiveness [1–4].

Global analysis techniques (see for instance [5,6]) rely on heuristics such as term co-occurrence that are applied to the collection as a whole and are not directly query dependent. These techniques are inherently efficient during query evaluation as most of the analysis can be done at indexing time. However, they have been found to be generally less effective [2,7].

For local analysis methods, the original query is used to determine top-ranked documents from which expansion terms are subsequently extracted. A major drawback of such methods is the need to retrieve those documents during query evaluation, greatly increasing costs. In other work [8], we explored the use of surrogates built from past queries as a cheap source of expansion terms, but such surrogates require large query logs to be usable.

In this paper, we identify the factors that contribute to the cost of QE, and explore the alternatives for reducing these costs. Many of these approaches compromise effectiveness so severely that they are not of practical benefit. However, one approach is consistently effective: the use of brief summaries of each document that contain the most document-characteristic terms. These *surrogates* are much smaller than the source documents, and can

*Corresponding author.
E-mail addresses: bodob@cs.rmit.edu.au (B. Billerbeck), jz@cs.rmit.edu.au (J. Zobel).

0306-4379/$ - see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.is.2005.11.002

be rapidly processed during expansion. In experiments with several test sets, we show that our approach reduces the time needed to expand a query by a factor of 5–10, while approximately maintaining effectiveness compared to standard QE.

The remainder of the paper is structured as follows. Background on automatic QE and ranking functions used for the experiments in this paper is in Section 2. The steps involved in expanding queries using local analysis techniques are examined in terms of the effect on query throughput in Section 3. We then propose several approaches that are aimed at reducing the effect of the investigated practicalities in Section 4. Finally, we describe the experiments undertaken and analyse the results in Section 5.

2. Background

Relevance feedback is used to refine a query using knowledge of whether documents retrieved by the query are relevant. Weighted terms from judged documents are added to the original query, where they act as examples of the terms that should or should not occur in relevant and non-relevant documents. The modified query is then reissued, in the hope of ranking the remaining relevant documents more highly [1,9]. Interactive QE can significantly increase effectiveness [10], although on average—for non-expert users—automatic QE is more likely to lead to better performance [11].

In automatic QE, also called pseudo relevance feedback, the query is augmented with expansion terms from highly ranked documents [3]. An alternative [12,13] is to examine the document collection ahead of time and construct similarity thesauri to be accessed at query time. The use of thesauri in general has been shown to be less successful than automatic QE [14], though the two approaches can be successfully combined [7].

An effective method for QE, used throughout this paper, is based on the Okapi BM25 measure [3,15]. Slightly modified, this measure is as follows:

$$bm25(q,d)=\sum_{t\in q} w_t \times \frac{(k_1+1)f_{d,t}}{k_1((1-b)+b\times(|d|\times N)/\sum_{i\in N}|d_i|)+f_{d,t}},$$

where terms t appear in query q; the collection contains N documents; a particular term t occurs in a particular document $df_{d,t}$ times; constants k_1 and b, respectively, are set to 1.2 and 0.75; and $|d|$ is a measurement for the document length in a suitable unit.

The term weight w_t is based on the document frequency f_t (which measures how many documents contain term t) and calculated by a derivation of the inverse document frequency:

$$w_t=\log\left(\frac{N-f_t+0.5}{f_t+0.5}\right).$$

This formulation of w_t reduces the impact on the ranking of query terms that are common in the collection, and the second term of the *bm*25 formula promotes documents that contain a high proportion of query terms. The modifications to the original formulation (see [16] for a detailed explanation) is the omission of a component that deals with repeated query terms. We made the assumption that in most queries terms occur once only.

In this paper we use the expansion method proposed by [3], where E terms with the lowest *term selection value* are chosen from the top R ranked documents:

$$TSV_t=\left(\frac{f_t}{N}\right)^{r_t}\binom{R}{r_t},$$

where a term t is contained in r_t of the top-ranked R documents. The expansion terms get added to the original query, but instead of using w_t, their weight [3] is chosen by the formula[1]:

$$w'_t=\frac{1}{3}\times\log\left(\frac{(r_t+0.5)/(R-r_t+0.5)}{(f_t-r_t+0.5)/(N-f_t-R+r_t+0.5)}\right).$$

We have shown previously that best choices of R and E depend on the collection used and should in principle be carefully optimised [18]; to reduce the complexity of the experiments, in this paper we use the standard values of $R=10$ and $E=25$.

Although there has been a great deal of research on efficient evaluation of ranked queries [19, pp. 207–210], there is no prior work on efficient QE for text retrieval, the focus of this paper.

3. Query expansion practicalities

In most expansion methods making use of local analysis, there are five key stages. First, the original query is used to rank an initial set of documents. This set is then retrieved from disk and all terms are

[1]The factor of $\frac{1}{3}$ was recommended by unpublished correspondence with Steve Robertson. It de-emphasises expansion terms and prevents query drift, that is, "alteration of the focus of a search topic caused by improper expansion" [17]. We confirmed in unpublished experiments that the value of the factor is suitable.

extracted from those documents. Terms are evaluated and ranked in order of their potential contribution to the query. The top ranked terms are appended to the query, and finally the reformulated query is reissued and a final set of documents is ranked.

Each phase of the ranking process has scope for efficiency gains, but some of the gains involve heuristics that can compromise effectiveness. In this section we explore these options; this exploration provides a focus for the experiments reported later in this paper. Some of the concepts introduced here—in particular, associations and surrogates—are described in more detail in the next section.

3.1. Initial ranking

During the first stage, documents are ranked according to the original query. For each query term the inverted list is retrieved, if it has not already been cached, and processed. An inverted list specifies which documents that particular term occurs in. For each document referenced in the list, a score is calculated and added to a list of scores that is kept for (say) 20,000 documents [20]. Once all query terms have been processed, the top R documents are used for the next stage.

The cost of accessing an inverted list depends on the disk access time which is made up of three components: the seek time, the rotational delay, and the transfer time.

Seek time. A typical hard disk has an average seek time of roughly 10 ms. The seek time is only indirectly dependent on the list size, since average seek times are lower if inverted lists to be fetched are small, as they are more likely to be proximate on disk.

Rotational delay. Typically the average rotational delay is about 3 ms [21, p. 683]; since there is very little one can do to improve on rotational delay (apart from using fewer disk accesses), we do not consider it further.

Transfer time. The transfer time—that is, the time required to read a file from the hard drive into memory once the head is positioned correctly—is directly proportional to the size of the file and depends on the disk throughput. Disk throughput is the rate at which bytes are read from disk and fed onto the bus. It is typically on the order of 100 MB per second. Given that the size of a typical inverted list for a query term is far less than 1 MB[2] (inverted lists are typically stored in compressed form), the transfer time would usually be smaller than 10 ms.

If the list is organised by document identifier, the whole list must be fetched for each query term. A way of reducing the cost of retrieving and processing the inverted lists is to cut down the volume of list information that has to be retrieved. This has been achieved by, for example, Anh and Moffat [22], where documents are not stored in the order they are encountered during indexing, but in order of the *impact* a term has in a particular document. For instance, a term has more impact in a document in which it occurs twice, than in another document of the same length in which it occurs once. Using this ordering means that either the processing of lists can be stopped once a threshold is reached, or that the lists are capped to begin with, leading to lower storage requirements, reduced seek times, and allowing more lists to be cached in memory. We have not used impacts in our experiments, but the performance gains that they provide are expected to be in addition to the gains that we achieve with our methods. Efficiency should be increased even more than expected, since the expanded queries are—by definition—longer than original queries and therefore stand to gain considerably by using impact-ordered indexes.

Another way to reduce list length, discussed in more detail later, is to index only a fraction of the document collection for the initial ranking. The initial ranking is in all previous work on the document collection, but there is no particular reason why other collections should not be used. Another option of this kind is to use document surrogates, described in more detail in Section 4. A drawback of these approaches is that the full index still needs to be available for the final ranking and thus is loaded at the same time as auxiliary indexes. This means that some of the advantage of using shorter lists is negated by having less space available to cache them.

However, using the full text collection makes the initial ranking relatively least efficient, since in this case the index, including the in-memory dictionary and the inverted lists (stored on disk), is large. This

[2]For WT10 g the inverted list of a common term is up to 45 MB in size (2 MB if the list does not contain offsets to words, but only the document number and a count of how often a term appears within that document); the list of a less common term would be around 200 KB (or 100 KB without word offsets). Word offsets are needed in order to quickly evaluate phrase queries.

means that disk seek and transfer times for the inverted lists are increased and the likelihood of an inverted list being cached is decreased, given that memory volume is fixed and the dictionary as well as all inverted lists cannot be held in memory simultaneously.

3.2. Fetching documents

Having identified the highly ranked documents, these need to be fetched. In the vast majority of cases these documents are not cached from a previous expansion or retrieval process (assuming a typical memory size), and therefore have to be fetched from disk, at a delay of a few milliseconds each.

Fetching documents or surrogates from disk is costly, since disk accesses are slower than memory accesses by orders of magnitudes [21, pp. 390–391], whereas the larger the collection accessed, the higher the typical cost, due to greater seek times and reduced opportunity for caching.

Traditionally, full-text documents are fetched. Due to the high cost associated with disk accesses, which are much slower than memory access times, this is the most expensive stage of expansion and therefore the area where the greatest gains are available. We have shown previously that surrogates—which are a fraction of the size of the documents—can be more effective than full-text documents [8]. Using surrogates such as query associations is more efficient, provided that those surrogates can be precomputed, as discussed in Section 4.

Another approach is limiting the number of documents available for extraction of terms, which should result in higher efficiency, due to reduced cache misses when retrieving the remaining documents and otherwise smaller seek times as it can be expected that the limited number of documents are clustered on disk. Documents could be chosen by, for example, discarding those that are the least often accessed over a large number of queries [23].

A more radical measure is to use in-memory document surrogates that provide a sufficiently large pool of expansion terms, as described in the following section. If such a collection can be made sufficiently small, the total cost of expansion can be greatly reduced. Typically full text document collections do not fit into main memory, but well-constructed surrogates may be only a small fraction of the size of the original collection. Our surrogates are designed to be as small as possible while maintaining effectiveness.

3.3. Extracting candidate terms

Next, *candidate terms* (that is, potential expansion terms) are extracted from the fetched documents. These documents need to be parsed, and terms need to be stopped. (We do not use stemming in our experiments.)

This phase largely depends on the previous phase; if full text documents have been fetched, these need to be parsed and terms need to be stopped. In the case of query associations, the surrogates are pre-parsed and pre-stopped and extraction is therefore much more efficient. Since no additional information from disk is needed, costs are much greater than R, the number of documents fetched, and their length.

The in-memory surrogates we propose can be based on pointers rather than the full terms in memory. The pointers reference terms in the dictionary used for finding and identifying statistics and inverted lists. They have a constant size (4 bytes) and are typically smaller than a vocabulary term. This approach also eliminates the lookups needed in the next stage.

3.4. Selecting expansion terms

In this phase, all candidate terms are considered using the heuristics from the expansion method employed, and the best terms are then appended to the original query.

The information (such as the inverse document frequency) necessary for calculation of a term's TSV is held in the vocabulary, which may be held on disk or, as in our implementation, in memory; even when held on disk, the frequency of access to the vocabulary means that typically much of it is cached. As a result, this phase is the fastest and can only be sped up by providing fewer candidate terms for selection.

Query associations typically consist of 20–50 terms, as opposed to the average of 200 or more for web documents. Use of surrogates could make this stage several times more efficient than the standard approach. Surrogates are a strict subset of full text documents, and usually are a tiny fraction thereof, ensuring that selection is efficient.

3.5. Final ranking

Finally the document collection is ranked against the reformulated query. Similar considerations as in

the first phase are applicable here. We have shown previously [8] that final ranking against surrogates is, unsurprisingly, ineffective. The only option for significant efficiency gains at this stage is to use an approach such as impact-ordering, as discussed earlier.

4. Methods of increasing efficiency for QE

In the previous section we identified costs and plausible approaches for reducing them. In this section, we consider the most promising methods in more detail, setting a framework for experiments. In particular, we propose the novel strategy of using bag-of-word summaries as a source of expansion terms.

4.1. Query associations

Query associations [24] capture the topic of a document by associating past user queries with the documents that have been highly ranked by that query. Typically, for the association process a query log is used. Associations are derived as follows: every query of the query log is in turn submitted to a search system, and a similarity score is calculated based on the similarity of the query at hand to documents that contain query terms. The query then becomes associated with a fixed number of top documents that are returned. For efficiency, an upper bound is imposed on the number of queries that can become associated with a single document. Once a document has a full set of associations, the least similar associated query is dynamically replaced with a new, more similar query.

We have previously shown [8] that associations are effective when useful query logs are available. Query associations have since also been successfully used to increase the weighting of terms that encapsulate the "aboutness" of a document [25]. A disadvantage of using associations is that an extra index needs to be loaded and referenced during query evaluation. This extra index is small but not insignificant. The size of the associations is roughly 3% of the full text collection: for the 2 GB news wire collection of TRECs 5–8, associations take up 73 MB, or 233 MB for the 10 GB WT10g collection from TRECs 9 and 10. The advantages are that associations are usually pre-stemmed and stopped, stored in a parsed form, and cheap to retrieve. Most importantly, though, disk access times are much reduced: transfer times are reduced since associations are smaller than the documents they represent, which also means that seek times are reduced, assuming that associations are stored continuously. The cost of the term selection phase is also lower, since fewer candidate terms need to be evaluated.

Rather than indexing the associations, it would be possible in principle to rank using the standard index, then fetch and expand from the associations, but in our earlier work [8] we found that it was necessary to rank against the associations themselves.

We have assumed in the discussion above that associations are pre-computed; in a live environment such as the web, associations might need to be updated continuously.

4.2. Reducing collection size for sourcing expansion terms

The intuition underlying expansion is that, in a large collection, there should be multiple documents on the same topic as the query, and that these should have other pertinent terms. However, there is no logical reason why the whole collection should have to be accessed to identify such documents. Plausibly, documents sampled at random from the collection should represent the overall collection in respect of the terminology used. In our experiments, we sampled the collection by choosing every nth document, for n of 2 and 4. Other options would be to use centroid clusters or other forms of representative chosen on the basis of semantics. Documents could also be stored in a pre-parsed format (such as a forward index, see for instance [26]), which we have not tested.

4.3. In-memory document summaries

The major bottleneck of local analysis is the reliance on the highly ranked documents for useful expansion terms. These documents typically need to be retrieved from disk. We propose that summaries of all documents be kept in memory, or in a small auxiliary database that is likely to remain cached. A wide range of document summarisation techniques have been investigated [27], and in particular Lam-Adesina and Jones [28] have used summarisation for QE. In this work, representative sentences are selected, giving an abbreviated human-readable document.

However, summaries to be used for QE are not for human consumption. We propose instead that the summaries consist of the terms with the highest

tf.idf values, that is, the terms that the expansion process should rank highest as candidates if given the whole document. To choose terms, we use the function:

$$tf.idf = \log\left(\frac{N}{f_t}\right) \times \log(1 + f_{d,t}),$$

where N is the number of documents in the collection, f_t of which contain term t, and $f_{d,t}$ is the number of occurrences of t in document d.

Given these values, we can then build summaries in three ways. One is to have a fixed number S of highly ranked terms per document. The second is to choose a global threshold C, in which case each summary consists of all the document terms whose *tf.idf* value is lower than C. The third is to limit the volume of terms used in the summary to P% of the underlying document, up to a maximum of 100 terms. Again, the terms with the lowest *tf.idf* scores are chosen. Instead of representing summaries as sequences of terms, it is straightforward to instead use lists of pointers to the vocabulary representation of the term, reducing storage costs and providing rapid access to any statistics needed for the *TSV*. During querying, all terms in the surrogates that have been ranked against the original query are then used for selection. This not only avoids long disk I/Os, but also the original documents—typically stored only in their raw form—do not need to be parsed. S, C, and P can be chosen depending on collection size or available memory.

Although it is plausible that query-biased summaries [29]—as provided in most contemporary web search engines—would be more effective [28], such a method cannot be applied in the context of efficient QE, as query-biased summaries cannot be precomputed.

4.4. Other approaches

Since the original query terms effectively get processed twice during the ranking process, it seems logical to only process the original query terms during the initial ranking, and then, later, process the expansion terms without clearing the accumulator table that was used for the initial ranking.

However, as explored previously [20], limiting the number of accumulators aids efficiency and effectiveness. To support this strategy, query terms must be sorted by their inverse document frequency before the query is processed. Because most expansion terms have a high inverse document frequency—that is, they appear in few documents and are relatively rare—they must be processed before most of the original query terms, which typically have lower values. (The effect is similar, albeit weaker, to that of impact ordered indexes as discussed previously.) This means that the original query must be processed again with the expansion terms for final ranking. Intuition suggests that this argument is incorrect, and the original query terms should be allowed to choose the documents; however, in preliminary experiments we found that it was essential to process the original terms a second time. Processing only expansion terms in the second phase reduced costs, but led to poor effectiveness.

Other strategies could also lead to reduced costs. Only some documents, perhaps chosen by frequency of access [23] or sampling, might be included in the set of surrogates. A second tier of surrogates could be stored on disk, for retrieval in cases where the highly ranked documents are not amongst those selected by sampling. Any strategy could be further improved by compressing the in-memory surrogates, for example with d-gapping [19, pp. 115] and a variable-byte compression scheme [30].

Note that our summaries have no contextual or structural information, and therefore cannot be used—without major modifications—in conjunction with methods using such information, such as the local context analysis method of Xu and Croft [7] or the summarisation method of Goldstein et al. [27].

5. Experiments

Evaluating these approaches to QE requires that we test whether the heuristics degrade effectiveness, and whether they lead to reduced query evaluation time. To ensure that the time measurements were realistic, we used Zettair[3] as the underlying search engine. In order to cut down the size of the vocabulary without affecting effectiveness significantly, we indexed only terms that contain no more than a certain number of non-alphabetical characters [31]. This reduces the total number of unique terms by 30–40%, and leads to a similar decrease in the combined size of the inverted lists. In addition, it

[3]Zettair is an open source search engine being developed at RMIT University by the Search Engine Group. The primary aim in developing Zettair is to test techniques for efficient information retrieval. It is available from http://www.seg.rmit.edu.au/.

increases the likelihood that in-memory summaries contain useful candidate terms.

The test data is drawn from the TREC conferences [32]. We used two collections. The first was of newswire data, from TREC 7 and 8. The second was the WT10g collection, consisting of 10 GB of web data crawled in 1997 [33] for TREC 9 and 10. Each of these collections has two sets of 50 topics and accompanying relevance judgements. As queries, we used the title field from each TREC topic. We use the Wilcoxon signed rank test to evaluate the significance of the effectiveness results [34].

For timings, we used 10,000 stopped queries taken from two query logs collected for the Excite search engine [35]; these are web queries and thus are suitable for the WT10g runs. Since we were not able to obtain appropriate query logs for the newswire data, we used the same 10,000 queries for this collection. We used a dual Intel Pentium IV 2.8 GHz with 2 GB of main memory running Fedora Core 2. In earlier experiments, not reported here, we used an otherwise similar machine with much less main memory (768 MB) and observed comparable differences in timings [36].

5.1. Results

We used the TREC 8 and TREC 10 query sets to explore the methods. Results for this exploration are shown in Tables 1 and 2. These results are also summarised in Table 3. We applied the best methods found in these tables to the TREC 7 and TREC 9 query sets, as shown in Table 4. The tables detail the collection, the method of expansion, average precision, precision at 10, and r-precision values, as well as auxiliary memory required. A second index is needed for the runs where associations or fractional collections are used for initial ranking and candidate term extraction.

For TREC 8 and to a lesser extent TREC 10, standard QE improves over the baseline, but in both cases query evaluation takes around nine times as long. Several of the methods proposed do not succeed in our aims. Associations take nearly as long as standard QE, and effectiveness is reduced. For TREC 8 the surrogates are arguably inappropriate, as the web queries may not be pertinent to the newswire data; however, this issue highlights the fact that without a query log associations cannot be used.

Using halves ($n = 2$) or quarters ($n = 4$) of the collection also reduces effectiveness, and has little impact on expansion time; this is due to the need to load and access a second index. Larger n led to smaller improvements in QE; in experiments with $n = 8$, QE gave no improvements. Reducing R to roughly a quarter of its original size in order to cater for a smaller number of relevant documents—as

Table 1
Effectiveness of expansion of TREC-8 queries on the TREC newswire data

Expansion method	Query time (ms)	AvP	P@10	R-P	Mem (MB)
None	15	0.221	0.440	0.262	n/a
Standard	98	0.248‡	0.464	0.291‡	n/a
Associations	72	0.227	0.444	0.269†	Index
Eighth 1	80	0.183	0.330‡	0.228	Index
Eighth 2	71	0.172	0.338‡	0.211†	Index
Eighth 3	85	0.175	0.360‡	0.222	Index
Eighth 4	70	0.200	0.364‡	0.243	Index
Eighth 5	73	0.143‡	0.334‡	0.190‡	Index
Eighth 6	72	0.147‡	0.328‡	0.199†	Index
Eighth 7	71	0.186	0.368‡	0.227	Index
Eighth 8	69	0.177†	0.356‡	0.230	Index
Quarter 1	55	0.182†	0.362‡	0.234	Index
Quarter 2	55	0.206	0.394	0.245	Index
Quarter 3	69	0.234	0.410	0.280‡	Index
Quarter 4	56	0.227	0.396	0.269	Index
Half 1	76	0.234	0.428	0.281‡	Index
Half 2	65	0.236	0.426	0.280†	Index
$S = 1$	20	0.232‡	0.448	0.265	6
$S = 10$	20	0.237‡	0.444	0.272‡	24
$S = 25$	21	0.241‡	0.450	0.274†	54
$S = 50$	22	0.244‡	0.452	0.277‡	102
$S = 76$	24	0.246‡	0.462	0.285‡	146
$S = 100$	26	0.242‡	0.456	0.279‡	179
$C = 0.5$	21	0.231†	0.416†	0.273‡	11
$C = 1.0$	22	0.244‡	0.446	0.278‡	54
$C = 1.25$	23	0.246‡	0.456	0.287‡	82
$C = 1.5$	24	0.244‡	0.446	0.280‡	106
$C = 3.0$	24	0.241‡	0.458	0.279‡	79
$C = 0.25$–1.25	22	0.243†	0.454	0.284‡	81
$P = 1$	21	0.230†	0.452	0.275‡	8
$P = 10$	21	0.244‡	0.446	0.272†	38
$P = 15$	22	0.246‡	0.460	0.279‡	55
$P = 25$	23	0.244‡	0.454	0.279‡	85
$P = 50$	25	0.241‡	0.456	0.281‡	137
$P = 100$	25	0.242‡	0.456	0.279‡	179

Results shown are average precision (AvP), precision at 10 (P@10), and R-Precision (R-P). Also shown is the average query time over 10,000 queries and the amount of overhead memory required for each method; "index" marks the need to refer to an auxiliary index during expansion. A † marks results that are significantly different to the baseline of no expansion at the 0.10 level, and ‡ at the level of 0.05. S is the number of summary terms used, C specifies the cutoff threshold for the selection value, and P is the maximal percentage of the original document to be used for summaries.

Table 2
As in Table 1, showing effectiveness of expansion of TREC 10 queries on the WT10g collection

Expansion method	Query time (ms)	AvP	P@10	R-P	Mem (MB)
None	30	0.163	0.298	0.190	n/a
Standard	296	0.186	0.304	0.205	n/a
Associations	264	0.171	0.278	0.212	Index
Eighth 1	273	0.142	0.206‡	0.175	Index
Eighth 2	259	0.125†	0.218‡	0.155	Index
Eighth 3	279	0.143	0.263	0.172	Index
Eighth 4	270	0.127‡	0.214‡	0.147‡	Index
Eighth 5	249	0.146	0.267	0.172	Index
Eighth 6	259	0.143	0.247‡	0.173	Index
Eighth 7	297	0.149	0.229‡	0.185	Index
Eighth 8	253	0.130†	0.220‡	0.163	Index
Quarter 1	277	0.157	0.247†	0.188	Index
Quarter 2	258	0.142	0.245‡	0.164	Index
Quarter 3	276	0.153	0.265	0.202	Index
Quarter 4	269	0.134‡	0.218‡	0.163†	Index
Half 1	303	0.164	0.282	0.197	Index
Half 2	283	0.145	0.229‡	0.173	Index
$S = 1$	62	0.158	0.280	0.201	19
$S = 10$	78	0.176	0.300	0.212†	76
$S = 25$	81	0.179	0.306	0.202	165
$S = 50$	82	0.184	0.314	0.208	292
$S = 60$	83	0.188†	0.318	0.212	336
$S = 100$	86	0.186†	0.302	0.208	476
$C = 0.5$	79	0.166	0.296	0.190	44
$C = 1.0$	83	0.183†	0.312	0.202	186
$C = 1.05$	81	0.185†	0.312	0.202	200
$C = 1.5$	84	0.184	0.308	0.206	306
$C = 3.0$	86	0.186†	0.302	0.208	444
$C = 0.25$–1.05	79	0.182	0.308	0.202	197
$P = 1$	71	0.170	0.294	0.214	27
$P = 10$	75	0.180†	0.312	0.206	114
$P = 17$	76	0.187‡	0.316	0.205	171
$P = 25$	79	0.187‡	0.302	0.208	225
$P = 50$	81	0.184†	0.306	0.206	348
$P = 100$	82	0.186†	0.302	0.208	477

Table 3
Summarising the best effectiveness results from tuning collections TREC 8 and TREC 10 shown in Tables 1 and 2, respectively

Data set	Expansion method	Query time (ms)	AvP	P@10	R-P	Mem (MB)
TREC 8	None	15	0.221	0.440	0.262	n/a
	Standard	98	0.248‡	0.464	0.291‡	n/a
	$S = 76$	24	0.246‡	0.462	0.285‡	146
	$C = 1.25$	23	0.246‡	0.456	0.287‡	82
	$P = 15$	22	0.246‡	0.460	0.279‡	55
TREC 10	None	30	0.163	0.298	0.190	n/a
	Standard	296	0.186	0.304	0.205	n/a
	$S = 60$	83	0.188†	0.318	0.212	336
	$C = 1.05$	81	0.185†	0.312	0.202	200
	$P = 17$	76	0.187‡	0.316	0.205	171

intuition might suggest—only further degrades results. These results are consistent with previous work that shows that retrieval effectiveness, especially in the top-ranked documents, is greater for larger collections than for sub-collections [37]. This means that there is a higher likelihood of sourcing expansion terms from relevant documents when using local analysis QE if the largest collection is used. It was also found that QE works best when expansion terms are sourced from collections that are a superset of documents of the one targeted [38].

However, our simple *tf .idf* summaries work well. Even one-word ($S = 1$) summaries yield significantly improved average precision on TREC 8, for a memory overhead of only a few megabytes. The best cases were $S = 76$ on TREC 8 and $S = 60$ on TREC 10, where overall processing costs were less than a third those of standard QE. These gains are similar to those achieved by [28] with summaries of 6–9 sentences each, but our summaries are considerably more compact, showing the advantage of a form of summary intended only for QE. While the memory overheads are non-trivial—over 300 megabytes for TREC 10—they are well within the memory capacity of a desktop machine.

Results on TREC 7 for the summaries are equally satisfactory, with good effectiveness and low overheads. Results on TREC 9 are, however, disappointing. We had already discovered that expansion on TREC 9 does not improve effectiveness [18]; our results here are, in that light, unsurprising. The principal observation is that QE based on summaries is still of similar effectiveness to that based on full documents.

We show results for several parameter settings. These lead to similar effectiveness for similar memory overhead. Summaries and choice of S, C, and P are further examined in Figs. 1 and 2 for newswire and web data, respectively. These show that a wide range of S (top figure), C (centre figure), and P values (bottom figure) lead to improved effectiveness, in some cases exceeding that of standard QE.

The middle graph in Fig. 1 shows a drop in average precision if summaries consist only of terms with extremely low *tf .idf* values, of below 0.25. This would suggest that average precision could be

Table 4
As in Table 3, but showing results for test collections TREC 7 and TREC 9

Data set	Expansion method	Query time (ms)	AvP	P@10	R-P	Mem (MB)
TREC 7	None	15	0.191	0.452	0.246	n/a
	Standard	98	0.229‡	0.450	0.282‡	n/a
	$S = 76$	24	0.219‡	0.440	0.272‡	146
	$C = 1.25$	23	0.216‡	0.438	0.270‡	82
	$P = 15$	22	0.210	0.432	0.268‡	55
TREC 9	None	30	0.195	0.271	0.228	n/a
	Standard	296	0.182	0.260	0.210†	n/a
	$S = 60$	83	0.166	0.265	0.184†	336
	$C = 1.05$	81	0.166†	0.262	0.174‡	200
	$P = 17$	76	0.168†	0.267	0.187	171

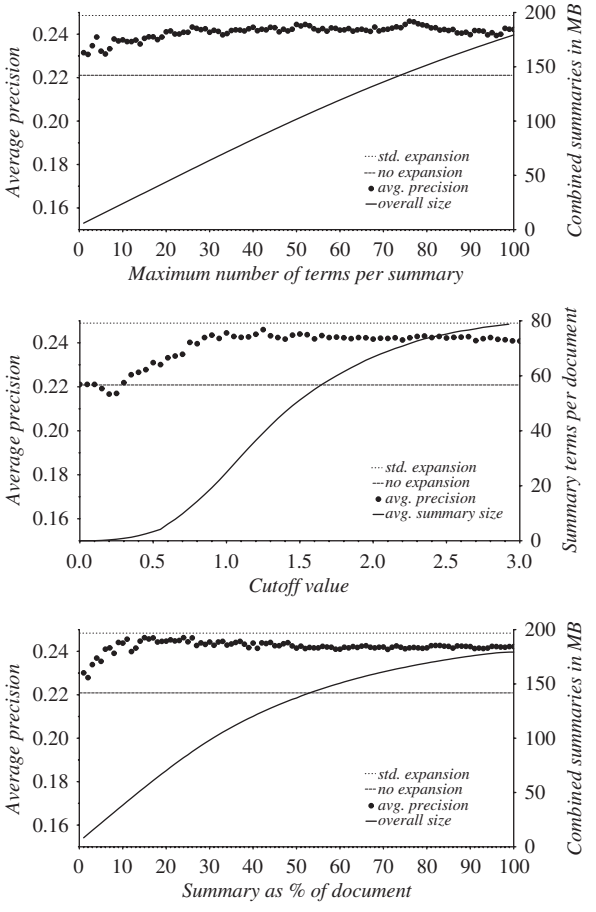

Fig. 1. Varying average precision and associated memory cost with the number, cutoff value of summary terms and percentage of document used for summaries, respectively. Using the TREC 8 collection and queries.

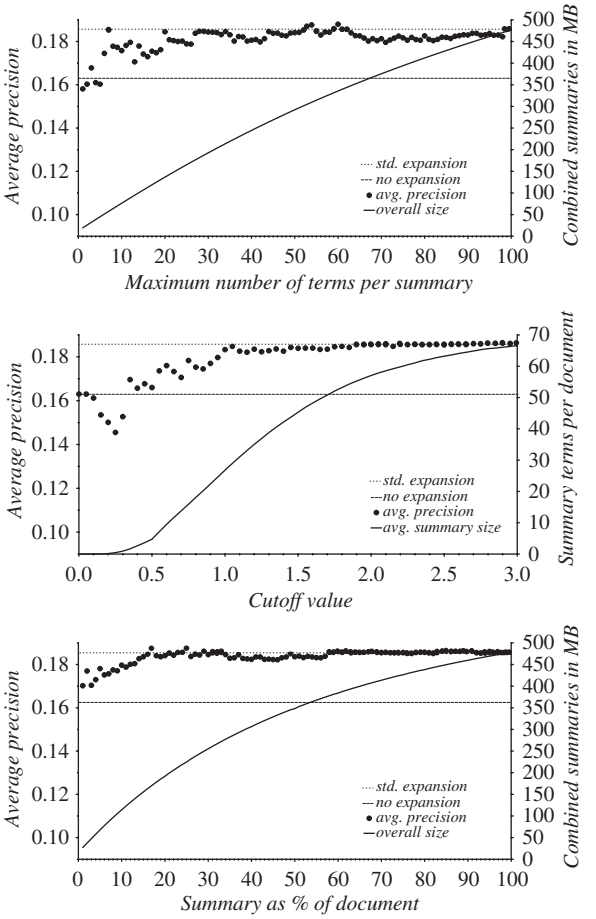

Fig. 2. As in previous figure, except that the TREC 10 collection and queries are used.

optimised by using only terms that have a *tf.idf* value which falls into the band between 0.25 and 1.25 for TREC 8 and between 0.25 and 1.05 for TREC 10, where the drop in average precision at the lower cutoff is even more pronounced. However, as shown in Tables 1 and 2, this is not the case,

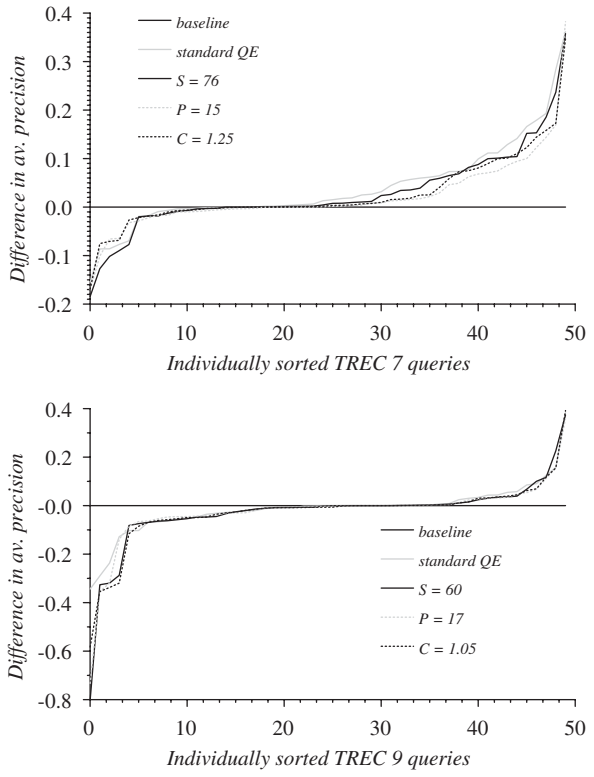

Fig. 3. The graphs show per-query-differences in average precision between each of the methods and the respective baselines.

which leads to the conclusion that some low *tf.idf* terms are chosen as expansion terms from a small pool of candidate terms to the detriment of queries. When the pool size is increased, other terms that are just as important in documents get selected before the first group of terms since they have a higher *tf.idf* value in other top ranked documents (that is, are not quite as important).

Fig. 3 compares the robustness of the different expansion methods on the data shown in Table 4. It can be seen that the effectiveness of our proposed summaries is comparable to that of the standard QE approach. However, improvements through summary-based QE are generally more moderate and a minority of queries is degraded more severely. This is most noticeable for the TREC 9 query set.

6. Conclusions

We have identified the main costs of QE and, for each stage of the query evaluation process, considered options for reducing costs. Guided by preliminary experiments, we explored two options in detail: expansion via reduced-size collections and expansion via document surrogates. Two forms of surrogates were considered: query associations, consisting of queries for which each document was highly ranked, and *tf.idf* summaries.

The most successful method was the *tf.idf* summaries. These are much smaller than the original collections, yet are able to provide effectiveness close to that of standard QE. The size reduction and simple representation means that they can be rapidly processed. Of the three methods for building summaries, slightly better performance was obtained with those consisting of a particular number of terms. The key to the success of this method is that it eliminates several costs: there is no need to fetch documents after the initial phase of list processing, and selection and extraction of candidate terms is trivial.

Many of the methods we explored were unsuccessful. Associations can yield good effectiveness if a log is available, but are expensive to process. Reduced-size collections yielded no benefits; it is possible that choosing documents on a more principled basis would lead to different effectiveness outcomes, but the costs are unlikely to be reduced. Streamlining list processing by carrying accumulator information from one stage to the next led to a collapse in effectiveness. Our *tf.idf* summaries, in contrast, maintain the effectiveness of QE while reducing time by a factor of 5–10.

Acknowledgements

This article is an extension of Billerbeck and Zobel [36]. The research is supported by the Australian Research Council, the State Government of Victoria and an RMIT VRII grant. We thank the SEG group members, in particular Hugh E. Williams, for suggesting the *P* method, Nick Lester and William Webber, for help with Zettair, and Steven Garcia for providing statistics of inverted list sizes.

References

[1] J.J. Rocchio, Relevance feedback in information retrieval, in: G. Salton (Ed.), The SMART Retrieval System—Experiments in Automatic Document Processing, Prentice-Hall, Englewood Cliffs, NJ, 1971, pp. 313–323.

[2] E.M. Voorhees, D.K. Harman, Overview of the Seventh Text REtrieval Conference (TREC-7), in: E.M. Voorhees,

D.K. Harman (Eds.), Proceedings of the Text Retrieval Conference (TREC), National Institute of Standards and Technology Special Publication 500-242, Gaithersburg, MD, 1998, pp. 1–24.

[3] S.E. Robertson, S. Walker, Okapi/Keenbow at TREC 8, in: Proceedings of the Text Retrieval Conference (TREC), NIST Special Publication 500-246, Gaithersburg, MD, 1999, pp. 151–161.

[4] C. Carpineto, R. de Mori, G. Romano, B. Bigi, An information-theoretic approach to automatic query expansion, ACM Trans. Inform. Syst. 19 (1) (2001) 1–27.

[5] G. Grefenstette, Explorations in Automatic Thesaurus Discovery, Kluwer Academic Publishers, Dordrecht, 1994.

[6] D.A. Evans, R.G. Lefferts, Design and evaluation of the CLARIT-TREC-2 system, in: Proceedings of the Text Retrieval Conference (TREC), 1993, pp. 137–150.

[7] J. Xu, W.B. Croft, Improving the effectiveness of information retrieval with local context analysis, ACM Trans. Inf. Syst. 18 (1) (2000) 79–112.

[8] B. Billerbeck, F. Scholer, H.E. Williams, J. Zobel, Query expansion using associated queries, in: Proceedings of the International Conference on Information and Knowledge Management, ACM Press, New York, 2003, pp. 2–9.

[9] I. Ruthven, M. Lalmas, A survey on the use of relevance feedback for information access systems, Knowl. Eng. Rev. 18 (2) (2003) 95–145.

[10] M. Magennis, C.J. van Rijsbergen, The potential and actual effectiveness of interactive query expansion, in: Proceedings of the ACM-SIGIR International Conference on Research and Development in Information Retrieval, ACM Press, New York, 1997, pp. 324–332.

[11] I. Ruthven, Re-examining the potential effectiveness of interactive query expansion, in: Proceedings of the ACM SIGIR International Conference on Research and Development in Information Retrieval, ACM Press, New York, 2003, pp. 213–220.

[12] Y. Qiu, H.-P. Frei, Concept based query expansion, in: Proceedings of the ACM SIGIR International Conference on Research and Development in Information Retrieval, ACM Press, New York, 1993, pp. 160–169.

[13] S. Gauch, J. Wang, A corpus analysis approach for automatic query expansion, in: Proceedings of the International Conference on Information and Knowledge Management, ACM Press, New York, 1997, pp. 278–284.

[14] R. Mandala, T. Tokunaga, H. Tanaka, Combining multiple evidence from different types of thesaurus for query expansion, in: Proceedings of the ACM-SIGIR International Conference on Research and Development in Information Retrieval, ACM Press, New York, 1999, pp. 191–197.

[15] S.E. Robertson, S. Walker, M. Hancock-Beaulieu, A. Gull, M. Lau, Okapi at TREC, in: Proceedings of the Text Retrieval Conference (TREC), 1992, pp. 21–30.

[16] K. Sparck Jones, S. Walker, S.E. Robertson, A probabilistic model of information retrieval: development and comparative experiments. Parts 1& 2, Inf. Process. Manage. 36 (6) (2000) 779–840.

[17] M. Mitra, A. Singhal, C. Buckley, Improving automatic query expansion, in: W.B. Croft, A. Moffat, C.J. van Rijsbergen, R. Wilkinson, J. Zobel (Eds.), Proceedings of the ACM-SIGIR International Conference on Research and Development in Information Retrieval, ACM Press, New York, Melbourne, Australia, August 1998, pp. 206–214.

[18] B. Billerbeck, J. Zobel, Questioning query expansion: an examination of behaviour and parameters, in: K.-D. Schewe, H.E. Williams (Eds.), Proceedings of the Australasian Database Conference, vol. 27, Australian Computer Society, Inc., 2004, pp. 69–76.

[19] I.H. Witten, A. Moffat, T.C. Bell, Managing Gigabytes: Compressing and Indexing Documents and Images, second ed., Morgan Kaufman, San Francisco, CA, 1999.

[20] A. Moffat, J. Zobel, Self-indexing inverted files for fast text retrieval, ACM Trans. Inf. Syst. 14 (4) (1996) 349–379.

[21] J.L. Hennessy, D.A. Patterson, Computer Architecture: A Quantitative Approach, Morgan Kaufmann, San Mateo, CA, 2002.

[22] V.N. Anh, A. Moffat, Impact transformation: effective and efficient web retrieval, in: Proceedings of the ACM-SIGIR International Conference on Research and Development in Information Retrieval, ACM Press, New York, 2002, pp. 3–10.

[23] S. Garcia, H.E. Williams, A. Cannane, Access-ordered indexes, in: V. Estivill-Castro (Ed.), Proceedings of the 27th Conference on Australasian Computer Science, vol. 26, Australian Computer Society, Dunedin, New Zealand, January 2004, pp. 7–14.

[24] F. Scholer, H.E. Williams, Query association for effective retrieval, in: C. Nicholas, D. Grossman, K. Kalpakis, S. Qureshi, H. van Dissel, L. Seligman (Eds.), Proceedings of the International Conference on Information and Knowledge Management, McLean, VA, 2002, pp. 324–331.

[25] F. Scholer, H.E. Williams, A. Turpin, Query association surrogates for web search, J. Am. Soc. Inform. Sci. Technol. 55 (7) (2004) 637–650.

[26] S. Brin, L. Page, The anatomy of a large-scale hypertextual web search engine, in: Proceedings of the World-Wide Web Conference, Elsevier Science Publishers B.V., Amsterdam, The Netherlands, 1998, pp. 107–117.

[27] J. Goldstein, M. Kantrowitz, V. Mittal, J. Carbonell, Summarizing text documents: sentence selection and evaluation metrics, in: Proceedings of the ACM-SIGIR International Conference on Research and Development in Information Retrieval, ACM Press, New York, 1999, pp. 121–128.

[28] A.M. Lam-Adesina, G.J.F. Jones, Applying summarization techniques for term selection in relevance feedback, in: Proceedings of the ACM-SIGIR International Conference on Research and Development in Information Retrieval, ACM Press, New York, New Orleans, LA, 2001, pp. 1–9.

[29] A. Tombros, M. Sanderson, Advantages of query biased summaries in information retrieval, in: Proceedings of the ACM-SIGIR International Conference on Research and Development in Information Retrieval, ACM Press, New York, 1998, pp. 2–10.

[30] F. Scholer, H.E. Williams, J. Yiannis, J. Zobel, Compression of inverted indexes for fast query evaluation, in: Proceedings of the ACM-SIGIR International Conference on Research and Development in Information Retrieval, ACM Press, New York, 2002, pp. 222–229.

[31] H.E. Williams, J. Zobel, Searchable words on the web, Int. J. Digital Libraries. 5 (2) (2005) 99–105.

[32] D. Harman, Overview of the second Text REtrieval Conference (TREC-2), Inf. Process. Manage. 31 (3) (1995) 271–289.

[33] P. Bailey, N. Craswell, D. Hawking, Engineering a multi-purpose test collection for web retrieval experiments, Inf. Process. Manage. 39 (6) (2003) 853–871.

[34] J. Zobel, How reliable are the results of large-scale information retrieval experiments?, in: W.B. Croft, A. Moffat, C.J. van Rijsbergen, R. Wilkinson, J. Zobel (Eds.), Proceedings of the ACM-SIGIR International Conference on Research and Development in Information Retrieval, ACM Press, New York, Melbourne, Australia, 1998, pp. 307–314.

[35] A. Spink, D. Wolfram, M.B.J. Jansen, T. Saracevic, From e-sex to e-commerce: web search changes, IEEE Comput. 35 (3) (2002) 107–109.

[36] B. Billerbeck, J. Zobel, Techniques for efficient query expansion, in: A. Apostolico, M. Melucci (Eds.), Proceedings of the String Processing and Information Retrieval Symposium, Springer, Padova, Italy, September 2004, pp. 30–42.

[37] D. Hawking, S.E. Robertson, On collection size and retrieval effectiveness, Kluwer Int. J. Inf. Retr. 6 (1) (2003) 99–150.

[38] K.L. Kwok, M. Chan, Improving two-stage ad-hoc retrieval for short queries, in: Proceedings of the ACM-SIGIR International Conference on Research and Development in Information Retrieval, ACM Press, New York, 1998, pp. 250–256.