# Efficient Query Expansion

A thesis submitted for the degree of
Doctor of Philosophy

Bodo Billerbeck, B.App.Sci/B.Eng. (RMIT University, Melbourne, Australia)
Hons.Eng. (RMIT University, Melbourne, Australia)
School of Computer Science and Information Technology,
Portfolio of Science, Engineering and Technology,
RMIT University,
Melbourne, Victoria, Australia.

September, 2005

## Declaration

I certify that except where due acknowledgement has been made, the work is that of the author alone; the work has not been submitted previously, in whole or in part, to qualify for any other academic award; the content of the thesis is the result of work which has been carried out since the official commencement date of the approved research program; and, any editorial work, paid or unpaid, carried out by a third party is acknowledged.

Preliminary versions of some results or discussions in this thesis have been previously published. Chapter 4 contains material that appeared in Billerbeck and Zobel (2003) and Billerbeck and Zobel (2004*a*). Findings described in this chapter also built the foundations of some of the work done in Billerbeck et al. (2004) and in Bernstein et al. (2005). Chapter 5 contains material that appeared in Billerbeck et al. (2003). Chapter 6 contains material that has been published in Billerbeck and Zobel (2004*b*) and Billerbeck and Zobel (to appear). Chapter 7 contains material that appeared in Billerbeck and Zobel (2005).

Bodo Billerbeck
School of Computer Science and Information Technology
RMIT University
September, 2005

## Acknowledgements

First and foremost I thank my primary supervisor Justin Zobel, who always made time to discuss research, and supported me as a friend through the last three years.

My relationship with Hugh Williams goes back some more years, when he first placed trust in me to coordinate tutors, and later to deliver lectures to large classes. It is him I want to thank for introducing me to academia, and for creating so many opportunities along the way.

When he left RMIT University, he was replaced as a second supervisor by Falk Scholer, who – together with Justin – gave me detailed feedback on drafts of the thesis and helped me to ’Englify’ it. Thank you to Falk and Hugh; it was very satisfying and a good experience all round as they collaborated with Justin and me on the work for Chapter 5. In particular I thank Falk (who was a student at that time) for generating the query associations and for his contributions to the work.

I also thank the many students in the Search Engine Group at RMIT University, in particular Nick Lester and William Webber for helping me understand the large code base of the search engine, but also all others who at various times brainstormed ideas with me and whose company in the SEG Cave and outside made the PhD even more enjoyable: Saied Tahaghoghi, Sandra Uitdenbogerd, Vaughan Shanks, Abhijit Chattaraj, Pauline Chou, Ranjan Sinha, Jovan Pehcevski, Steven Garcia, Halil Ali, Jelita Asian, Ying Zhang, Yaniv Bernstein, Timo Volkmer, Nikolas Askitis, Jonathan Yu, Michael Cameron and Ying Zhao. A big thanks to Kwong Lai, who has supported me all the way through my academic career.

I also thank Audrey Tam for her advice on all matters academic and otherwise, Michael Winikoff for giving me feedback on a draft of this thesis, and Rita Healy, Chin Scott and Beti Dimitrievska for their support.

I am indebted to Don Metzler of the Center for Intelligent Information Retrieval at the University of Massachusetts, who gave me a great deal of feedback and help while writing the section on language models.

Finally, I thank my parents for encouraging me to learn all throughout my life, and for supporting me in every possible way, and of course my partner, Heather Benbow, for supporting me in many ways throughout the duration of my PhD candidature.

This research was conducted with the support of an Australian Postgraduate Award with stipend, and a Victorian State Government Information and Communication Technology scholarship. Hardware for some experiments was provided with the support of an RMIT University VRII grant. Also, thanks to the School of Computer Science and Information Technology for their conference travel support and thanks to Excite (http:www.excite.com), who allowed us to use their query logs.

# Contents

# List of Figures

# List of Tables

# List of Algorithms

# Summary

Hundreds of millions of users each day search the web and other repositories to meet their information needs. However, queries can fail to find documents due to a mismatch in terminology. Query expansion seeks to address this problem by automatically adding terms from highly ranked documents to the query. While query expansion has been shown to be effective at improving query performance, the gain in effectiveness comes at a cost: expansion is slow and resource-intensive.

Current techniques for query expansion use fixed values for key parameters, determined by tuning on test collections. We show that these parameters may not be generally applicable, and, more significantly, that the assumption that the same parameter settings can be used for all queries is invalid. Using detailed experiments, we demonstrate that new methods for choosing parameters must be found.

In conventional approaches to query expansion, the additional terms are selected from highly ranked documents returned from an initial retrieval run. We demonstrate a new method of obtaining expansion terms, based on past user queries that are associated with documents in the collection.

The most effective query expansion methods rely on costly retrieval and processing of feedback documents. We explore alternative methods for reducing query-evaluation costs, and propose a new method based on keeping a brief summary of each document in memory. This method allows query expansion to proceed three times faster than previously, while approximating the effectiveness of standard expansion.

We investigate the use of document expansion, in which documents are augmented with related terms extracted from the corpus during indexing, as an alternative to query expansion. The overheads at query time are small. We propose and explore a range of corpus-based document expansion techniques and compare them to corpus-based query expansion on TREC data. These experiments show that document expansion delivers at best limited benefits, while query expansion – including standard techniques and efficient approaches described in recent work – usually delivers good gains. We conclude that document expansion is unpromising, but it is likely that the efficiency of query expansion can be further improved.

Chapter 1

# Introduction

Large volumes of data on the web or other sources provide users with a wealth of information. However, archives are becoming more diverse and the language used in text is increasingly diverse; this is particularly true for the web. Two authors writing about the same topic may use a different vocabulary to express themselves. Users who try to find information on this topic ideally express their information need in words that might have been used by both authors. However, this is difficult without prior knowledge of the authors' preferred terminology. Query expansion is a method to overcome this mismatch in terminology: in addition to the original terms that a searcher used to express their information need, other related words are appended to the query, so that documents which are on the same topic can be found even if they do not contain the original query terms.

An example of such a situation is as follows. A mechanic might be interested in the types of spanners that are available in the marketplace. Situated in Australia, a typical query might be: "spanner types". A query formulated as such would exclude all spanners sold in the United States, where they are known as "wrenches". The query might therefore be expanded to include this term: "spanner wrench types".

The problem of terminology mismatch is illustrated in Figure 1.1. In some of the recent TREC collections (described in detail in Section 2.6.2), between 10% and 25% of all known relevant documents contain none of the query terms apart from stop words. The actual proportion of relevant documents that do not contain any of the query terms might be even larger than this figure suggests: in the TREC framework the relevance of only a relatively small number of documents is judged for each query, and documents containing the query terms are more likely to have been ranked and subsequently judged than those that do not.

In this thesis we show that, despite extensive previous research, query expansion does not provide consistent gains across different query sets and collections. Nevertheless it has potential to be used effectively for many specialised applications. However, if it can not be made use of efficiently, it will not be used in an on-demand search system as users are reluctant to wait more than a minimum amount of time for results.

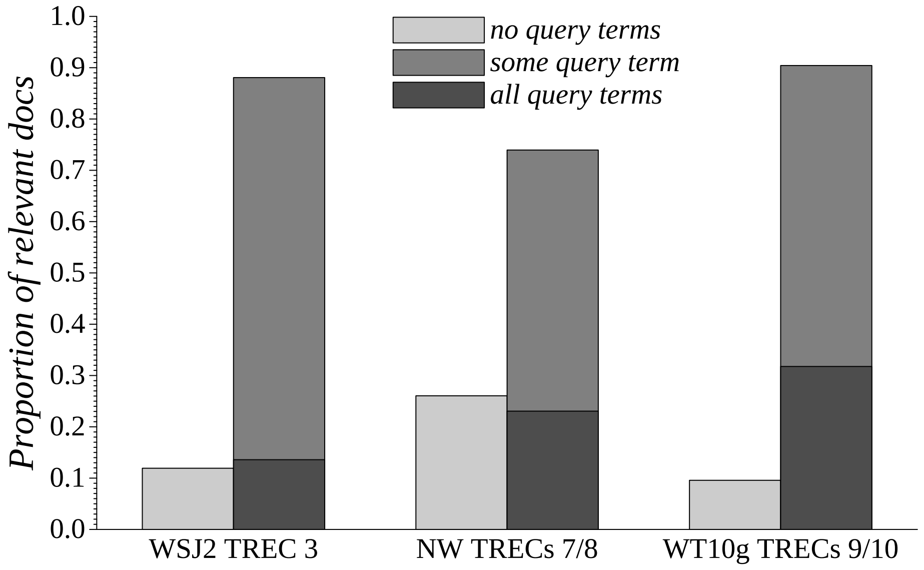


*Figure 1.1: The proportion of relevant documents that contain none, some, or all query terms over TREC title queries for different collections (all collections are discussed in Section 2.6.2), and query sets, respectively. Stopping, but no stemming was used to generate this graph.*

## 1.1 How reliable is query expansion?

Although query expansion has been shown to give good improvements on many collections, it is not clear that queries are consistently improved. An investigation into the robustness of query expansion is needed to determine whether average improvements of query expansion are hiding the fact that the retrieval effectiveness of some queries is severely degraded.

Query expansion has been widely researched. In this thesis, we focus primarily on one particular form of query expansion, known as local analysis. For this form of expansion, a set of documents is ranked according to the original query. From the documents of this initial set, terms are selected and added to the original query. This new, expanded, query is then re-run on the collection.

Two parameters in particular are important in local analysis query expansion. One is the number of documents from which to retrieve possible expansion terms, and the other is the total number of expansion terms to be added to a query. One extreme is to not add any expansion terms to the query, thus preserving the original query and leaving retrieval effectiveness unchanged. On the other end of the scale, all available terms would be added to the query. Adding any new term introduces a risk that the topicality of the original query may be changed, taking search into a different direction. Using the previous example “spanner types” and adding such a large number of term, the query could drift to include all different types of tools, rather than only be aimed at spanners or wrenches.

Most techniques use fixed parameters for expansion. It is therefore valuable to investigate what impact these parameters have, and whether the underlying assumption that one single optimal setting for all queries and collections can be determined is correct.

Using comprehensive experiments on two test collections, we have investigated both average effectiveness and per-query effectiveness for a wide range of parameter choices. Our results show that it is far from clear that the standard parameter choices are optimal. Other choices of values can lead to higher effectiveness, but no fixed choice is robust: entirely different values are preferable for different collections. Furthermore, the optimal parameter choices vary widely from query to query. We conclude that the assumptions underlying most current approaches to query expansion are not well founded.

However, our results also show that the performance of query expansion has significant scope for improvement: tuning parameters for individual queries can give much better performance than using fixed values. Whether such tuning is feasible is an open question.

## 1.2 Can web search effectiveness be improved through query expansion?

Query expansion works well on coherent collections such as newswire data; see for instance Mitra et al. (1998). A technique that is designed to increase effectiveness is the association of past user queries to documents, also known as query association (see Section 5.1.3 for details). We investigate how query associations can be used for query expansion in a web search environment, and experimentally evaluate the effectiveness of this approach.

Given a log containing a large number of queries, it is straightforward to build a surrogate for each document in a collection, consisting of the queries that were a close match to that document. Scholer et al. (2004) have shown that query associations can provide a useful document summary; that is, the queries that match a document are a fair description of its content. Here, we investigate whether query associations can play a role in query expansion.

Ranking with query expansion consists of three phases: ranking the original query against the collection or a document set; extracting additional query terms from the highly ranked items; then ranking the new query against the collection. We show that query associations can be a highly effective source of expansion terms. On the TREC-10 data, average precision rises from 0.158 for full text expansion with optimised expansion parameters to 0.189 for expansion via association, a dramatic relative improvement of 19.5%.

## 1.3 How can the efficiency of local analysis query expansion be improved?

Expanding queries through local analysis is one of the most effective methods of reformulating queries without relying on user input, as detailed in Section 3.3.2. However, in order for query expansion to be useful in a production environment, it should not slow down query evaluation, as users expect answers in the shortest amount of time possible. If there is a choice between a system that employs query expansion to produce slightly better results, and one that does not but returns answers considerably faster, the user is likely to choose the latter. In this thesis, we investigate the steps in the query expansion process that have the greatest impact on retrieval speed.

While exploring the aspects of each of the steps in regards to their impact on efficiency, we confirmed that the largest bottleneck stems from the need to retrieve documents from disk. Disks are a secondary data storage device, and are significantly slower than main memory and the CPU cache. This makes disks the slowest of the computer components that are typically used in the query expansion process. Our results demonstrate that disk accesses are by far the biggest contributor to the overall time requirements to expand queries. In our experiments, the standard approach to query expansion slows down retrieval by a factor of six.

Since retrieval is mostly slowed down by frequent disk accesses, which are inherently very time-intensive, we investigate whether it is possible to circumvent disk accesses completely. We explore alternatives for reducing the costs of expanding queries through local expansion. 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 that are a pool of the most important terms of each document. These *surrogates* are much smaller than the source documents, and can be rapidly processed during expansion. In experiments with several test sets, we show that our approach reduces the time needed to expand and evaluate a query by a factor of three, while approximately maintaining effectiveness compared to standard query expansion.

## 1.4 Is document expansion a viable alternative to query expansion?

Additional terms are appended to queries using query expansion. Seeing that this process involves at least a minimal computational overhead, a possible alternative is to reverse the process and expand documents instead of expanding queries.

We explore the use of document expansion as an alternative to query expansion. In document expansion, documents are enriched with related terms. Although there is a significant cost associated with expanding documents, this is undertaken at indexing time; there is only a marginal additional cost at retrieval time. It is plausible that document expansion should help to resolve the problem of vocabulary mismatch, and thus yield benefits like those obtainable with query expansion.

We propose two new corpus-based methods for document expansion. The first method is based on adding terms to documents in a process that is analogous to query expansion: each document is run as a query and is subsequently augmented by expansion terms. The second method is based on regarding each term in the vocabulary as a query, which is expanded using query expansion and used to rank documents.

The original query term is then added to the top-ranked documents. In both of these approaches there are several parameters that need to be tuned.

Our experiments measure the efficiency and effectiveness of query expansion and document expansion on several collections and query sets. We find that, on balance, document expansion leads to improvements in effectiveness, but few of the measured gains are statistically significant. Importantly, the computational cost at query time is small. In contrast, both standard query expansion and the more efficient query expansion that makes use of summaries as proposed above lead to gains in most cases, many of them significant, while an efficient query expansion process is less than twice the cost of querying without expansion.

We tested several alternative configurations of document expansion and explored the parameters, but did not observe useful gains in effectiveness. We conclude that corpus-based document expansion is unpromising, while we found query expansion to improve retrieval consistently, and thus believe that further gains in performance may be available.

## 1.5 Thesis structure

Chapter 2 gives an overview over the relevant background in information retrieval. In particular we discuss what constitutes an information need and how this might be expressed as a query. We also introduce document indexing, the concept of relevance, how queries are evaluated in general, and how they can be evaluated efficiently. In Section 2.6, we describe in detail some retrieval models, in particular the vector space model, probabilistic models, and language models. Additional data that can be made use of for web retrieval, such as anchor text and past queries, is also considered. Finally we explain how we measure the performance of our experiments, in terms of efficiency and effectiveness, the latter using the data sets provided by TREC.

A history of relevance feedback and query expansion is given in Chapter 3. We first discuss what constitutes an optimal query and how relevance feedback can be made use of in order to get arbitrarily close to the optimal query. We then discuss interactive query expansion, where user input is sought not on the relevance of documents, but on the usefulness of possible expansion terms. Finally we give an overview of past and present techniques for expanding queries automatically. In the past, automatic query expansion focused on thesauri and dictionaries, whereas modern approaches favour a technique that relies on the documents retrieved using the initial query to deliver appropriate expansion terms, named local analysis. Efficient approaches to the latter technique are the topic of this thesis.

In Chapter 4 we explore what impact different parameter settings for local analysis query expansion have on retrieval effectiveness. In particular, we investigate how the effectiveness of query expansion varies between different collections and individual queries, and what the optimal parameter settings for collections and queries are. We also examine the robustness of local analysis query expansion.

Our work in Chapter 5 is based on query association. We therefore give an outline of this approach of linking past user queries to web documents. We then explain how associations can be made use of for query expansion in the context of web retrieval.

In Chapter 6 we discuss the considerations for a more efficient approach to local analysis. In particular, we describe in detail what the bottlenecks for query expansion are and speculate about how those could be alleviated. We propose the use of in-memory data structures to remove the need for disk accesses to retrieve documents, which is the biggest obstacle to a more efficient query expansion solution.

Document expansion for ad-hoc retrieval is explored as an alternative to local analysis in Chapter 7. We propose two different approaches to document expansion, one centred on each document in the collection, the other on the terms in the vocabulary.

Finally, we conclude this thesis in Chapter 8 and give an outline of avenues for future research.

# Chapter 2

# Information Retrieval

The largest text repositories today hold terabytes of data. This prompts the question: how can large amounts of data be searched efficiently in order to find the required information? The naïve approach is a manual search, where the researcher brings up one record after another onto the screen and checks whether it fulfils the search criteria. Due to the numbers of documents in typical repositories, this is infeasible.

Information retrieval attempts to identify documents that the searcher is looking for. Typically this involves the searcher providing some keywords to the retrieval engine, which then uses these to search the collection.

## 2.1 Information needs and queries

To begin with, the user is in a *problematic situation* (Belkin et al., 1993) or has a *real information need* (Mizzaro, 1998), of which they might not even be fully aware. From this real information need the *perceived information need* (or simply *information need*) arises. The keywords used to formulate the *query* are therefore only a particular representation of what the searcher is looking for.

For instance, an osteoporosis researcher might have the real information need that they lack knowledge about osteoporosis patients that have suffered multiple fractures. While the researcher might not be completely conscious of this lack of knowledge, they are aware that they would like to find out more about those patients, which is an information need. A query is then an attempt by the user to formulate their information need, whereas the information need may include a much more detailed context. In this example, a query could be formulated as “osteoporosis multiple fractures”. The information need also depends on any prior information available that might not be wanted in the result set. In our example, the searcher might already be familiar with the cases of the last two years in their affiliated hospital, and therefore would not be interested in records from that time. It is impossible for a query to be an exact and encompassing formulation of the information need because – even if the user was able to express their information need completely in the query – it lacks the context in which the user is looking for the information. The query is therefore not an exact verbalisation of the information need; it does not encapsulate all the information the researcher already has and does not need to retrieve again, and it lacks any contextual information. It follows that the same information need may be expressed as several different queries, none of which is sufficient to encompass the whole information need on its own.

Conversely, the same query can represent different information needs, since it is a relatively small and more or less concise statement of what a searcher is looking for. An example would be that a doctor poses the same query ("osteoporosis multiple fractures") but rather than being interested in all patients who had multiple fractures, the doctor may want to retrieve the medical record of a particular patient whose name they cannot remember, but still knows why the patient saw them. In this case the query might take the same form as before but the intended result set is quite different.

Furthermore, an information need is typically not static, that is, the information a user is seeking changes during the course of a search (Ellis, 1989). One reason is the aforementioned criteria that, in many search scenarios, information that addresses an aspect of the user's information need has previously been seen by the user, and is therefore not helpful in further fulfilling the user's information need (Spink and Wilson, 1999). Recent research within the TREC framework (discussed in Section 2.6.2) is concerned with this issue, in the form of the *novelty* track (Soboroff and Harman, 2003).

The other – and less predictable – reason is that the actual information need that a user had at the outset of the search can change as the user reads more and more documents (Kuhlthau, 1993). It could be said that the focus of the search is shifting. Often this directional change of the search is large enough that a change of the initial query is warranted. In the above example the user chooses the query "osteoporosis multiple fractures" with the intention of finding the medical records of all patients that had multiple fractures. However, while searching and reading the first records, the user discovers that it would actually be more interesting to investigate the circumstances in which femurs in particular were fractured. Hence a new query, better suited to the new information need, could be: "osteoporosis fractures femur". While this *transient* nature of searching has strong implications for information retrieval, it is not investigated in this thesis. In this chapter we explain how retrieval systems attempt to satisfy information needs.

## 2.2 Indexing

Just having access to data is not of much use in cases where the data is large and only a particular part of the data needs to be viewed. In the example above, the researcher has access to all data about all previous patients, but without the means to efficiently search those, this is not useful for the task at hand. One way to overcome this is by ordering the data in a meaningful way, however this carries certain difficulties. When taking the query "osteoporosis multiple fractures" as an example, it would be helpful if data was sorted by the type of injury or diagnosis. While the data can be arranged in this way – for instance, using

| Document ID | Document text |
|---|---|
| Document 1: | “How unfair! Only one health, and so many diseases.”[1] |
| Document 2: | “The miserable have no other medicine but only hope.”[2] |
| Document 3: | “Nearly all men die of their medicines, not of their diseases.”[3] |
| Document 4: | “Medicine, the only profession that labors incessantly to destroy the reason for its own existence.”[4] |

*Table 2.1: Example collection of documents. This collection is used throughout this chapter to illustrate issues and techniques related to information retrieval.*

a sophisticated directory structure – the searcher then faces the problem of finding the relevant records with minimal effort.

A better option is to use an *index*. The first concordance – an elaborate form of an index, where each term is explained in the contexts it appears in – was completed in 1230 by Hugo de Sancto Charo, who led 500 fellow Dominicans to index the bible in Latin (Witten et al., 1999, page 19). In information retrieval, an index is a list of the terms that occur in a collection along with references as to where these terms can be found. As indexes are one of the important building blocks of a search engine, various construction algorithms have been developed (see for instance Moffat and Bell, 1995, or Heinz and Zobel, 2003).

Consider the example collection of documents shown in Table 2.1. This collection is used in the remainder of this chapter to illustrate issues and techniques related to information retrieval.

A list for each term in a collection contains pairs of document identifiers, together with the number of occurrences within that document (the *document frequency*). An index of our example collection, alphabetically sorted by indexed term, is shown in Table 2.2. During query evaluation, discussed in detail in Section 2.4, the documents within which each query term occurs are found using the index.

The index structure used in most practical information retrieval systems is an *inverted index* or *inverted file*. During traditional index construction, a list of terms for each document is created, effectively forming a large matrix of documents and terms, where the number of term occurrences for each document is accumulated. This initial matrix is also referred to as a *forward index*. Upon completion of the data gathering process, each row of the matrix gives the number of occurrences of every single term in the collection for a particular document. The matrix is then inverted, and each row now shows the number of occurrences for each document for a particular term. The lists for each term detailing where the term occurs, are referred to as *inverted lists*.

[1]Victor Schlichter.
[2]William Shakespeare’s “Measure for Measure”.
[3]Moliere.
[4]James Bryce.

| Term | Inverted list | Term | Inverted list |
|---|---|---|---|
| all: | $\langle\langle 3,1\rangle\rangle$ | men: | $\langle\langle 3,1\rangle\rangle$ |
| and: | $\langle\langle 1,1\rangle\rangle$ | miserable: | $\langle\langle 2,1\rangle\rangle$ |
| but: | $\langle\langle 2,1\rangle\rangle$ | nearly: | $\langle\langle 3,1\rangle\rangle$ |
| destroy: | $\langle\langle 4,1\rangle\rangle$ | no: | $\langle\langle 2,1\rangle\rangle$ |
| die: | $\langle\langle 3,1\rangle\rangle$ | not: | $\langle\langle 3,1\rangle\rangle$ |
| diseases: | $\langle\langle 1,1\rangle\ \langle 3,1\rangle\rangle$ | of: | $\langle\langle 3,2\rangle\rangle$ |
| existence: | $\langle\langle 4,1\rangle\rangle$ | one: | $\langle\langle 1,1\rangle\rangle$ |
| for: | $\langle\langle 4,1\rangle\rangle$ | only: | $\langle\langle 1,1\rangle\ \langle 2,1\rangle\ \langle 4,1\rangle\rangle$ |
| have: | $\langle\langle 2,1\rangle\rangle$ | other: | $\langle\langle 2,1\rangle\rangle$ |
| health: | $\langle\langle 1,1\rangle\rangle$ | own: | $\langle\langle 4,1\rangle\rangle$ |
| hope: | $\langle\langle 2,1\rangle\rangle$ | profession: | $\langle\langle 4,1\rangle\rangle$ |
| how: | $\langle\langle 1,1\rangle\rangle$ | reason: | $\langle\langle 4,1\rangle\rangle$ |
| incessantly: | $\langle\langle 4,1\rangle\rangle$ | so: | $\langle\langle 1,1\rangle\rangle$ |
| its: | $\langle\langle 4,1\rangle\rangle$ | that: | $\langle\langle 4,1\rangle\rangle$ |
| labors: | $\langle\langle 4,1\rangle\rangle$ | the: | $\langle\langle 2,1\rangle\ \langle 4,2\rangle\rangle$ |
| many: | $\langle\langle 1,1\rangle\rangle$ | their: | $\langle\langle 3,2\rangle\rangle$ |
| medicine: | $\langle\langle 2,1\rangle\ \langle 4,1\rangle\rangle$ | to: | $\langle\langle 4,1\rangle\rangle$ |
| medicines: | $\langle\langle 3,1\rangle\rangle$ | unfair: | $\langle\langle 1,1\rangle\rangle$ |

*Table 2.2: Example of an inverted index: for each term $t$ there is a list that consists of one or more entries, each containing the identifier of document $d$ and the number of times t occurs therein ($f_{d,t}$).*

An alternative method of indexing was to use *signature files* (see for instance Faloutsos (1985) or Sacks-Davis et al. (1987)), where each document is allocated a certain number of bits, a signature. Every word in a particular document is hashed several times and bits in the signature are set accordingly. During query evaluation, a signature for the query is established in the same way and compared to the document signatures in the index. When a potential match is found, the actual document needs to be checked whether the query terms actually appeared, since a bit set in the signature could have also been set because another word with the same signature leads to this bit being set. Zobel et al. (1998) found that signature files are inferior compared to inverted indexes in terms of query evaluation speed, storage requirements, and functionality.

In earlier times only a limited number of keywords was assigned to each document to aid the retrieval of that document (Maron, 1961), often from a physical archive. Modern indexing methods for text document retrieval typically include all terms that appear in a document in the index.

| term | inverted list | term | inverted list |
|---|---|---|---|
| destroy: | $\langle\langle 4,1\rangle\rangle$ | medicine: | $\langle\langle 2,1\rangle\ \langle 4,1\rangle\rangle$ |
| die: | $\langle\langle 3,1\rangle\rangle$ | medicines: | $\langle\langle 3,1\rangle\rangle$ |
| diseases: | $\langle\langle 1,1\rangle\ \langle 3,1\rangle\rangle$ | men: | $\langle\langle 3,1\rangle\rangle$ |
| existence: | $\langle\langle 4,1\rangle\rangle$ | miserable: | $\langle\langle 2,1\rangle\rangle$ |
| health: | $\langle\langle 1,1\rangle\rangle$ | nearly: | $\langle\langle 3,1\rangle\rangle$ |
| hope: | $\langle\langle 2,1\rangle\rangle$ | profession: | $\langle\langle 4,1\rangle\rangle$ |
| incessantly: | $\langle\langle 4,1\rangle\rangle$ | reason: | $\langle\langle 4,1\rangle\rangle$ |
| labors: | $\langle\langle 4,1\rangle\rangle$ | unfair: | $\langle\langle 1,1\rangle\rangle$ |

*Table 2.3: Example of a stopped inverted index: only non-stopped terms have an entry in the index. As before, for each term there is an inverted list that consists of one or more entries each containing the document identifier and the number of occurrences of the term in that particular document.*

Although the size of an index is typically only a fraction of the collection at hand, this can be further reduced through compression techniques. Various techniques have been used, including integer coding schemes, such as Golomb coding (Golomb, 1966) or Elias coding (Elias, 1975). Originally, compression was used primarily to reduce the space required for an index. However, by using different coding schemes, it has since been possible to also reduce query processing time by reducing disk access times (see for instance Bell et al., 1995, or Scholer et al., 2002). The significance of compression is likely to increase as the gap widens between processor speeds and disk access times.

### 2.2.1 Stopping

*Stopping* is the process of removing frequently occurring terms from indexes and queries (Witten et al., 1999, pages 147–149). The justification for this process is that terms that occur in most documents are not very useful for identifying relevant documents. Although those *stopwords* have a grammatical function and are important for comprehension of sentences, they are of little use in discriminating some documents from others. For example, the word “the” occurs in most documents. If “the” was used as part of a query, it would not have a significant impact on the answer set, if any at all.

Stopwords include articles, prepositions, and conjunctions; a *stoplist* may contain 300–500 terms. We distinguish *stopping* an index from stopping a query: in the first case, inverted lists of stopwords are purged from the index prior to querying time, whereas, in the latter, stopwords are removed from the query during querying time. Stopping has two main advantages: first, the index size is reduced by a small percentage, resulting in decreased storage requirements. Second, during query evaluation, the inverted lists for stopwords, which are usually longer than average, need not be processed, which can lead to a considerable time saving.

There are drawbacks to stopping, however: it is difficult to predict exactly which terms will not be of interest to current and future searchers. For example, it is feasible that a linguist might be interested in ordering documents by stopwords alone, so as to obtain a subset of documents that is interesting in terms of word usage characteristics. Another disadvantage is that queries which contain only stopwords, such as "to be or not to be" or "the who", cannot be serviced with a stopped index.

Because of these problems, and because the savings in index size due to improved compression techniques are typically small, in modern search engines the indexes are not usually stopped. A more common alternative is to stop queries, unless – based on some heuristics or as indicated explicitly by the user – the inclusion of those terms is required to in answer a query.

The stopped index shown in Table 2.3 is considerably smaller compared to the original, non-stopped index shown in Table 2.2. However, in a large scale system, the effect of stopping would not be as great as in this example.

### 2.2.2 Case folding

Another technique used at indexing time is *case folding*, that is to change all upper case letters into lower case letters or vice versa. This is motivated by the fact that users searching for documents that contain the term "osteoporosis" are most likely also interested in documents that contain "Osteoporosis".

### 2.2.3 Stemming

*Stemming* is a technique which removes suffixes (Porter, 1980, Lovins, 1968, Frakes, 1992) from terms in order to reduce them to a common stem form. In languages other than English, stemming might also remove prefixes or infixes, as for instance is needed for Bahasa Indonesia (Asian et al., 2005), or Bahasa Malay (Ahmad et al., 1996). Stemming typically removes gerunds ("ing"), plurals, and past tenses.

In the running example, where the case was already folded, terms are shown stemmed in Table 2.4. This means that the terms that previously had two or more entries (in this case "medicine" and "medicines") only have one entry after having been stemmed to the same root. The lists of those terms are then merged to build a new inverted list for the root.

## 2.3 Relevance

The purpose of ranking algorithms (some of which we discuss in the next section) is to present the user with documents that address their information need. A document is deemed to be *relevant* if it in some way satisfies a user's information need. This does not necessarily mean that the document is useful to a user. As a simple example, if there are two identical documents, both would be considered to be relevant,

| term | inverted list | term | inverted list |
|---|---|---|---|
| destr: | $\langle\langle 4,1\rangle\rangle$ | medicin: | $\langle\langle 2,1\rangle\ \langle 3,1\rangle\ \langle 4,1\rangle\rangle$ |
| die: | $\langle\langle 3,1\rangle\rangle$ | man: | $\langle\langle 3,1\rangle\rangle$ |
| diseas: | $\langle\langle 1,1\rangle\ \langle 3,1\rangle\rangle$ | miserabl: | $\langle\langle 2,1\rangle\rangle$ |
| existen: | $\langle\langle 4,1\rangle\rangle$ | near: | $\langle\langle 3,1\rangle\rangle$ |
| health: | $\langle\langle 1,1\rangle\rangle$ | profes: | $\langle\langle 4,1\rangle\rangle$ |
| hope: | $\langle\langle 2,1\rangle\rangle$ | reason: | $\langle\langle 4,1\rangle\rangle$ |
| incess: | $\langle\langle 4,1\rangle\rangle$ | unfair: | $\langle\langle 1,1\rangle\rangle$ |
| labor: | $\langle\langle 4,1\rangle\rangle$ | | |

*Table 2.4: Example of a stopped and stemmed inverted index; only non-stopped terms have an entry in the index. As before, for each term $t$ there is an inverted list that consists of one or more entries each containing the identifier of document $d$ and the number of occurrences $f_{d,t}$ of $t$ in that particular document.*

but in most cases only one of them will be beneficial to the user in resolving their information need – there is no use to see the same document twice. The less often used *utility* tries to capture how useful a document is not only to satisfy a particular information need, but also in respect to a certain retrieval task (see for instance Brajnik et al., 1996 or Cooper, 1973). These are similar considerations as those examined in Section 2.1.

Saracevic (1999) proposes a hierarchy of relevance. On the lowest level, *system or algorithmic relevance* is achieved if a document satisfies a query on a syntactic level (for instance, there is an overlap of terminology). More grades of relevance (*topical or subject relevance*, *cognitive relevance or pertinence*, *situational relevance or utility*) determine ever stronger user satisfaction, while *motivational or affective relevance* completely addresses the user's information need.

Although often challenged and discussed (see for instance Eisenberg and Hu, 1987, Janes, 1991, Kekäläinen and Järvelin, 2002), in information retrieval the simplification of *binary relevance* is commonly made, where a document can only be either relevant or not relevant to an information need. However, there are many exceptions, such as the field of XML retrieval, where a finer granularity of relevance is typically used (Fuhr et al., 2004, Kazai et al., 2004).

Relevance is discussed in greater detail in Saracevic (1975), van Rijsbergen, 1979, Chapter 7, Schamber (1994), and Mizzaro (1997).

## 2.4 Query evaluation and ranking

Having constructed an index on a document collection, queries need to be matched to documents and a list of answers returned. A user query is usually – at least in the context of a text search engine – a string of characters, typically comprised of two or three terms. There are two main types of queries: *Boolean queries* and *ranked queries*.

Boolean queries are answered by following Boolean logic to determine which documents should be listed in response to a query. The basic Boolean logic operators are OR, AND, and NOT. Each pair of terms in a *Boolean query* is separated by the OR or AND operator, or a term is preceded by the unary operator NOT (Witten et al., 1999, pages 153 and 154). Each *conjunct* in turn can be connected with a further Boolean operator to another term, pair of terms, or other, more complex, nested Boolean constructs.

An example of a Boolean query is: "osteoporosis AND multiple AND (fractures OR (broken AND bones))". The answer set for this query would contain documents that include the terms "osteoporosis", "multiple" and either the term "fractures" or both of the terms "broken" and "bones".

Boolean queries were used with early retrieval engines, but lost popularity in favour of other evaluation schemes; however, most modern search engines still support the use of Boolean operators as part of ranked queries.

Ranked queries can be written in natural language and result in answer sets where documents are ranked by some similarity measure of how alike a document and the query are. The term "ranked query" is somewhat of a misnomer as it is the documents in the result set that are ranked, rather than the query itself. An example of a ranked query is: "osteoporosis multiple fractures". Typically no Boolean operators are used, however most search engines connect terms with an implicit AND or OR. That is, either all documents in the answer set contain all query terms (AND), or returned documents contain at least one of the terms in the query (OR).

A phrase query (Bahle et al., 2002) is a special type of a Boolean AND query, where terms must not only occur within the same document, but consecutively in the same order as stated in the query – that is, they must occur as a phrase. Phrase queries are commonly used in conjunction with ranked queries. Using the earlier example, one might search for the phrase "multiple fractures" in the context of osteoporosis and use the ranked query "osteoporosis 'multiple fractures"'.

Upon submission of a query by the user to the search system, the query is parsed into different terms, which may then be case-folded, stemmed, and stopped – depending on the search system. Once tokenised, the inverted lists of the query terms are retrieved from disk, if they are not cached from a previous access. An *accumulator table*, where document scores are entered, is cleared. The first inverted list is then traversed and the document number as well as the term count (or individual term offsets, if these are used) are decoded, as inverted lists are typically stored in compressed form (see for instance

Scholer et al., 2002). For each document a separately stored document map is consulted, where other document specific information – such as the document length or the document weight – is stored. Relying on the term counts and document weights, a partial score is calculated for the current term/document pair using one of the similarity metrics described in the remainder of this section. If this particular document does not already have an entry in the accumulator table, an entry is created and the document score is initialised with the previously calculated partial score. If, however, an entry already exists for the current document, it is updated by adding the new partial score to the so far accumulated score. This process is repeated until either all inverted lists have been traversed, or until certain conditions, which are dictated by efficiency considerations, have been met upon which processing is altered or stopped completely (see Section 2.4.4 for more detail). Once all lists have been processed, the documents that have the highest entries in the accumulator table are partially sorted in descending order by their accumulated score, typically using a *heapsort* (Knuth, 1978, Volume 3, pages 144–148). Finally, the top documents (say 10) are returned to the user for viewing.

Boolean queries need to be evaluated differently. To evaluate a Boolean query, the inverted lists of each query term are examined. For each term that the query specifies to be present in a document, the document identifiers are added to a pool of possible answers. Conversely, for each term that is specified not to be in a document, the respective document identifier is expressively excluded from the pool of possible answers. For more complex conjuncts that specify for instance that a particular term is present while another one is absent, the lists for both of these terms are processed in parallel to identify matches which are then added to or excluded from the pool. Once all terms have been processed, all documents remaining in the pool are presented to the user.

All documents in the answer pool are assumed equally likely to be relevant. Unlike for ranked queries, answers are not sorted in any particular order when presented to the user.

In the following section we describe and derive different similarity metrics.

### 2.4.1 Vector space model

The framework of the vector space model (see also Witten et al., 1999, page 187) employs a ranking algorithm that tries to rank documents in order of how much of an overlap there is between the terminology of the query and each document (Salton, 1971, Bookstein, 1982), where relatively rare terms have a comparatively high weight.

In the vector space model all queries and documents are represented as vectors in $|V|$-dimensional space, where $V$ is the set of all distinct terms in the collection (the *vocabulary*). Conceptually, documents are ranked by the magnitude of the angle between the document vector and the query vector. Mathematical notation is summarised in Table 2.5.

| | |
|---|---|
| $q$ | A query |
| $\vec{q}$ | The query vector of query $q$ |
| $\|\vec{q}\|$ | The vector length of the query $q$ |
| $d$ | A particular document |
| $\|d\|$ | The length of document $d$ in some suitable unit |
| $\vec{d}$ | The document vector of document $d$ |
| $\|\vec{d}\|$ | The vector length of the document $d$ |
| $t$ | A particular term |
| $V$ | The set of distinct terms in the vocabulary |
| $\|V\|$ | The total number of distinct terms in the vocabulary |
| $N$ | The number of all documents in the collection |
| $w_{q,t}$ | The weight of a particular term in the query |
| $w_{d,t}$ | The weight for a particular term in a particular document |
| $f_t$ | The number of documents term $t$ appears in |
| $f_{d,t}$ | The number of occurrences of term $t$ within document $d$ |

*Table 2.5: Summary of notation for the vector space model.*

At the centre of the vector space model is the *cosine measure*, which is used to measure the angle between two vectors. The cosine between two vectors is determined as the dot product between each document vector $\vec{d}$ and the query vector $\vec{q}$, normalised by the lengths of the document and the query:

$$cosine(q,d) = \frac{\vec{q} \cdot \vec{d}}{|\vec{q}||\vec{d}|}$$

Since the contribution of any term not in both the document and the query will be 0, it is only necessary to calculate the dot product for the terms in the intersection. Using query term weights $w_{q,t}$ and document term weights $w_{d,t}$ and applying the dot product for the intersection of all terms $t$ occurring in the query and the document currently being examined gives:

$$cosine(q,d) = \frac{\sum\limits_{t \in q \cap d} (w_{q,t} \times w_{d,t})}{\sqrt{\sum\limits_{t \in q} w_{q,t}^2} \times \sqrt{\sum\limits_{t \in d} w_{d,t}^2}} \qquad (2.1)$$

The vector space model does not specify how to set the document term weight and the query term weight, but in practice these weights are often calculated using their collection frequency and within-document frequency, respectively:

$$w_{q,t} = \ln\left(1 + \frac{N}{f_t}\right) \tag{2.2}$$

and:

$$w_{d,t} = 1 + \ln(f_{d,t}) \tag{2.3}$$

where $N$ is the number of documents in the collection, $f_t$ is the number of documents term $t$ occurs in, and $f_{d,t}$ is the number of times the term appears in document $d$. The first equation is also referred to as the *term frequency* or *inverse document frequency* and gives a measure of how common or rare a term is in a collection, and the latter is also known as the *document frequency*, providing a measure of the emphasis a term is given within a document. Both measures are discussed in further detail in Section 3.3.2.

Since the query length is constant for the evaluation of a single query, we can ignore this factor, while preserving ranking order. Finally, dropping this factor and substituting the weights for each component as defined in Equations 2.2 and 2.3 into Equation 2.1, yields:

$$cosine(q,d) \overset{rank}{=} \frac{\sum\limits_{t \in q \cap d} \left(\ln\left(1 + \frac{N}{f_t}\right) \times (1 + \ln(f_{d,t}))\right)}{\sqrt{\sum\limits_{t \in d} (1 + \ln(f_{d,t}))^2}}$$

To evaluate a query, all documents are ranked by their similarity score.

The vector space model continues to be used in a variety of information retrieval areas apart from document retrieval, such as document categorisation (Joachims, 1997, Hull, 1994), collaborative filtering (Soboroff and Nicholas, 2000) or topic tracking (Connell et al., 2004).

### 2.4.2 Probabilistic retrieval models

Probabilistic retrieval models are used to estimate the probability of documents being relevant to a query (Robertson and Sparck Jones, 1976, Robertson et al., 1981). This probability is then used to rank all documents in response to a query (this is also referred to as the *probability ranking principle*).

In the following we derive one of the most commonly used probabilistic models, *Okapi* (Robertson and Walker, 1999), and in particular *Okapi BM25* (Robertson et al., 1992). We follow the explanations given by Sparck Jones et al. (2000) and Robertson et al. (1981). Notation is summarised in Table 2.6.

We assign $L$ to the event that a document is relevant to a query (it is “liked”) and $\overline{L}$ to the event that a document is not relevant to a query. Then $P(L|d)$ is the probability that document $d$ is relevant, and $P(\overline{L}|d)$ that it is not.

| | |
|---|---|
| $q$ | A query |
| $d$ | A particular document |
| $L$ | The event that document $d$ is relevant (liked), given query $q$ |
| $\overline{L}$ | The event that document $d$ is not relevant (not liked), given query $q$ |
| $\lvert d\rvert$ | The length of document $d$ in some suitable unit |
| $t$ | A particular term |
| $A_{i,d}$ | The $i^{th}$ attribute used to describe document $d$ |
| $a_{i,d}$ | The value of $A_{i,d}$ in document $d$ |
| $V$ | The set of all distinct terms in the vocabulary |
| $R$ | The set of all relevant documents, given a particular query |
| $\lvert R\rvert$ | The number of all documents in $R$ |
| $r_t$ | The number of documents in $R$ that contain term $t$ |
| $N$ | The number of all documents in the collection |
| $C$ | The collection, made up of $N$ documents |
| $f_t$ | The number of all documents that contain term $t$ |
| $f_{d,t}$ | The number of occurrences of term $t$ in document $d$ |
| $k_1, b, k_3$ | Tuning constants |

*Table 2.6: Summary of notation for the probabilistic model.*

Let $sim(d, q) = \frac{P(L|d)}{P(\overline{L}|d)}$, that is, the similarity of a particular document $d$ to a query $q$ is defined by the ratio of probabilities of relevance to that of non-relevance. Bayes theorem $\left(P(A|B) = \frac{P(B|A)P(A)}{P(B)}\right)$ (Gelman et al., 2004, pages 7 and 22) allows us to rewrite this similarity:

$$sim(d, q) = \frac{P(d|L)}{P(d|\overline{L})} \times \frac{P(L)}{P(\overline{L})}$$

The ratio $P(L)/P(\overline{L})$ is a constant that is independent of the characteristics of particular documents. Discounting this portion of the equation and using log-odds leads to the simpler, rank-equivalent formulation:

$$sim(d, q) \overset{rank}{=} \log \frac{P(d|L)}{P(d|\overline{L})} \tag{2.4}$$

Sparck Jones et al. introduce the concept of *document attributes*, which can be any features a document may have, such as the structure of a document or images. However, most commonly the document attributes are taken to be the terms in a document. Sparck Jones et al. make the assumption that document attributes are independent of each other. As in this thesis we only consider attributes that are terms, this assumption means that the occurrence of one term in a document is completely independent from another term appearing in this document. This assumption simplifies the mathematics involved greatly,

| | Relevant | Non-relevant | All documents |
|---|---|---|---|
| Documents containing term $t$ | $r_t$ | $f_t - r_t$ | $f_t$ |
| Documents not containing term $t$ | $\|R\| - r_t$ | $N - f_t - \|R\| + r_t$ | $N - f_t$ |
| All documents | $\|R\|$ | $N - R$ | $N$ |

*Table 2.7: Incident contingency table: collection and term statistics are divided into sets of relevant and non-relevant documents. Although $|R|$ and $r_t$ are dependent on an information need, for simplicity, we do not introduce another subscript, but rather treat this dependency as implicit.*

although it does not hold in practice – consider for instance the common co-occurrence of terms such as “New” and “York” or “United” and “States”. Sparck Jones et al. (2000) refer to van Rijsbergen (1977), who reports that taking term interdependencies into account has little impact on any resultant ranking.

It follows that the probability of a document being relevant can be calculated by the product over all $i$ attributes $A_{i,d}$ that describe document $d$ (or, in the simplified case, terms in the vocabulary), where $a_{i,d}$ indicates the value of that attribute (whether the term is present or absent):

$$P(d|L) = \prod_i P(A_{i,d} = a_{i,d}|L) \tag{2.5}$$

Analogously for non-relevant documents:

$$P(d|\overline{L}) = \prod_i P(A_{i,d} = a_{i,d}|\overline{L}) \tag{2.6}$$

Substituting Equations 2.5 and 2.6 into 2.4:

$$sim(d,q) \overset{rank}{=} \sum_i \log \frac{P(A_{i,d} = a_{i,d}|L)}{P(A_{i,d} = a_{i,d}|\overline{L})}$$

Furthermore, the sum of the ratio of all probabilities of liked to unliked non-present attributes can be discounted without changing the ordering of the similarities:

$$sim(d,q) \overset{rank}{=} \sum_i \log \frac{P(A_{i,d} = a_{i,d}|L)}{P(A_{i,d} = a_{i,d}|\overline{L})} - \sum_i \log \frac{P(A_{i,d} = 0|L)}{P(A_{i,d} = 0|\overline{L})}$$

$$sim(d,q) \overset{rank}{=} \sum_i \log \frac{P(A_{i,d} = a_{i,d}|L)P(A_{i,d} = 0|\overline{L})}{P(A_{i,d} = a_{i,d}|\overline{L})P(A_{i,d} = 0|L)} \tag{2.7}$$

After sacrificing the generality of attributes in favour of terms, and re-arranging:

$$sim(d,q) \overset{rank}{=} \sum_{t \in V} \log \frac{P(t \text{ present}|L)(1 - P(t \text{ present}|\overline{L}))}{P(t \text{ present}|\overline{L})(1 - P(t \text{ present}|L))} \tag{2.8}$$

Making use of *incident contingencies*, summarised in Table 2.7, the probability of term $t$ being present in a relevant document can be expressed as the ratio of the number of relevant document that contain that term, $r_t$ to the number of relevant documents $|R|$:

$$P(t \text{ present}|L) = \frac{r_t}{|R|} \tag{2.9}$$

For non-relevant documents, we introduce $N$, the total number of documents in the collection and the *term frequency* $f_t$, the total number of documents that contain term $t$:

$$P(t \text{ present}|\overline{L}) = \frac{f_t - r_t}{N - |R|} \tag{2.10}$$

Substituting Equations 2.9 and 2.10 back into Equation 2.8 gives:

$$sim(d,q) \overset{rank}{=} \sum_{t \in V} \log \frac{r_t(N - f_t - |R| + r_t)}{(|R| - r_t)(f_t - r_t)}$$

The value of $0.5$ is added to $r_t$ in order to avoid problems where no relevant documents contain one of the query terms (for example because of vocabulary mismatch):

$$sim(d,q) \overset{rank}{=} \sum_{t \in V} \log \frac{(r_t + 0.5)(N - f_t - |R| + r_t + 0.5)}{(|R| - r_t + 0.5)(f_t - r_t + 0.5)}$$

However, in an initial ad-hoc setting no prior information about relevance of documents is available, and only term statistics can be made use of. Since neither $|R|$ nor $r_t$ are known, they are set to $0$, which reduces the term weight to:

$$sim(d,q) \overset{rank}{=} \sum_{t \in V} \log \frac{N - f_t + 0.5}{f_t + 0.5} \tag{2.11}$$

**Term frequency weighting**

So far it was not specified what the $A_t = a_{t,d}$ in Equations 2.5 to 2.7 is, but one solution would be to use the within-document term frequency $f_{d,t}$. Robertson and Walker (1994) examine the assumption that so-called *elite* terms follow a different term distribution to more common terms, where elite terms are effectively those terms that are good discriminators between documents. They employ a 2-Poisson distribution, associating one distribution with elite terms, and another with all other terms. The assumption that one set of terms follows a certain probability distribution whereas another set follows a different one has been underlying information retrieval concepts for some time; Harter (1975) uses this assumption in order to find suitable keywords as index terms of a sparse index. More recently an n-Poisson model was proposed and tested (Margulis, 1993). Ponte and Croft (1998) review and critique the 2-Poisson model.

Although Sparck Jones et al. (2000) consider the 2-Poisson model as a good match for their approach, it is not suited to practical application. An estimation of the 2-Poisson model is therefore used instead, based on the within document term frequency $f_{d,t}$, giving a count of the number of occurrences of a particular term $t$ in a particular document $d$ (Robertson and Walker, 1994). The 2-Poisson estimation equation $((k_1 + 1)f_{d,t})/(k_1 + f_{d,t})$ inserted into Equation 2.11, and reducing the space over which terms this equation is applied to from all terms in the vocabulary to those appearing in the query, results in:

$$sim'(d, q) = \sum_{t \in q} \log \left( \frac{N - f_t + 0.5}{f_t + 0.5} \right) \times \frac{(k_1 + 1)f_{d,t}}{k_1 + f_{d,t}} \tag{2.12}$$

The first of several tuning constants present in the final Okapi BM25 equation is $k_1$. If it is set to $0$, the weight is reduced to the simple term presence, meaning the weight of a term is not increased by the number of times it appears in a document. If it is set to a large value, the term weight increases roughly linearly with the term frequency. In our experiments we use the value of $1.2$, recommended by Robertson and Walker (1999); this is at the lower end of the range suggested by Sparck Jones et al. (2000).

### Document length normalisation

As document length normalisation and the number of re-occurring query terms are of importance to the experiments in Chapter 6, we discuss these here in detail. Since long documents have a greater likelihood of containing more distinct terms than short documents, long documents have a greater chance to be highly ranked for any query, even though the information conveyed in those documents may be of a different kind to that sought by the query. A factor, that is linked to the lengths of documents, normalises the contribution of term weights to the matching for documents of different lengths. The length $|d|$ of document $d$ is calculated in some suitable measure, for instance the number of terms, the number of non-stopped terms, or the number of bytes. In the Okapi BM25 similarity measure, document length normalisation is given by:

$$(1 - b) + b \times \frac{|d| \times N}{\sum_{i \in C} |d_i|}$$

A tuning constant, $b$, is used to control the impact of the normalisation factor. If $b$ is set to $0$, no normalisation is performed, while if it is set to $1$, the full impact of the length normalisation is used. In our implementation we use the value $0.75$, as recommended by Robertson and Walker (1999) and Sparck Jones et al. (2000).

Combining the document normalisation with Equation 2.12 results in:

$$sim''(d, q) = \sum_{t \in q} \log \left( \frac{N - f_t + 0.5}{f_t + 0.5} \right) \times \frac{(k_1 + 1)f_{d,t}}{k_1 \times \left( (1 - b) + b \times \frac{|d| \times N}{\sum_{i \in N} |d_i|} \right) + f_{d,t}} \tag{2.13}$$

Document length normalisation has also been discussed by Singhal et al. (1996).

### Re-occurring query terms

It is possible that long queries contain some query terms more than once. Without the use of natural language processing or other more intelligent approaches to understanding a query than just as a bag-of-words, this duplication could be taken as emphasising the importance of these terms. The query term frequency $f_{q,t}$ is handled analogously to the within document frequency which was shown earlier. Adding into Equation 2.13 a query term frequency formulation results in the full Okapi BM25 measure:

$$BM25(d,q) = \sum_{t \in q} \log \left( \frac{N - f_t + 0.5}{f_t + 0.5} \right) \times \frac{(k_1 + 1) f_{d,t}}{k_1 \times \left( (1-b) + b \times \frac{|d| \times N}{\sum_{i \in N} |d_i|} \right) + f_{d,t}} \times \frac{(k_3 + 1) f_{q,t}}{k_3 + f_{q,t}} \tag{2.14}$$

where $k_3$ is a tuning constant that can range from $0$, where only one instance of each query term contributes to the ranking, to a very large value, such as 10,000, where query terms contribute as often as they occur. In our implementation we set $k_3$ to a very large constant. In practice, $k_3$ has little impact on most of the experiments detailed in this thesis, since we use TREC title only queries that are typically short and do not contain any duplicate terms (see Section 2.6.2 for details). An exception is our work on document expansion, detailed in Chapter 7.

### Okapi BM25

The complex Poisson function is estimated by trial and error. The name Okapi BM25 is derived of “BM”, which is the abbreviation of “Best Match”, and a version number of the latest trial, in this case 25, which is a combination of BM11 and BM15 (Robertson et al., 1994).

In summary, the first term of the BM25 equation shown in Equation 2.14 above de-emphasises the importance of query terms that appear in many documents in the collection. The second term gives documents that contain query terms in larger numbers a higher weight and normalises by document length. The third term is used for tuning the impact of duplicated query terms.

The Okapi framework theoretically does not allow for query expansion since it implicitly postulates that terms that do not occur in the user query have the same likelihood of occurring in relevant and non-relevant documents (Lavrenko, 2004, page 11). Therefore, adding expansion terms to the query should not have any statistically significant effect. This, of course, has been proven wrong in many query expansion experiments including those detailed in this thesis.

## 2.4.3 Language models

An alternative way of ranking documents has been put forward by Ponte and Croft (1998). In the language modelling approach, queries and documents are in principle generated from a *language model*.

Here, documents are not seen as a string of terms put together by an author, but rather as one – albeit the most likely one – of many possible manifestations of the information that the author wanted to convey. In other words, a language model for a particular document would not only be based on the usage of terms and sentence structure that appear in that document, but the document would be seen as a sample of what the document could also look like, while still conveying the same information. One can understand this as a document being just one particular version that has been chosen from an unlimited

| | |
|---|---|
| $q$ | A query |
| $\|q\|$ | The length of query $q$, expressed in the number of term occurrences |
| $d$ | A particular document |
| $\|d\|$ | The length of document $d$, expressed in the number of term occurrences |
| $\theta_d$ | The language model of document $d$ |
| $\theta_{q,t}$ | The distribution of terms given the query $q$ and the term $t$ |
| $\theta_{d,t}$ | The distribution of terms given the document $d$ and the term $t$ |
| $t$ | A particular term |
| $f_{d,t}$ | The number of occurrences of term $t$ in document $d$ |
| $f_{q,t}$ | The number of occurrences of term $t$ in query $q$ |
| $C$ | The collection, made up of $N$ documents |
| $F_t$ | The total number of occurrences of term $t$ in the collection |
| $F$ | The total number of occurrences of all terms in the collection |
| $V$ | The set of distinct terms in the vocabulary |
| $\|V\|$ | The number of distinct terms in the vocabulary |
| $\lambda$ | Jelinek-Mercer smoothing parameter |
| $\alpha_d$ | Document dependent smoothing parameter |
| $\mu$ | Dirichlet smoothing parameter |
| $R$ | The set of relevant documents, given an information need |
| $R'$ | A set of documents *assumed* to be relevant |

*Table 2.8: Summary of notation for language models.*

number of drafts to convey the information that the document contains. Intuitively, the basis for this is sensible, as someone might have used slightly different terms or a different sentence structure and still authored a document with the same information as the one at hand. In practice, due to the lack of information about the actual language model of a document, the model is *inferred* from actual term distributions of the document under examination and those of a wider collection of documents (such as the web or a particular collection of newswire documents).

Conceptually, to rank documents, the most likely relations between documents and queries are established. There are three main approaches; the first assumes that a query was *generated* by a document model. The second reverses this and assigns probabilities to documents in order of their likelihood of having been generated by the model of a query. The last one compares the models of both the query and each document. Each of those is explained in detail in the following sections.

Language models have also been used for parsing documents and *part of speech* tagging, in general (Jelinek, 1990). Similarly as with speech recognition, language models can help to solve the problem whether the next item in a stream of text is more likely to be for instance a noun or a verb. Since their introduction, language models have shown good results, often better than Okapi BM25 (see for instance Lavrenko and Croft (2001)). For a collection of pertinent papers on language modelling, see Croft and Lafferty (2003).

**Origins of language models**

Language models have previously been used in applications such as speech recognition (De Mori and Brugnara, 1996), where it is often difficult to distinguish similar sounding phonemes. As an example, when sung, the first line of the Australian National Anthem "Australians all let us rejoice" is often miss-understood as "Australians all are ostriches", or the title of a famous book "To kill a Mockingbird" could be misunderstood as "Tequila Mockingbird". Using language models, each of the different options derived from acoustics are given a probability of occurrence. Mixing the degree of certainty that a particular phoneme has been accurately recognised and the probability of the next character or word occurring given the previous terms, leads to higher recognition accuracy.

More generally, this idea can also be used on a word level, where each atom is a term, rather than a character. In particular, the probability of the next term occurring can be based on previous term occurrences. These dependencies are also called *n-grams* (Chen and Goodman, 1996). The probability of a string $s$ of length $l$ that consists of terms $t_1$ to $t_l$ is given by:

$$P(s) = P(t_1)P(t_2|t_1)P(t_3|t_1t_2)...P(t_l|t_1t_2...t_{l-1}) = \prod_{i=1}^{l} P(t_i|t_1...t_{i-1})$$

We briefly examine *bigram models*, which approximate n-grams in that only the immediately preceding term has an influence on the identity of the next term:

$$P(s) \approx P(t_1)P(t_2|t_1)P(t_3|t_2)...P(t_l|t_{l-1}) = \prod_{i=1}^{l} P(t_i|t_{i-1})$$

Unlike the application of language modelling for information retrieval, where a language model is derived for each document, in speech recognition (or the use of n-grams more generally), a model for the whole collection is established. In order to estimate the probabilities for each term pair $P(t_{i-1}|t_i)$ we can use the *maximum likelihood estimator* (denoted $\hat{P}$), where the number of occurrences (the *count*) of a particular pair are compared to the total number of pairs in the collection (in order to normalise occurrences):

$$\hat{P}(t_i|t_{i-1}) = \frac{count(t_{i-1}t_i)}{\sum_j count(t_{j-1}t_j)}$$

A full derivation of the maximum likelihood estimator is given by Bilmes (1998), who shows that this estimator will arrive at the best possible estimate, if only a document is given but no collection. The derivation is based on the *EM step* algorithm (Wu, 1983, Dempster et al., 1977).

Using the running example shown in Table 2.1 as training data for the probabilities, we can estimate the probability of the string $s_1$ “the only profession” occurring (ignoring case and not applying stemming or stopping) as:

$$P(s_1) \approx \frac{3}{44} \times \frac{1}{3} \times 1 = \frac{1}{44} = 2.3\%$$

The calculation above can be explained as following: “the” occurs three times in the example collection which consists of a total of 44 tokens, and “only” is only preceded once by “the” in three places where it occurs. “Profession” occurs once only, straight after the term “only”. The string “the only one” has the same probability using bigrams, but would have a 0 probability if trigrams (or higher order models) were used, since this string does not occur anywhere in the training data as a complete sequence of terms.

Even though the string “the only medicine” seems like a reasonable string in English, it has a zero probability of occurring according to this model, as “medicine” is never preceded by “only” in the example collection. The problem of completely discounting a particular string – no matter how likely it might seem to an observer who has greater knowledge of term distributions than just those provided by the training data – stems from the relatively limited sample space that is available to observe probabilities from. This difficulty can be overcome by the use of *smoothing*, which prevents a missing n-gram from resulting in a maximum likelihood estimator of zero. We introduce smoothing in the next section and explain it in detail later on.

### Query likelihood approach

In the most common language modelling approach for information retrieval, the *query likelihood model*, documents are ranked against a query by the probability that the language model of a document has *generated* this query. This terminology can be slightly misleading, in that under typical retrieval circumstances queries are posed by a user and not generated by a document model, however the idea is that a query identical to the one posed by the user could have been generated by a document model.

We develop a basic mathematical model of the query likelihood model following the approach used by Nallapati et al. (2003). As the basis of this approach, we calculate the probability of a query occurring, given the language model of a particular document. The resulting probability will be used to rank documents in the collection against the query. The likelihood of a query $q$ having been generated by the model of a particular document $d$ is calculated as the product of the likelihoods of each query term $t$ being part of the language model $\theta_d$ of that document:

$$P(q = t_1 ... t_n | \theta_d) = \prod_{t \in q} P(t|\theta_d) \qquad (2.15)$$

where terms are assumed to be independent of each other, as with the probabilistic model. From Equation 2.15 it can be seen that the traditional query likelihood model (as explained in this thesis) makes use of unigram models only (that is, terms are not dependent on context), even though higher order models can be more powerful.

In theory, correct probabilities for $P(t|\theta_d)$ would be determined by randomly sampling terms from the document's language model until the probabilities for all terms converge. Since this is not practical (we do not have access to the actual language model of a document), the maximum likelihood estimator is used instead. We use $P'(t|\theta_d)$ instead of $P(t|\theta_d)$ to make clear that this is only a first approximation; a better estimate is formulated later. The maximum likelihood estimator assigns the highest possible likelihood to observed distributions, which in this case means using the within-document frequency $f_{d,t}$ of terms:

$$P'(t|\theta_d) = \hat{P}(t|d) = \frac{f_{d,t}}{|d|} \tag{2.16}$$

where $|d|$ is the number of term occurrences in document $d$. Note the difference between $\theta_d$ and $d$; the first refers to the language model of document $d$, the latter treats a document just as a bag-of-words.

There is a problem with this formulation. When given a collection, only one concrete instance of a language model, namely the document, is available. Therefore so the language model for a particular document cannot be derived from the larger theoretical sample space. Consider the query "who has only hope, but no pharmaceutical?" and Document 2 from the example collection shown in Table 2.1: "The miserable have no other medicine but only hope." After stopping the query becomes "hope pharmaceutical", for which Document 2 is a good answer, however the terms "pharmaceutical" and "medicine" are mismatched. From a language modelling point of view, the word "pharmaceutical" would most likely have been in another draft of the same document, however the maximum likelihood estimator for the term "pharmaceutical" results in the value of $0$ for Document 2 as the within-document term frequency is $0$ (this problem is also referred to as the *zero-frequency problem*). Using $P'(t|\theta_d)$ as shown in Equation 2.16 has the effect that the query likelihood calculated in Equation 2.15 is forced to $0$ because of the one missing term and, in the given example, Document 2 would get a $0$ score against the query. That is, it has a 0 probability of being relevant, according to the model.

This problem can be addressed by assigning a small but non-zero probability to terms that do not occur in a document. A technique called *smoothing* does this and is described in more detail in Section 2.4.3. Although no other drafts of a particular document might be available, this missing "hidden" part of the language model of the document can be estimated by statistical analysis of the collection as a whole, in the hope that this will compensate for the "hidden" part. A parameter $\lambda$ determines how much probability space is allocated to terms that do not occur. Through this method, the document is effectively blended in with the rest of the collection. Zero-probability effects of non-occurring terms are therefore removed by allocating this extra probability to those terms. A side-effect of this is that terms which actually do occur in a document are given a distribution of probabilities that is closer to what one can expect to find given a particular collection. The term distribution is closer to the distribution of terms in the collection as a whole. A more accurate representation of the probability $P(t|\theta_d)$ using a simple form of smoothing is therefore:

$$P(t|\theta_d) = (1-\lambda)\hat{P}(t|d) + \lambda\hat{P}(t|C) = (1-\lambda)\frac{f_{d,t}}{|d|} + \lambda\frac{F_t}{F} \tag{2.17}$$

where $F_t$ is the total number of occurrences of term $t$ in the collection $C$, which contains a total of $F$ token occurrences.

Re-writing the query likelihood equation for $P(q|\theta_d)$ given in Equation 2.15 using the estimate given in Equation 2.17:

$$P(q|\theta_d) = \prod_{t\in q}\Big((1-\lambda)\hat{P}(t|d) + \lambda\hat{P}(t|C)\Big) = \prod_{t\in q}\left((1-\lambda)\frac{f_{d,t}}{|d|} + \lambda\frac{F_t}{F}\right) \tag{2.18}$$

The version of smoothing used in Equation 2.18 above leads to the *mixture model*, as the term statistics of both the documents and the collection are linearly interpolated, or *mixed*, via the parameter $\lambda$. Zhai and Lafferty (2002) found that a wide range of values, typically between $0.2$ and $0.8$, achieve good effectiveness, depending on collections and the length of queries.

The approach detailed above is a version of the *query likelihood model*. Versions of this model are most commonly used in information retrieval systems that make use of language models, since they are reliable and relatively easy to implement, as well as relatively efficient during operation (it is comparable to the query evaluation time needed by the probabilistic model, described earlier).

The equation given in Equation 2.18 is an example of a *multinomial* approach to language modelling, whereas the original language modelling framework as proposed by Ponte and Croft (1998) was based on *multiple-Bernoulli* distributions. This made the approach to language modelling in the field of information retrieval less powerful compared to the use of language modelling in areas such as machine learning or speech recognition, in that the term order of neither query terms nor terms in documents is taken into account. In the multiple-Bernoulli models only the information content – or *entropy* (Shannon, 2001) – is used for analysis. However, the more popular multinomial models (Metzler et al., 2004) see queries and documents as a sequence of terms and thereby do take term order into account in the form of n-grams, introduced in Section 2.4.3. As these approaches bring with them a high level of implementation complexity, unigrams are typically used, even though bigrams have been used successfully in the past (Song and Croft, 1999). This again means that language models in information retrieval are commonly using the benefits that language modelling can offer to a lesser extent than theoretically possible. Also, since the original framework considers probability distributions over a binary event space, only the first occurrence of query terms was taken into consideration. Multinomial models however do take re-occurrences of terms into consideration.

**Similarities with *tf.idf* ranking**

In the following we re-formulate Equation 2.18 in order to develop different smoothing methods and show how language models are similar to the *tf.idf* weighting of the vector space model and the Okapi formula. We first break up the formulation into two parts, one that covers terms that occur in both the query and the document, and a second that contains the remaining terms of the query ($t \in (q - d)$, that is, the set of terms that appear in the query but not in the document).

$$\begin{aligned} P(q|\theta_d) &= \prod_{t \in q} ((1-\lambda)P(t|d) + \lambda P(t|C)) \\ &= \prod_{t \in q \cap d} ((1-\lambda)P(t|d) + \lambda P(t|C)) \times \prod_{t \in q-d} (\lambda P(t|C)) \end{aligned}$$

As before, substituting with $P(t|d) = \frac{f_{d,t}}{|d|}$ and $P(t|C) = \frac{F_t}{F}$ gives:

$$P(q|\theta_d) = \prod_{t \in q \cap d} \left( (1-\lambda)\frac{f_{d,t}}{|d|} + \lambda \frac{F_t}{F} \right) \times \prod_{t \in q-d} \left( \lambda \frac{F_t}{F} \right)$$

The set for the second part of the equation can be simplified as follows:

$$\begin{aligned} P(q|\theta_d) &= \prod_{t \in q \cap d} \left( (1-\lambda)\frac{f_{d,t}}{|d|} + \lambda \frac{F_t}{F} \right) \times \prod_{t \in q-d} \left( \lambda \frac{F_t}{F} \right) \times \prod_{t \in q \cap d} \left( \frac{\lambda \frac{F_t}{F}}{\lambda \frac{F_t}{F}} \right) \\ &= \prod_{t \in q \cap d} \left( \frac{\lambda \frac{F_t}{F} (1-\lambda) \frac{f_{d,t}}{|d|}}{\lambda \frac{F_t}{F}} + \lambda \frac{F_t}{F} \right) \times \prod_{t \in q \cap d} \frac{1}{\lambda \frac{F_t}{F}} \times \prod_{t \in q-d} \left( \lambda \frac{F_t}{F} \right) \times \prod_{t \in q \cap d} \left( \lambda \frac{F_t}{F} \right) \\ &= \prod_{t \in q \cap d} \left( \frac{(1-\lambda)\frac{f_{d,t}}{|d|}}{\lambda \frac{F_t}{F}} + 1 \right) \times \lambda^{|q|} \times \prod_{t \in q} \left( \frac{F_t}{F} \right) \end{aligned}$$

Taking logs on the right-hand side preserves the ranking (it is *rank equivalent*) and finally lets us re-write the equation in a way that provides further insight into the query likelihood model:

$$P(q|\theta_d) \overset{rank}{=} \sum_{t \in q \cap d} \log \left( \frac{(1-\lambda)\frac{f_{d,t}}{|d|}}{\lambda \frac{F_t}{F}} + 1 \right) + |q| \log \lambda + \sum_{t \in q} \log \frac{F_t}{F}$$

The final term in this equation is constant for a given query and can be discounted, while maintaining rank equivalence:

$$P(q|\theta_d) \overset{rank}{=} \sum_{t \in q \cap d} \log \left( \frac{1-\lambda}{\lambda} \frac{f_{d,t}}{|d|} \frac{F}{F_t} + 1 \right) + |q| \log \lambda \qquad (2.19)$$

In this form, it can be seen that $f_{d,t}/|d|$ acts as a *term frequency* component and $F/F_t$ as an *inverse document frequency* component. The similarity to the probabilistic model (see Section 2.4.2) is apparent. This is in direct contrast to the claim that "language models do not need a *tf.idf* (or any other) term weighting algorithm" (Hiemstra et al., 2004). In fact, out of the vector space model, the probabilistic models and language models, the latter is the only ranking framework that lead to the use of *tf.idf* formulations naturally. In contrast, in the probabilistic model *tf.idf* weighting is used because it is a close fit to the ideal weighting equation, rather than out of mathematical necessity.

This rewriting demonstrates that only terms that appear in both the query and the document currently ranked are considered in Equation 2.19. This makes ranking using the query likelihood model as efficient as Okapi BM25 for instance.

An interesting point is that smoothing seems to be redundant since Equation 2.19 only sums over terms that appear in both the document and the query. Therefore the zero-frequency problem cannot occur. However, smoothing is vital to the derivation of the model; without it, the inverse document frequency component would not arise.

### Smoothing for language models

As mentioned earlier, smoothing is a solution for the *zero-frequency problem*, where documents that do not contain one of the terms in the query can still attain a non-zero probability of being relevant to the query. Smoothing is also important in other areas, such as text compression, where commonly each token is assigned a probability of being the next token on the input. In the text compression domain, the zero-frequency problem occurs when a term that has not been seen before is encountered in the input.

We used the aforementioned mixture model – also known as *Jelinek-Mercer smoothing* (Chen and Goodman, 1996) – for all smoothing so far in our explanation of language models. Liu and Croft (2004) refine this model, by also breaking up the collection into clusters, which then contribute to the language model of each document. A document model therefore is calculated using statistics from the document, the cluster it has been assigned to, and the collection as a whole.

This model helps to clarify the difference between $\prod_{t \in V} P(t|\theta_d)$ and $\prod_{t \in V} P(t|d)$; the first refers to the true probability distributions of the query terms over document $d$, whereas the latter are unsmoothed distributions that are typically derived using a maximum likelihood estimator.

In the previous section, the last portion of Equation 2.19 is $|q| \log \lambda$. This does not depend on any query terms, but only the length of the query. If Jelinek-Mercer smoothing is used, then $\lambda$ does not depend on the document $d$ and is a constant. In this case the last portion of the equation can be ignored for ranking.

An alternative smoothing technique is *Dirichlet smoothing*. Here, the document-dependent constant $\alpha_d$ is used instead of $\lambda$, and is set to $\frac{\mu}{|d|+\mu}$, where $\mu$ is a constant. This value needs to be computed for each document, as $\alpha_d$ is dependent on the number of tokens in document $d$ (Zhai and Lafferty, 2001*b*). Dirichlet smoothing takes the document length into account, with the reasoning that smoothing for large documents – which provide a larger sample space – is necessary to a smaller degree than for short documents. An extension of Dirichlet smoothing is *two-stage smoothing* (Zhai and Lafferty, 2002) where $\alpha_d$ is maximised by removing one term at a time and calculating the smoothing parameter with that term being absent. It has the advantage over Dirichlet smoothing that no tuning is needed and performs about as well; see for instance Chen and Goodman (1996) or Zhai and Lafferty (2004) for a comparison of different approaches to smoothing.

### Document likelihood approach

An alternative to the query likelihood model is the *document likelihood model*. It is an inverse of the query likelihood model; it builds a language model of the query and calculates the likelihood of documents having been generated from this model. The problem with this approach is that queries are often very short – 20% of web queries in 2002 only contained a single term (Jansen et al., 2005). Therefore, due to limited data, the models derived from queries are relatively unreliable, and the document likelihood model tends to perform poorly (Lavrenko, 2004, Table 5.5, page 69). Although better query language models can be derived using different approaches, as detailed in the next section, other language modelling approaches are typically favoured over the document likelihood model. This approach is not considered further in this thesis.

### Model comparison approach

In is the *model comparison* approach, separate language models are developed for the query and each document in the collection. The models are then compared to each other.

At the centre of the model comparison approach is the *Kullback-Leibler divergence* (Kullback, 1959, page 6), introduced to language modelling by Lafferty and Zhai (2001). It has been used to determine the *query clarity* where the language used in a query is evaluated against the language of the corpus against which the query is being ranked (Cronen-Townsend et al., 2002). The Kullback-Leibler Divergence (KLD) compares the language models of a query ($\theta_q$) and a document ($\theta_d$):

$$KLD(\theta_q, \theta_d) = \sum_{t \in V} \left( \theta_{q,t} \log \frac{\theta_{q,t}}{\theta_{d,t}} \right) = \sum_{t \in V} \left( P(t|\theta_q) \log \frac{P(t|\theta_q)}{P(t|\theta_d)} \right)$$

where $V$ is the set of all unique terms in the collection. The Kullback-Leibler divergence can also be seen as the *relative entropy* (Manning and Schütze, 2003, page 72) between the probability distributions of the query language model and the document language model, that is, it is a measure of how many additional bits would be needed if one distribution was encoded given the distribution of the other. If both distributions are very similar, few bits would be used and the result of the formulation above is a small number. If one of the distributions is quite distinct from the other, it would take many bits to encode the difference. Since the Kullback-Leibler divergence gives the dissimilarity between the two sets of probability distributions, documents need to be ranked in inverse order of the resulting value.

The language model for a document can be estimated as for the query likelihood model, however the problem again (as with the document likelihood model) is to estimate a good language model for the query. Analogously to the first estimation of a document model shown in Section 2.4.3, in the simplest approach we could just use the maximum likelihood estimator over the query, which is then treated as a bag-of-words. We label this estimation $P'(t|\theta_q)$ in the following to show that this is only a first approximation. The unsmoothed probability of a term $t$ given query $q$ is then calculated by the query term frequency $f_{q,t}$ and $|q|$, the number of terms in the query:

$$P'(t|\theta_q) = \hat{P}(t|q) = \frac{f_{q,t}}{|q|} \tag{2.20}$$

Using this simplistic approximation, the probability of the word "pharmaceutical" in a query such as "who has only hope, but no pharmaceutical" would be calculated as $P'(t|\theta_q) = 1/7$, when not using stopping. However, estimating the language model of a query in this way only leads to an approach that is rank equivalent to the query likelihood model. Therefore, using this simple approach of calculating $P(t|\theta_q)$ is unhelpful (it offers no improvement over the query likelihood model) and a more complex technique for estimating language models for queries is needed.

Therefore, the language model of a query is formulated in a more effective way by estimating the model over a large set of documents that could be retrieved by a simple bag-of-words approach or the query likelihood model. Thus the sparseness problem of the query is side-stepped. This method of deriving a better query language model is referred to as the *relevance model* and could also be used in the document likelihood approach, but works best when used in combination with model comparisons.

The *true* Bayesian estimate of any particular word given an information need can be calculated as follows, where the language model of the query is constructed from the document models of all relevant documents $R$:

$$P(t|R) = \int_\theta P(t|\theta)P(\theta|R)d\theta$$

where $\theta$ is in effect a placeholder for any imaginable multinomial model, since we cannot know exactly what the true query model is. We use $\theta$ rather than $\theta_q$, as we do not want to exclude any model while deriving a language model for the query.

However, there are two problems with constructing this language model for a query. First, the set of relevant documents $R$ needs to be known. If this set is known a priori, then there would be no need for a retrieval system. $R$ could be estimated as $R'$ by some heuristic such as assuming all documents that contain any or all of the query terms to be relevant. A second, and more difficult problem, is that it is infeasible to compute the integral over all models. An approximation for the integral needs to be found. Lavrenko (2004) proposes that the integral can be estimated by summing over the set of documents assumed to be relevant ($R'$):

$$P(t|R') = \int_\theta P(t|\theta)P(\theta|R')d\theta \approx \sum_\theta P(t|\theta)P(\theta|R') \tag{2.21}$$

This method provides a way to estimate the query language model $\theta'_R$ by calculating the divergence between documents and the relevance model, according to which documents are ranked:

$$KLD(\theta'_R, \theta_d) = \sum_{t \in V} \theta_{R',t} \log \frac{\theta_{R',t}}{\theta_{d,t}} = \sum_{t \in V} P(t|\theta_{R'}) \log \frac{P(t|\theta_{R'})}{P(t|\theta_d)} \qquad (2.22)$$

where $\theta_{R'}$ is the language model of the set of documents used as a substitute of all relevant documents. As mentioned earlier, the set $R'$ can be estimated by for instance using a simple bag-of-words approach, where all documents containing one or more query terms are assumed to be relevant for the purposes of creating a language model for the query.

Other approaches for constructing query and document models for use in the comparison method have been proposed (see for example Hiemstra et al., 2004) but are not considered further here.

Even though the model comparison approach produces good results in practice, due to its high computational complexity and therefore low system throughput, it is usually not suitable for a live system.

### 2.4.4 Efficient query evaluation

The efficient evaluation of queries is essential for information retrieval systems such as web search engines that need to deal with large numbers of users. The efficient evaluation of ranked queries has lead to considerable research interest; see for example Witten et al. (1999, pages 207–210). Most of the strategies used to speed up evaluation focus on two aspects: reducing the number of inverted lists that need to be processed in order to service a query; and limiting the portion of the inverted lists that need to be processed.

#### Quit and continue strategies

Moffat and Zobel (1994) have proposed two classes of approaches that aim to both reduce the number of inverted lists and the amount of each list that needs to be processed (see also Moffat et al., 1994). Both of these are based on the observation that terms that are relatively frequent across the collection are less likely to change the ranking than those terms that are comparatively more rare. In both approaches, terms are evaluated in order from the rarest to the most frequent (that is, in order of inverse document frequency). The *quit* strategy stops processing lists as soon as a pre-defined percentage of the total documents in the collection have been given entries in the accumulator table for the current query. The *continue* strategy differs in that it does not stop processing lists completely after the threshold is reached, but rather only updates the scores of those documents that already have an entry in the accumulator table. Naturally, quit results in faster query evaluation, although it achieves a relatively poor accuracy. The continue strategy only marginally increases evaluation speed; more surprisingly, it sometimes improves effectiveness compared to exhaustive evaluation. When either of these strategies are used, accumulator table sizes are typically only $1\%$ of the number of documents in the collection at hand.

#### Frequency ordered indexes

Even though a particular query term might occur very infrequently, this alone is not necessarily a good basis for the decision that the inverted list of this term should not be evaluated, as detailed above. A potential shortcoming of this method is that the number of times a term appears in a document is not taken into account. A frequent term can still have a large impact on the ranking of a document if the term appears relatively often in that document compared to a rare term that appears only once.

*Frequency ordered indexes* make use of this information by storing document information in inverted lists in order of the within-document frequency, rather than in the order in which documents were indexed, which is typically an arbitrary ordering. During query evaluation, lists are not processed sequentially, but in parallel, starting with those lists that begin with the greatest within-document term frequency. The quit or continue strategies can optionally be employed. This approach has been shown to work at least as well as just employing the quit or continue strategies and leads to similar storage requirements for the index (Persin et al., 1996).

#### Impact ordered indexes

A frequent term that appears frequently in a large document probably does not contribute as much to the ranking as a rare term appearing once only in a relatively short document. *Impact ordering*, another technique for ordering entries in inverted lists, also takes the document length into account (Anh and Moffat, 2002, 2004). Here the order of the entries is given by the overall impact on the ranking a term has in a particular document on the overall ranking. As for the stop strategy described earlier, using this ordering means that the processing of lists can be stopped once a dynamic threshold is reached, leading to faster query evaluation. In addition, the inverted lists can be capped at a minimum threshold during indexing time, so that terms with an impact below the threshold are not included in the inverted list. This means that storage requirements are reduced, which in turn reduces the times required to fetch the inverted lists from disk during query evaluation. Through a pruning effect similarly to that achieved by employing the quit or continue strategies, effectiveness is slightly improved using this approach while evaluation time is decreased (Anh and Moffat, 2002).

## 2.5 Additional information useful for document ranking on the web

In addition to ranking documents according to their textual content, other sources of evidence can be made use of. For example metadata in the form of author details, or the creation date of a document, can give clues about the relevance of a document. Web search, in particular, gives opportunities for using additional information.

One web-specific form of metadata is the web address. When looking for a company's web page, in addition to the text appearing on the home page of a company, the address – or *universal resource locator*

(URL) – can be a good indicator of the relevance of this page. For example, when a user is looking for the web site of Qantas, the URL (http://www.qantas.com.au/) contains the name of the company.

Another resource of information is given by the links that connect resources on the web. To make use of this information, the Google founders developed *PageRank* (Page et al., 1998), where pages distribute a score amongst the pages that they point to while receiving a share of the score of pages that they in turn are referenced by. Since this is a recursive process, the page score continues to be updated over several iterations, until the score of each page converges. Convergence can be achieved more quickly if the initial weight of each page is carefully chosen. The score $pr(P)$ of page $P$ linked to by page $B$, which contains $OutDegree(B)$ links, can be calculated as follows:

$$pr(P) = \frac{\beta}{N} + (1 - \beta) \times \sum \frac{pr(B)}{OutDegree(B)}$$

where $\beta$ is the probability that a user might make a random jump to any page $P$, rather than following one of the links on the current page, and there are $N$ pages in the web graph (Xi et al., 2002). However, PageRank has not been shown to increase retrieval effectiveness in a controlled research environment (Upstill et al., 2003). An alternative methodology based on links is the hub-and-authority approach (Bharat and Henzinger, 1998).

### 2.5.1 Anchor text

Links in web pages can be bound to any component of a web page, such as pictures or text. The string of terms that make up such a link is called *anchor text*. Craswell et al. (2001) examined the effectiveness of document surrogates created from such anchor text in order to find entry pages to web sites. In their work, the text content of hypertext links or anchor tags that *in-link* to a document is extracted and compiled into a document surrogate. Experiments show that document surrogates derived from anchor text are significantly more effective than full-text retrieval for a named page-finding task (Hawking and Craswell, 2001). However, anchor text was not found to be useful for topic-finding tasks. Therefore, for example, anchor text could be expected to aid retrieval for queries such as “Qantas Airways”, but not for queries such as “dealing with air sickness”. We examine the use of anchor text as a source for query expansion terms in Chapter 5.

### 2.5.2 Past queries

A *query log* is a list of queries previously posed by users to a search service and may contain additional data such as the time when a query was posed, the IP address of the browser from which the query was issued, and click-through data (Joachims, 2002). The latter indicates which links a user followed subsequent to viewing search results. Search engines commonly keep query logs to enable caching of queries and the associated results, to aid reporting of statistics (such as Google’s Zeitgeist, http://www.google.com/press/zeitgeist.html), and for future research.

One technique that makes use of past user queries is query association, which was proposed for the creation of document summaries, in order to aid users in judging the relevance of answers returned by a search system (Scholer and Williams, 2002). In this thesis, we make use of associated queries as a source of terms for query expansion, as detailed in Chapter 5.

## 2.6 Evaluation of information retrieval systems

In information retrieval it is often necessary to compare the implementation of two algorithms against each other in order to check whether a new algorithm constitutes an advance over a previous one. There are typically two dimensions that need to be tested. The first is efficiency, which is a measure of the amount of resources used for an algorithm to perform a certain task. This can be further divided into space and time efficiency. The second major performance dimension is the effectiveness of the algorithms used; it measures whether the ranking produced by a new algorithm is more satisfactory for a user than that of a previous one. In the following we examine these performance measures.

### 2.6.1 Measuring efficiency

Efficiency can be measured in both the time and space domain. For many applications in the area of information retrieval and in particular query expansion, time is of principal importance and the only aspect of this thesis where we are interested in space efficiency is in Chapter 6, where we measure the amount of main memory taken up by data.

The use of memory can either be measured through reporting built into the test application, or through external profilers, such as *Valgrind* (Nethercote and Seward, 2003). Timings can be obtained in addition to using either of these methods, by using operating system tools such as *timex()* on a Unix platform. In this thesis, except where noted otherwise, we use 10,000 stopped queries taken from two query logs collected from the Excite search engine (Spink et al., 2002) to obtain durations for the average query evaluation time.

### 2.6.2 TREC testbed

One of the most commonly used testbeds in modern information retrieval research is provided by the annual *Text REtrieval Conferences* (TREC) by the National Institute of Standards and Technology of the United States (http://trec.nist.gov/). One of the reasons that this conference was set up was to give researchers from different countries the opportunity to test their systems against common collections and queries and therefore compare them on a sound basis.

Each year, TREC consists of a number of tracks, where each track represents a different task, such as web retrieval (Hawking and Craswell, 2004) or question answering (Voorhees, 2003). For each track a collection of documents (such as a newswire collection, or a subset of the web (Voorhees and Harman, 1999, 2000)) is distributed to participants along with a set of queries (these are described in detail below). The participants then rank the documents in the collection against the queries provided, where the algorithms used for ranking depend on the track the participants are subscribed to. After the participants submit their results, the relevance of the top answers (usually 100) for each run of results is pooled and assessed by a team of human assessors (Voorhees, 1998). It is important to note that documents that have not been judged for a particular query are assumed to be non-relevant to the query.

The accumulated collections and query sets of previous years give researchers the opportunity to compare new algorithms using consistent effectiveness measurements.

### TREC collections

Several test collections, including newswire data, genomic data, and web data have been used as part of TREC. Collection sizes have increased over the years, starting from around 2 GB at TREC 1 to 426 GB at TREC 2004. Each collection is made up of a number of documents, ranging from a few thousand to several million. Documents are marked up using SGML in a specific TREC format. This consists of a unique TREC document identifier, the content itself, and optionally some metadata such as authorship and a time stamp. An example TREC document is shown in Figure 2.1.

We make use of the following collections in this thesis (some details are given in Table 2.9):

- WSJ2: a portion of the Wall Street Journal collection that was distributed on TREC TIPSTER disk 2, containing news articles from the years 1990 to 1992.
- AP: Associated Press newswire articles from the years 1988 and 1989.
- NW: a group of newswire collections, described by Voorhees and Harman (1999). It is made up of four sources:
  - FT: the Financial Times from 1991 to 1994;
  - FR94: the Federal Register from 1994;
  - FBIS: collections of the Foreign Broadcast Information Service from 1994; and
  - LA-Times: a collection of news articles from the Los Angeles Times from 1989 and 1990.
- WT10g: a 10 gigabyte crawl of web data. It was designed to resemble a representative portion of the web and exhibits a large degree of interconnectivity so that anchor text and link structure could be made use of for retrieval experiments. It is a subset of the VLC (Very Large Collection) that was crawled in 1997 (Hawking et al., 1998). The WT10g collection is described in more detail by Bailey et al. (2003).

```
<DOC>
<DOCNO> WSJ870323-0181 </DOCNO>
<HL> South Korea's Current Account </HL>
<DD> 03/23/87 </DD>
<SO> WALL STREET JOURNAL (J) </SO>
<IN> FREST MONETARY NEWS, FOREIGN EXCHANGE, TRADE (MON) </IN>
<DATELINE> SEOUL, South Korea </DATELINE>
<TEXT> South Korea posted a surplus on its current account of
       $419 million in February, in contrast to a deficit of
       $112 million a year earlier, the government said.
       The current account comprises trade in goods and
       services and some unilateral transfers.
</TEXT>
</DOC>
```

*Figure 2.1: Example TREC document. Shown is Document 2 of the WSJ newswire collection used at TRECs 6 to 8. The TREC document identifier (or* document number*) is specified and other – collection specific – metadata includes the headline and date of the news article. Typically all text between the <TEXT> and </TEXT> flags and some of the metadata is indexed, such as the headline.*

**TREC queries**

Each TREC collection has typically 50 to 100 corresponding queries. A TREC query is structured as follows: it contains a unique TREC query identifier; a topic title; a more detailed description of the information need; and a narrative that explains under what circumstances a document should be judged relevant or not relevant to a query. A sample TREC query is shown in Figure 2.2.

In all of our experiments we use the title field of queries only, since these are similar to how many users would actually formulate their information needs (Jansen et al., 2005). For the TREC-9 web track, the topic titles were taken from search engine logs (Hawking, 2000).

In this thesis we primarily use queries 351 to 450, which is the data used at TREC-7 and 8 (Voorhees and Harman, 1998, 1999), and the query sets (451 to 550) that were developed as part of the web tracks at the TREC-9 and 10 conferences (Voorhees and Harman, 2000, 2001).

### 2.6.3 Measuring effectiveness

Multiple measures have been proposed for measuring effectiveness for information retrieval systems. Although Voorhees (2000) discusses that relevance assessors do not often agree in their judgements,

| Collection name | Number of documents | Size | Average size (terms) |
|---|---|---|---|
| WSJ2 | 74,520 | 242 MB | 508 |
| AP | 164,597 | 491 MB | 471 |
| NW | 528,155 | 1.9 GB | 498 |
| FT | 210,158 | 564 MB | 413 |
| FR | 55,630 | 395 MB | 645 |
| FBIS | 130,471 | 470 MB | 544 |
| LA-Times | 131,896 | 475 MB | 326 |
| WT10g | 1,692,096 | 10 GB | 597 |

*Table 2.9: Characteristics of some of the most common TREC collections.*

she found that the TREC based measurement of system effectiveness is reliable. However, using more than two dimensions of relevance can lead to unstable measurements, as Voorhees (2001) found using ternary relevance judgements. In this thesis we therefore consider only measures that are based on the assumption of binary relevance, as discussed in Section 2.3. An effectiveness measure is based on the ranks of relevant and non-relevant documents in the answer set. The main two measures, *recall* and *precision* are discussed in the following.

#### Recall

One important aspects of results is how many of the relevant documents in a collection have been found. *Recall* shows how many of the relevant documents a user could possibly come across when reading all documents in the answer set. It is calculated as follows:

$$\text{Recall} = \frac{\text{number of relevant documents retrieved}}{\text{number of relevant documents}}$$

Therefore the higher the level of recall, the better the system is.

Consider the following example, where a search engine has retrieved ten documents in response to a query. From Table 2.10, which shows a ranked answer set, we can see that only four relevant documents are amongst the top ten retrieved documents. Given that there are eight relevant documents in the collection, recall would be $|R| = 4/8 = 50\%$. Since all documents in a collection could conceivably be presented in the rankings (where the recall would be 100%), it is necessary to note at which level of cutoff this recall was produced. The recall achieved at cutoff level 10 would be 50% in the example, whereas the recall at level 5 would be 12.5%.

This and the following effectiveness measures have been widely discussed, see for example Voorhees and Harman (1998, 1999).

```
<num> Number:  405
<title> cosmic events
<desc>  Description:
        What unexpected or unexplained cosmic events or celestial
        phenomena, such as radiation and supernova outbursts or
        new comets, have been detected?
<narr>  Narrative:
        New theories or new interpretations concerning known
        celestial objects made as a result of new technology are
        not relevant.
```

*Figure 2.2: Example TREC query, including the query number, the title, a description of the information need and a narrative, which gives instruction on how to judge a document in response to a query.*

### Precision measures

Arguably, what is more important in a typical retrieval setting – where a user might be interested in finding more information on a particular topic and one of many relevant documents can satisfy a user's information need – is how many relevant documents a user can expect to find amongst a certain number of top ranked documents. The associated measure is called *precision* and is calculated using the following equation:

$$\text{Precision} = \frac{\text{number of relevant documents retreived}}{\text{number of retrieved documents}}$$

The maximum (and optimal) precision value would be 100% and the worst possible precision of 0% is achieved when not a single relevant document was found. As in the case of recall, precision must always be stated at a certain cutoff level. In the running example, precision at the cutoff level of 10 would be Precision$@10 = 4/10 = 0.4 = 40\%$. This is denoted as $P@10$. Since we know from Table 2.10 that amongst the top 5 results only Document 2 is relevant, we can calculate the precision at 5 documents as $P@5 = 1/5 = 20\%$. Precision is commonly stated at the cutoff levels of 10, 20, and 100.

A special value for the precision cutoff level is the number of relevant documents that exist in a collection for a particular query. Precision values at that point are called *R-Precision* (R-Pr.). It is maximised if every single relevant document appears in the ranking before any non-relevant document. In the above example, there are 8 relevant documents, so therefore R-Precision $=$ Precision$@8 = 3/8 = 37.5\%$.

A commonly used measure is the *average precision*, which is calculated over the whole answer set of ranked documents. It results in a single value for the whole ranking and is calculated by averaging the precision values obtained after each relevant document is retrieved. If a relevant document is not ranked,

| Rank | Relevance judgement |
|---|---|
| 1 | non-relevant |
| 2 | relevant |
| 3 | non-relevant |
| 4 | non-relevant |
| 5 | non-relevant |
| 6 | non-relevant |
| 7 | relevant |
| 8 | relevant |
| 9 | non-relevant |
| 10 | relevant |

*Table 2.10: Ranked answer set with relevance judgements used in example of effectiveness measures. Shown are the ranks of four of eight relevant documents in total.*

a zero will be added during the averaging process. As all relevant documents are thus contributing to the single value, it is not only a measure of precision but also of recall. Unlike the previously described precision-at-cutoff measure, it has the advantage that it is sensitive to the entire ranking order. If another system ranks the document currently at rank 2 at rank 3 instead, the average precision value will change, which is not the case for the non-averaged precision value, which would remain the same (for cutoff values above two). The average precision measure is also relatively stable, that is, a change as described above would result in only a fairly small change in the average precision value. Another advantage is that an average precision figure is influenced to a greater degree by relevant documents at higher ranks, while lower ranked documents play a smaller role. This could be seen as reflecting the perception of the usefulness of a system to a user in a typical retrieval situation (for example when exhaustive search is not performed).

In the above example, the average precision would be calculated as:

$$\text{Average Precision} = \frac{1}{8} \times \left(\frac{1}{2} + \frac{2}{7} + \frac{3}{8} + \frac{4}{10}\right) = \frac{1}{8} \times \frac{140 + 80 + 105 + 108}{280} = \frac{435}{2240} = 19.4\%$$

The *mean average precision* (MAP) is used for the average precision figures over a number of different queries. The measures described above are used throughout this thesis.

### Mean reciprocal rank

One other measure that is particularly relevant for the home page and named page finding tasks (Hawking and Craswell, 2004) is the *mean reciprocal rank* (MRR). This measure is calculated by the inverse rank of the first relevant document (or correct answer, which is a better terminology to use in this context), or 0, if no relevant document was ranked. MRR has also been used for the question answering task; for example, see (Clarke et al., 2001) and in many other contexts.

#### Statistical significance testing

It is a mistake to claim a significant change in performance based only on different effectiveness scores (Zobel, 1998). While post-hoc analysis of error rates can give valuable information about the properties of a collection – see for example Voorhees and Buckley (2002), who calculate error rates for the TREC-3 to TREC-10 data empirically – such thresholds cannot safely be extended to future runs; as these runs were not themselves part of the calculation, per-query variability would not be taken into account. A statistical significance test, on the other hand, enables conclusions to be drawn about whether a variation in retrieval technique leads to consistent performance gains.

In this thesis, we evaluate the significance of our results using the Wilcoxon matched-pairs signed rank test (see for instance Sheskin, 1997, pages 291–302). This is a non-parametric procedure used to test whether there is sufficient evidence that the median of two probability distributions differ in location. For information retrieval experiments, it can be used to test whether two retrieval runs based, for example, on different query expansion techniques, differ significantly in performance. As it takes into account the magnitude and direction of the difference between paired samples, this test is more powerful than the sign test (Daniel, 1990). Being a non-parametric test, it is not necessary to make any assumptions about the underlying probability distribution of the sampled population.

The t-test, an alternative, parametric test for paired difference analysis, assumes that the data is normally distributed. Zobel (1998) analysed the results of retrieval experiments from TREC-5, and concluded that the Wilcoxon signed rank test is a more reliable test for the evaluation of retrieval runs.

Hull (1993) compared various statistical tests and suggests that where multiple treatments exist, they can be combined in one statistical test.

## 2.7 Summary

In this chapter, we considered how an information need is conceived by a user, and saw that the user query is only an approximation of the underlying information need. An information need can therefore be expressed through different queries, while the same query can be the manifestation of different information needs.

We explained that indexes provide a means to quickly find documents that contain search terms. During indexing, all documents are parsed, and the encountered terms may be case folded, stemmed and stopped. For each term that occurs in the document at hand, an entry is made in that term’s inverted list.

We considered how queries are processed by matching documents against them. There are a variety of methods. We discussed Boolean queries, for which documents are returned to the user depending on a simple term matching algorithm. Documents are returned in no particular order as long as they fulfil the criteria of the user query: they contain the terms that the user specified relevant documents must contain, and they do not contain those that the user explicitly excludes. All retrieval models discussed make use of ranked queries, where documents are returned in a ranked order, according to specific heuristics. The vector space model translates queries into a vector space. All documents are represented in this vector space and documents are ranked in inverse order of the size of angle between the query and document vectors. In the probabilistic model, documents are ranked by their likelihood of being a good match to the current query. The model estimates the overlap in terminology between a query and a document, where rare terms are given a greater weight than frequent terms. Language models, a further class of retrieval models, on the other hand rank documents by their likelihood of having generated a user query.

Next we considered common techniques for evaluating queries more efficiently by biasing results towards the rare query terms. In addition, depending on some heuristics, the impact of more frequent terms can be neglected so that only part of the inverted lists have to be processed.

We explained that anchor text (and other link information) can have a large impact on retrieval on the web, as anchor text is typically a good descriptor of the page it is referring to. We also described the role of query logs, where past user queries are stored alongside other related information from which entities such as user access patterns, or query reformulation patterns, can be deduced.

Finally we described the TREC testbed used for evaluation of our experiments in this thesis and defined the most commonly used evaluation measures, namely recall and precision and related measures.

Chapter 3

# Query Reformulation

A user query is, in general, only one of many possible formulations of the information need that a user may have. As a result, user queries often do not reflect the exact terminology of a document that fulfils that information need, as discussed in Section 2.1. Whereas a document might contain the exact information that a user is seeking, this document cannot be retrieved if it does not contain any of the keywords used in the user's query, if an exact matching algorithm such as the vector space model or Okapi BM25 is used (see Sections 2.4.1 and 2.4.2, respectively). Reformulating a query is an attempt to improve poor user queries by: removing terms that degrade retrieval performance; adding terms that aid retrieval; re-weight existing or new query terms to give the query a different emphasis; or a combination of those methods.

A possible side-effect of query re-formulation is that queries can suffer from *query drift*, where badly performed expansion leads to a change in the topical direction of a query (Mitra et al., 1998). For example, a user might be interested in the state of mind reputedly achieved by Buddha, and a succinct query relating to this information need might be "Nirvana". When expanded, the query might become "nirvana cobain kurt band live music", which focuses on a completely different topic from the intended query. (This query is number 494 of TREC 9 (Voorhees and Harman, 2000), where the information need was to "find information on members of the rock group Nirvana", unlike in the example given above. The corpus searched – a subsection of the web – was amenable to this request and in our experiments the query was in fact expanded as above, leading to a good increase in effectiveness for this query.)

In a typical retrieval session, a user will formulate a query that satisfies their information need in several iterations. The first attempt at formulation of a query with a particular information need in mind is often inaccurate and can result in an answer set that does not satisfy the user's information need. The query might have been too broad, resulting in a wide range of documents being returned in the answer set, so the relevant documents are sparsely mixed in with documents that are far from the topic sought. Alternatively, the query might have been too narrow, only identifying a small subset of relevant documents. A third alternative is that the query entered by the user was completely deficient and the intended set of documents and the actual result set have no intersection.

In this last case, it is probably not possible to help the user other than encouraging them to start over with a new query. In either of the first two cases, however, the query can be improved. In the ideal case, the user will reformulate the query. Jansen et al. (2005) show that users – at least in a web setting – increasingly take this option, with more than 50% of users modifying their initial queries in 2002, up from 20% four years earlier.

After reading some of the documents in the initial result set, or the query-biased summaries that many querying interfaces provide (see for instance Tombros and Sanderson, 1998), the user might append additional terms to the query that they feel would clarify a particular topic, or might lead to the inclusion of particular documents that they may have found to be missing from the results. The user can also remove terms from the original query that they perceive to have been detrimental in the retrieval process, either because those terms overly restricted the result set, or because they were found to lead the query off-topic. The user then re-issues the updated query and the process is repeated. Leading on from our earlier example, after being presented with documents that are on the topic of the rock band nirvana, the user re-issues their modified query "nirvana buddhism" in order to shift the result set toward their information need. As the user reformulates their query upon judging the relevance of some of the answers provided in the result set, this approach is a form of manual *relevance feedback*, which is discussed in the following section.

In this chapter we give a brief history of query reformulation schemes, with an emphasis on automatic relevance feedback based on local analysis algorithms, which are the focus of this thesis. For a comprehensive review of query reformulation schemes that include a greater emphasis on other groups of algorithms, see Efthimiadis (1996), Ruthven and Lalmas (2003) or Spink and Losee (1996). Document routing and filtering are related to relevance feedback; however we do not go into detail about these topics here. Instead, please see for instance Broglio et al. (1994) or Buckley et al. (1995) for routing and Robertson and Soboroff (2001) for filtering.

## 3.1 Relevance feedback

*Relevance feedback* requires a user to classify documents in an answer set as relevant or non-relevant. Non-judged documents are often assumed to be non-relevant to the user's information need. Algorithms then make use of this knowledge in order to construct a better query.

Relevance feedback is a particular method of manual query reformulation, where the user carefully checks the answer set resulting from an initial query, and then reformulates the query. This places the burden of the query reformulation task on the searcher. This method is therefore often undesirable from a search provider's point of view, as the user may give up on the current search and seek an alternative

| | |
|---|---|
| $q$ | A query |
| $\vec{q}$ | The weighted query vector of query $q$ |
| $d$ | A particular document |
| $\vec{d}$ | The weighted document vector of document $d$ |
| $C$, $N$ | The set and number of all documents in the collection |
| $C'$, $N'$ | The set and number of all seen documents |
| $V$, $\|V\|$ | The set and number of distinct terms in the vocabulary |
| $f_t$ | The number of documents term $t$ appears in |
| $f'_t$ | The number of seen documents term $t$ appears in |
| $R$, $\|R\|$ | The set and number of all relevant documents, given a particular query |
| $R'$ | A subset of relevant documents, typically all relevant documents retrieved |
| $\overline{R}'$ | A subset of non-relevant documents, typically all non-relevant documents retrieved |
| $r'_t$ | The number of documents in $R'$ that contain term $t$ |
| $x$ | An item such as a term or document, that can be either relevant or non-relevant |
| $F_t$ | The total number of occurrences of term $t$ in the collection $C$ |
| $F$ | The combined total number of occurrences of all terms in the collection |

*Table 3.1: Summary of notation needed for the construction of an "optimal" query using relevance feedback. Note that* seen documents *are those that a user has judged for relevance.*

search service. Since the 1960s (see for instance Salton, 1971), it has been thought that the task of reformulating a query could be done by a computer; an algorithm might be more thorough than relying on a user, and have a greater chance of getting the final query right, since it can employ information that is not known to a user, such as collection statistics.

### 3.1.1 Construction of an optimal query

Before discussing how a query can be improved through relevance feedback, it is important to understand what constitutes an ideal query. A query is optimal if it ranks all relevant documents before those that are not relevant, that is, it would lead to a ranking with an average precision of $1.0$. If a ranking system based on term matching is used (such as the vector space model or Okapi), a query is most likely to achieve a ranking that is as close to optimal as possible if it contains all terms that appear in all relevant documents, but explicitly discounts all terms that occur in non-relevant documents.

Consider a collection $C$, consisting of $N$ documents. Then, using our definition of optimality, an "optimal" query vector could be constructed from the document vectors $\vec{d}$ of all $|R|$ documents in the relevant set $R$ of documents (Rocchio, 1971):

$$\vec{q}_{optimal} = \frac{1}{|R|}\sum_{\vec{d}\in R}\vec{d} - \frac{1}{N-|R|}\sum_{\vec{d}\in(C-R)}\vec{d}$$

where the first part of the equation adds terms that appear in relevant documents and the second removes terms (or gives a negative weight in the query vector to terms) that occur in non-relevant documents. (The symbols used in this chapter are summarised in Table 3.1.)

This approach does not guarantee that all relevant documents are ranked before all non-relevant documents, and would almost certainly fail if a new relevant document was introduced to the collection without updating the query. A query constructed in this way will therefore almost certainly be suboptimal.

### 3.1.2 Approaching optimality: the Rocchio method

Unfortunately the construction of an optimal query is hardly realistic. If the set of all relevant documents $R$ was known at the outset of a search session, then the use of a retrieval system would not be necessary.

Instead, we consider construction of a sub-optimal query using partial relevance judgements; here, the user only needs to identify a relatively small set of relevant and non-relevant documents. Rocchio (1971) proposed the following formula, which is used to improve an initial query $q$ with partial sets of relevant documents ($R'$) and non-relevant documents ($\overline{R'}$):

$$\vec{q}_{new} = \alpha \times \vec{q}_{initial} + \frac{\beta}{|R'|}\sum_{d\in R'}\vec{d} - \frac{\gamma}{|\overline{R'}|}\sum_{d\in\overline{R'}}\vec{d} \tag{3.1}$$

where $\alpha$, $\beta$, and $\gamma$ are tuning constants with typical values of $1$, $8$, and $1$, respectively; $\alpha$ governs the influence of the original query, while $\beta$ and $\gamma$ regulate the impact of relevant and non-relevant documents respectively. Equation 3.1 adds weighted terms from judged documents to the original query, denoted as $\vec{q}_{initial}$. The judged documents act as positive and negative examples of the terms that should occur in relevant and non-relevant documents. The modified query is then re-issued, with the expectation that the remaining relevant documents will be ranked more highly (Rocchio, 1971, Ruthven and Lalmas, 2003). The formula can be applied in several iterations, while ideally $q_{new}$ gets closer to the "optimal" query with each iteration. However, if the terminology and term distribution of the intended set of relevant documents is not sufficiently distinct from that of the non-relevant documents, then the query will never become close to optimal.

A simplified use of the formula above is achieved when setting $\gamma$ to 0, and thus not using any non-relevant documents in the feedback process. In most applications of the Rocchio algorithm, negative examples are not used, that is, no *negative feedback* is employed. This algorithm and variations thereof have found wide usage in information retrieval and related areas such as text categorisation (Joachims, 1997).

In the context of routing, Buckley and Salton (1995) extend Rocchio's work by training the term weights of query terms on a training set of documents with relevance judgements. They achieve a relative improvement of 10% to 15% on one test collection.

Relevance feedback has been used widely in order to improve upon search effectiveness (Salton and McGill, 1983, van Rijsbergen, 1979, Frakes and Baeza-Yates, 1992, Chapter 11, Leuski, 2000, Baeza-Yates and Ribeiro-Neto, 1999, Chapter 5). However, research shows that users are reluctant to make use of negative relevance feedback, that is, providing information that a document is not relevant, as opposed to indicating the relevance of a document (Belkin et al., 1995, 1996). Users feel that their indications of non-relevance might be mis-interpreted by the system and could lead to a low ranking of relevant documents or even to an omission of those documents from the result set.

### 3.1.3 Relevance feedback methods

In this section, we describe some of the most commonly cited relevance feedback mechanisms, other than Rocchio's as detailed above. An explanation and evaluation of those methods is given by Salton and Buckley (1990).

#### Evaluating retrieval effectiveness when employing relevance feedback

Before examining some of the relevance feedback mechanisms, it is worth noting that the evaluation measures typically used in information retrieval (see Section 2.6.3) are insufficient for measurement of manual relevance feedback techniques. Typically the positions of all retrieved relevant and non-relevant documents are taken into consideration when calculating effectiveness. However, when manual relevance feedback is used, where documents are confirmed as either relevant or not, and this has an influence on the next iteration of queries, then the resulting ranking of documents is affected by the user judgements. Depending on the effectiveness of the feedback mechanism, documents confirmed to be relevant are typically ranked before any other documents, and documents that are confirmed to be non-relevant are either ranked very low, or not ranked at all, if not all documents are ranked. This effect artificially inflates evaluation measurements. It is desirable that only documents that are not assessed – the *unseen documents* – are used for evaluation of a feedback mechanism.

Chang et al. (1971) offer three options to control for this effect. The first is called *modified freezing*. It is a modification of the freezing method (*full freezing*), where the ranks of all documents assessed so far are frozen and only unseen documents are re-ranked. In modified freezing, only the ranks of documents up to the lowest ranked relevant document are frozen. A problem with this approach is that at later iterations, an increasingly large number of the ranking is frozen and that the effectiveness of the relevance feedback mechanism can seem worse than it actually is.

A second option is *residual ranking*, where documents that are used for relevance feedback are removed from the collection before ranking with the reformulated query. A problem here is that eventually all relevant documents will be eliminated from the collection, which has an undesirable impact on evaluation measurements.

Finally, one can use *test and control groups*, where the collection is split into two groups: one from which documents are drawn for feedback, the second to evaluate the feedback mechanism. A problem with this approach is to find a good way of splitting the collection; it is neither guaranteed that the distribution of relevant documents is similar in both sub-collections, nor that those terms that may successfully distinguish relevant from non-relevant documents in one part of the collection are useful in the other.

As all of the approaches above have some disadvantages and might have a different impact on an experiment, depending on what is tested, various approaches have been used by different researchers.

### ***Ide dec-hi*** **and** ***Ide regular***

Two methods which are both similar to the Rocchio method shown earlier have been developed by Ide (1971); see also Ide and Salton (1971). Both are named after the author. The first is called *Ide dec-hi* and uses all relevant documents retrieved so far during the search process as positive examples, but only the first returned document that was identified as non-relevant is used for negative feedback:

$$\vec{q}_{new} = \vec{q}_{initial} + \sum_{d \in R'} \vec{d} - \vec{d}_{top\ non-relevant}$$

*Ide regular* is only different in that it treats all non-relevant documents as negative examples:

$$\vec{q}_{new} = \vec{q}_{initial} + \sum_{d \in R'} \vec{d} - \sum_{d \in \overline{R'}} \vec{d}$$

In either case, the full weights of terms are used to modify the query, as opposed to a dampened weight such as used in the Rocchio method, given in Equation 3.1.

According to Salton and Buckley (1990), Ide dec-hi performs consistently better than the Rocchio method with parameters of $\beta = 0.75$ and $\gamma = 0.25$, whereas Ide regular leads mostly to worse results.

### **Probabilistic relevance feedback**

Similarly to the derivation of the Okapi BM25 method given in Section 2.4.2, the probabilistic approach to relevance feedback is based on the ratio between the probability of an item $x$ (such as a term or document) being relevant or non-relevant (Robertson and Sparck Jones, 1976, Robertson et al., 1981):

$$\frac{P(x|\textit{relevant})}{P(x|\textit{non-relevant})} \tag{3.2}$$

Assuming that terms appear independently of relevance in documents, and binary relevance of documents is used, the weight $w'_t$ of each term can be determined as follows:

$$w'_t = \log \frac{p_t(1-q_t)}{q_t(1-p_t)} \tag{3.3}$$

where $p_t$ and $q_t$ are defined as $P(t = \textit{present}|\textit{relevant})$ and $P(t = \textit{present}|\textit{non-relevant})$, respectively. This weight is also referred to as the *binary independence weight*, because it assumes that terms appear in documents independently of the relevance of these documents (Robertson et al., 1981).

Note that, where a group of documents is considered in the context of relevance feedback with users judging the relevance of documents, this group of documents is the set of *seen* documents during a retrieval session, rather than the whole document collection, where unseen documents are assumed to be non-relevant. We therefore use $N'$ in the following discussion in order to distinguish it from $N$ in previous and following sections. The use of $f'_t$ is analogous to that of $N'$.

Following on from Equation 3.3, the statistics of the incident contingencies given in Table 2.7 give good fits for $p_t$ and $q_t$ as $r'_t/|R'|$ and $(f'_t - r'_t)/(N' - |R'|)$, respectively. Here, $r'_t$ is the number of seen relevant documents that contain term $t$. These can be used to formulate a new weight:

$$w_t = \log \left( \frac{r'_t}{|R'| - r'_t} \times \frac{N' - |R'| - f'_t + r'_t}{f'_t - r'_t} \right)$$

Using similar considerations to those in the previous chapter, where missing evidence can lead to probabilities with a value of zero, the value of $0.5$ is added to the final feedback values of the formulation:

$$w_t^{\textit{point-5}} = \log \left( \frac{r'_t + 0.5}{|R'| - r'_t + 0.5} \times \frac{N' - |R'| - f'_t + r'_t + 0.5}{f'_t - r'_t + 0.5} \right) \tag{3.4}$$

The resulting formula is known as the $F_4$*point-5* algorithm. A modified version ($F_4$*modified*) was devised (Robertson, 1986), where instead of eliminating the possibility of arriving at infinite values with $0.5$, $f'_t/N'$ and $1 - f'_t/N'$ is used as appropriate:

$$w_t^{\textit{modfied}} = \log \left( \frac{r'_t + \frac{f'_t}{N'}}{|R'| - r'_t + 1 - \frac{f'_t}{N'}} \times \frac{N' - |R'| - f'_t + r'_t + 1 - \frac{f'_t}{N'}}{f'_t - r'_t + \frac{f'_t}{N'}} \right) \tag{3.5}$$

According to Salton and Buckley (1990), this method performs slightly worse on average than the standard Rocchio technique with the same parameters as given earlier.

### 3.1.4 Other approaches to relevance feedback

Manual relevance feedback requires active input from the user. As it can be difficult to persuade users to make use of relevance feedback features (as discussed in Section 3.3), alternative approaches have been investigated.

Rather than requiring the user to judge the relevance of a whole set of documents, Aalbersberg (1992) sidesteps the issue by only displaying one document at a time and thereby effectively forcing the user to participate in relevance feedback. Using *incremental relevance feedback*, the user is able to see the next document only by clicking on one of two buttons that are displayed with the current document (labelled, for example, “document relevant” and “document non-relevant”) and therefore has no option but to provide feedback to the system. Aalbersberg shows that incremental feedback can work better than conventional relevance feedback. However, there is an increased risk that the order of documents may influence the relevance judgements by the user (Eisenberg and Barry, 1988), particularly in this narrow context. Vinay et al. (2004) use a similar approach in the context of mobile devices, where display area is limited.

Campbell (1995) proposes a graphical system for *ostensive browsing* where the user is shown a set of documents previously retrieved that are distributed along branches of the search so far. The user then can backtrack or move forward on the current branch or move to a different branch altogether. All documents (images) up to the current node are taken into consideration for relevance feedback for the expansion of the current node. Campbell’s system was designed for image retrieval, however the same approach could also be used for text retrieval. Relevance feedback was also made use of in the related area of cross-language image retrieval (Clough and Sanderson, 2004).

## 3.2 Interactive query expansion

Full reformulation of an initial, typically somewhat deficient, user query – including alteration of query term weights, adding terms, and subtracting terms – seems to be a sensible option, as this allows fine tuning of the query. There is a relatively high cost involved in this approach, however, as the weights of a potentially large number of terms have to be finely tuned. Because of this overhead, together with the common hypothesis that an initial user query is a good starting point for constructing a query that will result in a satisfying answer set, has the consequence that queries are typically expanded rather than fully reformulated. Negative feedback, where negatively weighted terms are added to the query with the effect that documents containing those terms are ranked lower, is not commonly used. Therefore, most modern approaches to query reformulation are referred to as *query expansion*, where the initial query is refined by adding new terms, rather than removing or replacing original terms, or altering their weights. Over the last decades, research into query expansion has generated a larger number of papers than relevance feedback.

In this and the following sections we discuss the use of query expansion, while concentrating on *interactive query expansion* in this section and on *automatic* approaches in the remainder of this chapter.

In contrast to automatic query expansion, interactive query expansion relies to at least some extent on the input of the user. Although the interactive aspect of query expansion could involve marking seen documents as relevant or non-relevant by a user, more commonly interactive query expansion refers to a process whereby a user chooses the expansion terms. Typically, the user is presented with a list of *candidate terms* (potential expansion terms) during the retrieval session. In order to limit the cognitive load for the user, this list is kept brief. Therefore, an important aspect of interactive query expansion is the determination of a relatively small subset of candidate terms. Harman (1988) investigated the possible usefulness of interactive query expansion. She asked testers to select expansion terms from lists of candidate terms that were generated by one of three methods: relevance feedback, deriving variants of the original terms, or retrieving terms from a manually constructed thesaurus (see also Section 3.3.1). Limiting the number of candidate terms presented to 20, Harman used different mechanisms to select candidate terms to be shown to the user and found that – in a limited experiment with one small collection – sorting candidate terms by *tf.idf*-based heuristics results in greatly improved retrieval. In another experiment she only added candidate terms that occurred in at least one of the relevant documents not seen by the user up to that point. This resulted in a further improvement, but is of limited usefulness, since in a typical retrieval session the relevance or non-relevance of unseen documents is not known to the system.

Magennis and van Rijsbergen (1997) expanded Harman's result, by ranking 20 candidate terms and asking the user to select either none, 3, 6, 10, or all 20 terms for expansion, while comparing results with using 0 terms for no expansion and all 20 ranked terms for automatic expansion (which is explained in detail in the next section and the remainder of this chapter). In contrast to Harman's experiment, Magennis and van Rijsbergen used term weighting for expansion terms. Repeating the experiment with different queries in effect allowed users to make different choices for the number of expansion terms based on each query, which is a clear advantage over automatic expansion, where the number of terms added is typically constant. They found that interactive query expansion can improve retrieval significantly. The results achieved through automatic expansion were less stable than those achieved through interaction. However Magennis and van Rijsbergen conclude that terms need to be selected by experienced or trained users. More recently, graphical user interfaces for the suggestion of expansion terms have been developed (Joho, Sanderson and Beaulieu, 2002). Joho, Coverson, Sanderson and Beaulieu (2002) and Joho et al. (2004) find that hierarchical presentation of search terms leads to statistically insignificant improvements in effectiveness, although users need less time to perform search tasks in this new retrieval environment; they are able to handle a large number of concepts more efficiently.

After a sensible set of candidate terms has been selected, the weighting of terms is of importance. Harman (1988) did not pre-calculate the weights of expansion terms, that is, she just added terms to the original query and then treated the expanded query as if it was a query fully entered by the user without any interaction. In later experiments Harman (1992*b*) improved her previous methods by weighting expansion terms according to how many of the relevant documents those occurred in. She also confirmed two other interesting aspects of query expansion: first, expansion is more beneficial than just re-weighting the original query terms heuristically. Second, the expansion process leads to even greater improvements when multiple iterations are used. The second finding can certainly be the case in an expansion process where the user has some involvement, but in automatic query expansion – discussed in the next section – this is not necessarily the case. It is more likely that the query will drift off-topic.

In the following, we explain some of the mechanisms that can be employed to rank terms in order of the estimated usefulness to increase retrieval effectiveness. Efthimiadis evaluates eight term ranking methods for interactive query expansion (Efthimiadis, 1993, 1995), including the $F_4$ algorithms discussed in Section 3.1.3. Depending on the system, either the user is shown this list of terms and selects terms, while their choice is guided by the order of the terms presented, or the system will automatically select a number of top ranked terms, and the user interaction is restricted to identify relevant documents. Each of the terms is then added to the initial query.

Of the eight ranking methods Efthimiadis evaluates, he identifies the following methods as being good in ranking candidate terms. Note that we are using $N'$, the number of *seen* documents, rather than $N$, the number of documents in the collection, and similarly $f'_t$ rather than $f_t$ so as to not confuse relevance feedback with automatic query expansion. Efthimiadis used $N' = 25$ documents in his experiments.

### 3.2.1 Porter

The *Porter algorithm* (as described by Efthimiadis, 1995) essentially sorts terms by the number of documents amongst the top $|R'|$ documents that each occurs in:

$$w_t^{porter} = \frac{r'_t}{|R'|} - \frac{f'_t}{N'}$$

where $|R'|$ is the number of seen relevant documents, $r'_t$ is the number of seen relevant documents that contain term $t$, $f'_t$ is the number of all seen documents that contain term $t$, and $N'$ is the total number of seen documents. The second part of the equation above does not have a big impact, since it tends to vary less than the first part and the fraction is typically very small, compared to that of the first one.

### 3.2.2 Expected mutual information measure

The *expected mutual information measure* (EMIM) algorithm makes use of the incident contingencies given in Table 2.7 (van Rijsbergen, 1977, Harper and van Rijsbergen, 1978, van Rijsbergen et al., 1981):

$$w_t^{emim} = \sum_{t,w_q} \left( \Delta_{t,q} \times P(t, w_q) \times \log \frac{P(t, w_q)}{P(t) \times P(w_q)} \right)$$

where $t$ indicates presence (1) or absence (0) of a term, and $w_q$ indicates whether a document is relevant (1) or non-relevant (0). $\Delta_{t,q}$ indicates the value of a term as a relevance discriminator and evaluates to 1 if $t = w_q$ (that is, both $t$ and $w_q$ are in agreement – either the term is present and a document is relevant, or term $t$ is absent and the current document is non-relevant), or $-1$ otherwise. This gives the term weight either a negative or positive weighting. $P(t)$ is the probability of the current term being present, given the document at hand, $P(w_q)$ is the likelihood that the current document is relevant, and $P(t, w_q)$ is the likelihood that the particular term $t$ is absent or present while the current document is relevant or non-relevant, respectively.

Collapsing both the document and term dimensions together, while keeping track of relevance and term occurrence and absence, we get:

$$\begin{aligned} w_t^{emim} &= r'_t \times \log \frac{r'_t}{\frac{f'_t}{N'} \times \frac{|R'|}{N'}} \\ &- (f'_t - r'_t) \times \log \frac{f'_t - r'_t}{\frac{f'_t}{N'} \times \frac{N'-|R'|}{N'}} \\ &- (|R'| - r'_t) \times \log \frac{|R'| - r'_t}{\frac{N'-f'_t}{N'} \times \frac{|R'|}{N'}} \\ &+ (N' - f'_t - |R'| + r'_t) \times \log \frac{N' - f'_t - |R'| + r'_t}{\frac{N'-f'_t}{N'} \times \frac{N'-|R'|}{N'}} \end{aligned}$$

This formula is derived by considering each of the four cases in the incident contingency table in turn, that is a term is present and the document is either relevant or non-relevant (the first two cases), or a term is not present with the documents currently examined being relevant or non-relevant (the final two cases). The value of the two middle cases is subtracted from the weight for the current term $t$, since in either of those cases the term appears either in a non-relevant document, or does not occur in a relevant document.

### 3.2.3 WPQ

The *wpq* algorithm (Robertson, 1990) is an extension of the $F_4$ weighting function, given in Equation 3.4. The $F_4$ function is based on several assumptions of independence: first, that terms are distributed independently of each other in relevant and non-relevant documents; and second that expansion terms are statistically independent of terms appearing in the original query, or any other previous search formulation, such as an earlier iteration of an expansion algorithm (Efthimiadis, 1995). However, the set of relevant documents can be further divided into two groups: those that contains a term $t$ that is considered as a candidate term, and another group that does not. The wpq weighting scheme takes this further step into account:

$$w_t^{wpq} = w_t \times (p_t - q_t)$$

where $w_t$ is any previously derived term weight of term $t$ (Efthimiadis uses the $F_4$ weight, which we also make use of in the following equation), and $p_t$ as well as $q_t$ are the probabilities that the term occurs in a relevant or non-relevant document, respectively. Again, making use of the incident contingency table, $p_t$ and $q_t$ can be determined and the formulation above is expanded to:

$$w_t^{wpq} = \log\left(\frac{r'_t + 0.5}{|R'| - r'_t + 0.5} \times \frac{N' - |R'| - f'_t + r'_t + 0.5}{f'_t - r'_t + 0.5}\right) \times \left(\frac{r'_t}{|R'|} - \frac{f'_t - r'_t}{N' - |R'|}\right) \tag{3.6}$$

Algorithms based on the relevance weighting theory attempt to increase effectiveness for each part of a recall-precision curve, which means that relevant documents ranked anywhere in an answer set are expected to be pushed towards the top. This is attempted through the formulation of the probability ranking principle (Robertson, 1977). In effect, the $F_4$ scheme models this behaviour, and incorporates it in the above formula through the log factor. Incorporating infrequent terms that are just frequent amongst the top relevant documents may lead to higher precision at the top ranks, and frequent terms may lead to a higher recall; however, it may be that terms that are between those two extremes are more helpful for retrieval effectiveness. This is what the wpq algorithm aims to make use of.

Experiments by Efthimiadis show that wpq achieves better retrieval effectiveness than the $F_4$ measure. This confirms Robertson's original concern, that measures whose aim is to identify those terms already part of the query that discriminate best are not necessarily good at selecting new terms (Robertson, 1977).

### 3.2.4 R-LoHi

The *r-lohi algorithm* (Efthimiadis, 1993) ranks terms in the relevant set of documents by $r'_t$, that is the number of (seen) relevant documents that contain the term. Terms that are ranked equally are differentiated by the term frequency $f'_t$, from low to high frequencies, giving rise to the second part of the algorithm name – lohi (low-high). The resulting ranking could be formulated as follows:

$$w_t^{\textit{r-lohi}} = r'_t + \left(1 - \frac{f_t}{N}\right) \tag{3.7}$$

where $r'_t$ is the dominant part of the weight, while the second part will always evaluate to between 0 and 1 and so therefore never interchange the ranking positions of terms that have adjacent $r'_t$ values.

## 3.3 Pseudo relevance feedback

Interactive query expansion can significantly increase effectiveness (Magennis and van Rijsbergen, 1997), although on average – for non expert users – automatic expansion is more likely to lead to better performance (Ruthven, 2003). We consider how queries can be expanded automatically in this section.

Users are generally reluctant to provide information on whether they are satisfied with a particular document or not. Dennis et al. (1998) found that, even though users could successfully be trained to use novel expansion techniques, they were not likely to use those methods because of the cognitive load associated with using them. Ruthven (2003) and Taghva et al. (2004) recently found that algorithms are just as likely as non-expert users to correctly identify terms that enhance (or conversely, do not enhance) retrieval. Research activity has slowly shifted over the last two decades to *pseudo relevance feedback* (also known as *automatic*, *blind* or *ad-hoc relevance feedback*), even though users are also somewhat weary of using this type of feedback, as they lack control over the search process (Hancock-Beaulieu et al., 1999). Here, terms are sourced either from the document collection at hand or from external sources. If terms are heuristically found to be related to the topic of the original query, they are added to the query. User intervention is not required at any stage other than the formulation of the original query. In the early 1990s, this was the most popular strategy employed for query expansion (Buckley et al., 1994). It has since been the employed in much research on query expansion (see for instance Robertson and Walker, 1999, 2000). As with manual relevance feedback, the purpose of query expansion is to adapt an original query so that it is better suited to target relevant documents. It does so by making queries more similar to the documents it is targeting, thus aiming for a greater term overlap between the query and relevant documents.

Two main approaches are used to expand queries automatically. One group of expansion methods relies on knowledge structures such as thesauri, whereas others make use of an initial set of search results. Both classes are described separately in this section.

### 3.3.1 Expansion through knowledge structures

The group of expansion techniques that make use of knowledge structures are based on the following hypothesis (van Rijsbergen, 1979):

> "If an index term is good at discriminating relevant from irrelevant documents then any closely associated index term is also likely to be good at this."

When using knowledge structures, expansion terms are determined from pre-fabricated term dependency matrices or lookup tables, and no significant work has to be done during query time to expand queries.

#### Collection-independent knowledge structures

Efthimiadis (1996) lists the following as examples of *collection-independent knowledge structures*:

- Manually constructed, domain-specific *thesauri*. A thesaurus is a manually crafted or automatically composed list of synonyms or related concepts. It has also been referred to as a "treasury of words" (Foskett, 1997). A thesaurus is domain-specific, if it contains terms from predominantly one particular area, such as medicine or architecture.

- General-purpose thesauri, such as WordNet (Miller, 1995).

- Dictionaries and lexicons, such as Collins dictionary (Hanks, 1988).

Query expansion algorithms based on such references are also known as *external techniques* as they do not make use of corpus statistics in order to find candidate terms. During query time, queries are expanded simply by looking up related terms in the appropriate structures.

Voorhees (1994) expanded queries using WordNet, and found that individual queries that are not well formulated, or do not describe the underlying information need well, can be improved significantly. However, automatic expansion can degrade overall performance. Mandala et al. (1998) describe experiments employing automatically created thesauri as well as using WordNet to expand queries, but do not detail whether they achieve better effectiveness over a baseline.

Algorithms using this method are not researched widely any more and we do not describe related approaches any further in this thesis.

**Collection-dependent knowledge structures**

In contrast to the collection-independent techniques such as the hand-crafted thesaurus used in the previous class of expansion techniques, according to Efthimiadis (1996), *collection-dependent knowledge structures* include *pseudoclassification* (Salton, 1980) and automatically constructed thesauri, such as *association thesauri* (see for instance Jing and Croft, 1994, Qiu and Frei, 1993, or Gauch and Wang, 1997). Automatically constructed thesauri were explored as early as 1965 by Doyle (1965) and even earlier as *clumps* (Needham and Sparck Jones, 1964), which are closely related to the emergence of *automatic indexing* (Maron, 1961), but which are still used recently, for instance by Grefenstette (1994) and Evans and Lefferts (1993). Thesauri can be constructed in various different ways; Crouch and Yang (1992) for instance describe experiments where documents are clustered and term relationships are deduced from the clusters obtained. Although Lesk (1969) for instance uses a manually crafted thesaurus (discussed in the previous section) and concludes that these are superior to automatically ones, Crouch (1988) in contrast shows later that automatically constructed thesauri can work better than those that are manually constructed. Mandala et al. (1999) find that a combination of different kinds of thesauri is generally more useful for expansion than any one kind.

Expansion methods based on collection dependent knowledge structures are also referred to as *global analysis techniques* and can also rely on suffix stripping or stemming (see Section 2.2.3), or soundex coding (Hall and Dowling, 1980).

A good example of this class of expansion methods is *term co-occurrence*. Smeaton and van Rijsbergen (1983) discuss the use of maximum spanning trees, where term-to-term dependencies are captured in a tree structure, that specifies the most strongly statistically related neighbour for each term, and the use of index terms from known relevant documents (at this time, it was not usually the case that all terms in a document were used for indexing). They found that terms selected using any of these approaches degraded results more than using randomly selected expansion terms. Peat and Willett (1991) conclude that expansion terms found using term co-occurrence do not discriminate well between relevant and non-relevant documents as they tend to have relatively high document frequencies. Sanderson and Croft (1999) suggest the use of co-occurring terms found through *subsumption* for query expansion. A term (“term1”) is said to subsume a second term (“term2”), if “term2” occurs in a strict subset of the documents in which “term1” occurs. Therefore “term1” generalises the concept of “term2”. However, Sanderson and Croft do not report any experiments involving expansion.

A less elaborate grouping of terms is *term clustering*, where terms are grouped together along any pre-defined criteria, such as their frequency of use in a collection. While Minker et al. (1972) found that there was no considerable improvement using term clustering, and claim instead that using term clusters to expand queries can lead to significant degradations, Sparck Jones and Jackson (1970) reports that automatically obtained term classification can increase retrieval effectiveness, especially when using infrequent terms.

*Latent semantic indexing*, also known as *latent semantic analysis*, builds on both term co-occurrence and term clustering and was first described by Dumais et al. (1988) and later by Deerwester et al. (1990) in more detail (see also Berry et al. (1995) for a survey of related papers). It was designed to automatically assign high quality keywords to documents, but has later found application for query expansion.

Latent semantic indexing creates an index of collection terms, after removing common adjectives and terms that either appear in all documents or only in a single one. Terms are then organised in a multidimensional space, where they are effectively clustered together so that terms that frequently appear close to each other are located close to each other. During query evaluation, a term can then easily be expanded using closely related terms, while the distance between each expansion term and the original query term conveniently gives an indication of an appropriate weighting of each expansion term.

Latent semantic indexing has some shortcomings. One is the difficulty in determining the optimal number of dimensions in which single value decomposition should occur, as this changes from collection to collection. For instance, Dumais (1994) used $k = 200$ to $350$ dimensions, whereas Dumais et al. (1988) used only $50$ in their original work. Since the single value decomposition algorithm is highly complex, with cost $O(N^2 \times k^3)$, using trial and error to determine a good number of dimensions is prohibitive. This cost also makes it difficult to use latent semantic indexing in a setting where new documents are added to a collection since all term relations would need to be re-calculated. Another problem is that automatically linking terms based on semantic relations derived from statistics is not failsafe, particularly since word order or syntactic relations are not used (Landauer et al., 1998). For instance, although the terms “hot” and “cold” are opposites, on another level they both describe the temperature of an object. Because one would expect to often find them in close proximity to each other, this would be reflected in the matrix created by a latent semantic analyser. However it would be wrong to use one as a synonym for the other. This is perhaps a problem with any automatic approach to classifying terms. Furthermore, Park and Ramamohanarao (2004) found that using latent semantic indexing for information retrieval is inferior to the vector space model for all but one of the collections and query sets they used for testing.

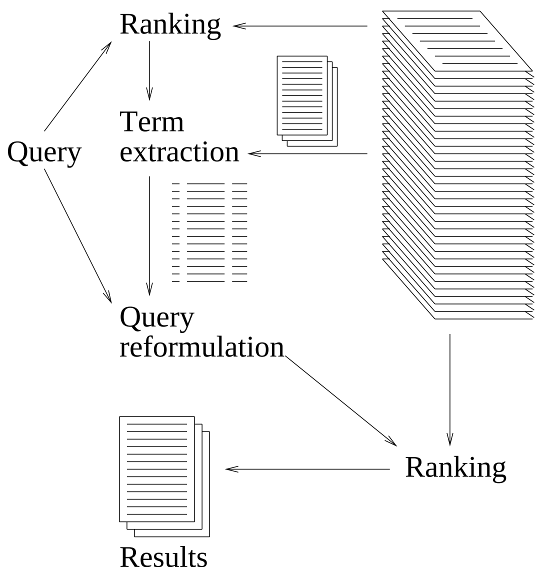


*Figure 3.1: Local analysis query expansion.*

Latent semantic indexing has been used for automatic query expansion with moderate success (Dumais, 1994). Ozcan and Aslandogan (2004) also experimented with latent semantic indexing and query expansion, but results were inconclusive. Dumais (1990, 1991) tested manual relevance feedback on a number of small scale test collections with a relative improvement in average precision of 67%. Park and Ramamohanarao (2004) achieve reasonably good expansion results using expansion terms derived from a latent semantic index, but using the vector space model to rank documents.

### 3.3.2 Expansion based on initial search results

The problem with the approaches discussed so far in this section is that only the context of individual query terms can be captured and used to find expansion terms.

Guo et al. (2004) remark that using synonyms for expansion increases recall as more documents are identified that may be relevant to the topic the user is interested in; however, precision is decreased as the inclusion of documents is too indiscriminant and the expansion terms may be too ambiguous to help differentiate relevant from non-relevant documents.

One way to overcome the problem of mismatched terms while not being able to rely on the user for feedback, is to retrieve an initial set of results using the original query. A certain number of terms is selected from all terms that occur in the top documents, taking into account the frequency of those terms amongst the set of retrieved documents as well as collection statistics. The terms with the highest score (which is discussed in greater detail in Section 3.3.2) are added to the query, with a reduced weighting. The modified query is then run again. This method is called *local analysis* and was first suggested by Croft and Harper (1979). A diagram detailing this process is shown in Figure 3.1.

Of the various approaches to automatic query expansion, those that are reliant solely on external knowledge structures are least successful (Mandala et al., 1999, R. Mandala, 2000) and the use of thesauri and global analysis mechanisms in general have been shown to be less successful than local analysis (Xu and Croft, 2000). As noted earlier, the difference in effectiveness is based on the problem that a term can take on different meanings, depending on the context that it appears in. This is particularly a problem for very short queries (Sanderson, 1994). Local analysis methods inherently disambiguate word senses better, as expansion terms are sourced from documents that are retrieved with the whole query, rather than individual query terms. This is one of the reasons why we examine this method for the remainder of this thesis and will refer to this process as query expansion, to the exclusion of any of the other approaches discussed previously. In particular, we follow the approach of Robertson and Walker (1999) when describing term selection and weighting.

While local analysis in general performs much better than other expansion approaches, the method also suffers from some shortcomings. One major disadvantage of local analysis methods, from the point of view of search providers, is that they are generally much less efficient than global methods, since terms occurring in the *local set* of documents need to be assessed. The local set is made up of a number of top documents that are assumed to be relevant and that are chosen to base the term selection heuristics on. This set of top ranked documents needs to be retrieved through a series of expensive disk accesses. The efficiency problem is explored in detail in Chapter 6.

Another problem with this approach is that queries have an increased risk of query drift, as the top ranked documents are assumed to be relevant, while they may in fact not be (Croft and Harper, 1979). While query drift was reduced in more recent years by Mitra et al. (1998), who used term co-occurrence and Boolean filters to place restrictions on the local set of documents, about 25% of queries are still affected by query drift.

In the following sections, we examine different selection heuristics for selecting terms and describe how we assign weights to expansion terms in this thesis. Note that we use the symbols as in Table 3.1, however $R'$ in this context is the local set, or the set of documents that is *assumed* to be relevant.

**Term selection measures**

In local analysis, after choosing the term selection and weighting functions, the size of the local set can be varied, as can the number of terms added to the query. Changing the parameters can have a large impact on retrieval effectiveness; we examine the impact of changing the size of the local set and the number of terms to add to the query in detail in Chapter 4.

Another parameter that can be varied is the minimum number of documents in which a term has to appear in, in order to be considered as an expansion term. In Figure 3.2 we examine how effectiveness varies with different minimum thresholds for the local term frequency. In this thesis we use one of three

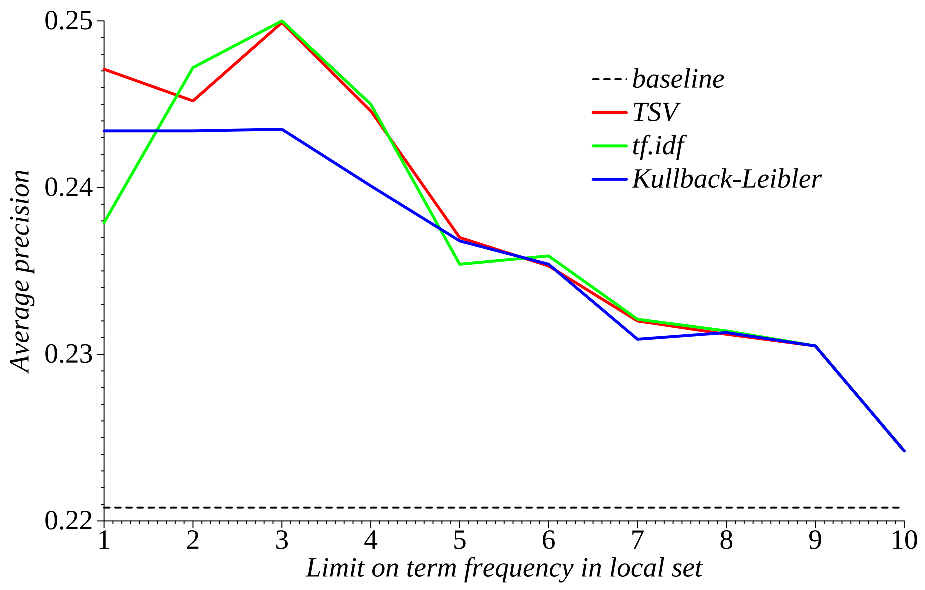


*Figure 3.2: Various selection measures and the impact of limiting the number of documents in the local set that contain a candidate term. Results are shown for TREC 8 data.*

selection measures, described in the following. It is reasonable to expect that setting $r'_t$ to a minimum of 2 would yield good results, as this means that the evidence of a particular term being a good candidate term is corroborated by two documents, rather than just one. Interestingly, we found $r'_t \geq 3$ seems consistently good (mean average precision peaks for all three selection measures at this cutoff) for this particular query set and collection, whereas $r'_t \geq 2$ does not work as well. Since a particular query set and collection (namely TREC 8) is used, this is not a conclusive experiment giving a definitive answer as to which of the selection measures is best, but rather it is a depiction of the impact that limiting the minimum local term frequency can have.

A variation on the term selection measures presented below is given by Hoashi et al. (1999), who measure the "word contribution". In this approach each term in a particular top-ranked document is assigned a score related to how much this term contributes to the similarity of the document and the query. Terms with high scores, usually those that appear in the query and the document under observation, indicate that the term contributes heavily to the similarity. Low scores on the other hand mean that a term contributes little to the similarity. Hoashi et al. assume that a term with a low score that appears in one of the top-ranked documents would be a good expansion term since it is presumably on the same topic as the query, but is different to the terms the user employed in the original query.

We now describe the three selection measures used in this thesis.

**Term selection value** The *term selection value* (*TSV*) was first proposed in this form by Robertson and Walker (1999), where $|E|$ terms are chosen from the top $|R'|$ ranked documents:

$$TSV_t = \left(\frac{f_t}{N}\right)^{r'_t} \times \binom{|R'|}{r'_t}$$

where $f_t$ is the number of documents in the collection in which term $t$ occurs in, $N$ is the total number of documents in the collection, $|R'|$ the size of the local set $R'$, and $r'_t$ is the number of documents in the set $R'$ in which term $t$ occurs. Note that the second factor in the formula above can be calculated as $\frac{|R'|!}{r'_t!(|R'|-r'_t)!}$. We use this scheme in our experiments for ranking candidate terms unless stated otherwise.

Robertson and Walker (1999, 2000) suggest using the following threshold to determine which terms to use for expansion:

$$TSV_t > \frac{1}{|V| \times e^c}$$

where $|V|$ is the size of the vocabulary $V$, that is the number of unique terms occurring in the collection. A value of $c = 0$ means that the threshold is 1/$|V|$. A value of $c = 4.6$ (or a threshold of $1/(100V)$) "corresponds to having a less than 1% chance of accepting any noise terms" (Robertson and Walker, 1999). The authors suggest that a slightly negative value of $c$ might be suitable. However, we do not use this threshold in our work as we found in unpublished experiments that using a cutoff is of varying effectiveness, improving retrieval for some collections but not for others. Our finding was confirmed through unpublished correspondence with Chris Buckley, who told us that the use of thresholding as suggested by Robertson and Walker is not recommended.

We found that the term selection value provides a good function to rank candidate values and we use this in all experiments in the remainder of this thesis, except where indicated otherwise.

**Kullback-Leibler divergence** Another ranking scheme for candidate terms is the *Kullback-Leibler divergence* (*KLD*). As discussed in Section 2.4.3, the Kullback-Leibler divergence specifies the distance between two probability densities. In the context of local analysis query expansion, each term in the local set of documents is assigned a value associated with the relative rareness of a term in the current set as opposed to the whole collection. In the context of local analysis, the Kullback-Leibler divergence can be calculated as (Croft, 2000, page 154):

$$KLD_t = \frac{r'_t}{|R'|} \times \log\left(\frac{r'_t}{|R'|} \times \frac{F + 0.01 \times |V|}{F_t + 0.01}\right)$$

where $F_t$ is the total number of occurrences of term $t$ in the collection and $F$ is the combined total number of occurrences of all terms in the collection.

The resulting weight of terms that occur relatively often in the local set in contrast to the rest of the collection is higher than that of terms that appear as often within the local set as their term frequency suggests. Conversely, a lower weight is associated with terms that are relatively rare in the local set.

We found that the ranking of candidate terms based on this divergence is often equivalent to using the term selection value, however, in particular applications, it does not work as well (see for instance Figure 3.2).

***tf.idf*** A third heuristic for ranking candidate terms is to use their *tf.idf* value. As with the term selection value, the *tf.idf* is dependent on both the presence of a term among the top ranked documents (their *term frequency* or *tf*), and its overall collection frequency (*inverse document frequency*, or $idf$). The term frequency is usually formulated as a variant of $f_{d,t}$ or $\log(1 + f_{d,t})$, whereas the inverse document frequency is usually mathematically described as $1/f_t$, $N/f_t$, or $\log(N/f_t)$. Together they may build the product:

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

where $f_{d,t}$ is the frequency of term $t$ in document $d$. Note that the $tf$ component is formulated slightly differently in the context of local analysis, since the frequency of a term in the local set has to be taken into consideration:

$$tf.idf = \log\left(1 + \frac{\sum_{d\in R'} f_{d,t}}{|R'|}\right) \times \log\left(\frac{N}{f_t}\right)$$

The *tf.idf* measure is often used to determine the importance of a particular term in a particular document. The vector space model, the probabilistic model, and the language modelling approach to ranking documents (detailed in Section 2.4) all rely on some variant of this formulation.

Before the 1990s, $\log$s were not used for the formulation of either the term frequency or the inverse document frequency. Later $\log$ normalisation was used to improve retrieval in the early TREC collections. For VLC2, the 100 GB TREC collection, taking the $\log$ of $\log$ can provide improvements. The usage of $\log$s is also referred to as *impact-normalisation*; it addresses the practical problem of collection growth, but no justification exists in principle.

Although we tried the *tf.idf* measure in various experiments, we found that terms were often better ranked according to their term selection value or using the Kullback-Leibler divergence, even though it seems competitive with the term selection value in Figure 3.2.

### Term weighting: Robertson/Sparck Jones weight

In the context of this thesis, we use the Robertson/Sparck Jones relevance weight (Robertson and Sparck Jones, 1976, Robertson and Walker, 1999) for weighting expansion terms (see also Section 3.1.3, where this weight is used for ranking candidate terms, and Section 2.4.2, where the relevance weight is used in the derivation of the Okapi formulation). This weight is used for expansion terms, instead of their weight as determined by the Okapi formula. It is is calculated as follows:

$$w_t = \frac{1}{3} \times \log \frac{(r'_t + 0.5)/(f_t - r'_t + 0.5)}{(|R'| - r'_t + 0.5)/(N - f_t - |R'| + r'_t + 0.5)}$$

The dampening factor of 1/3 was recommended by unpublished correspondence with Stephen Robertson. It de-emphasises expansion terms and helps to prevent query drift.

Salton and Buckley (1988) give an overview of term weighting functions used for retrieval in general. In the context of query expansion though, term weighting schemes usually revolve around term ranking functions, unlike the approach detailed above, where the weighting function is distinct from the ranking function (Robertson and Walker, 1999, 2000). For example, in TREC tasks previous to 1999, Robertson and Walker used the Robertson/Spark-Jones weight not only to weight expansion terms, but also to rank them (Robertson et al., 1998). Similarly, Mandala et al. (1998) weight and rank their expansion terms according to their similarity to the query.

### 3.3.3 Combinations of different feedback mechanisms

Lee (1998) combines the results from multiple expanded versions of an original query, using for instance automatic versions of some of the methods described in Section 3.1. Their results show that many different combinations achieve higher average precision than most single expansion methods.

The most notable approach that combines both local and global analysis techniques is *local context analysis* by Xu and Croft (1996, 2000). At first a local set of documents is retrieved, the documents of which are subsequently broken up into passages of 300 terms each. Then, treating the local set as the document collection and the passages as the documents within that collection, global analysis is performed, by building collection-dependent knowledge structures from the passages. Terms for expansion are ranked by their context score, which is based on the work by Qiu and Frei (1993) and arrived at as follows.

The degree of co-occurrence (*co-degree*) is measured by the number of passages that contain both the query term $t$ and the concept $c$, where concepts are either nouns or noun phrases appearing in the local set:

$$\textit{co-degree}(c,t) = \log_{10}(co(c,t)+1) \times \frac{idf(c)}{\log_{10}|R'|}$$

where $|R'|$ is the number of documents in the local set. The context $co(c,t)$ is calculated by:

$$co(c,t) = \sum_{p \in R'} (f_{p,c} \times f_{p,t})$$

where $f_{p,c}$ and $f_{p,t}$ are the number of occurrences of the concept $c$ or term $t$ in passage $p$, respectively. $idf(c)$ is calculated as follows:

$$idf(c) = \min\left(1.0, \frac{\log_{10}\frac{N}{f_c}}{5.0}\right)$$

where $f_c$ is the global concept frequency, that is, the number of documents that contain concept $c$. Finally the degree of co-occurrence of all query terms is combined ($\delta$ is used as part of a smoothing technique):

$$f(c,q) = \prod_{t \in q} (\delta + \textit{co-degree}(c,t))^{idf(t)}$$

Xu and Croft (2000) find that breaking up the local set of documents into passages or just performing conventional local analysis works equally well. However, comparing their work against a local analysis baseline that is based on the method suggested by Buckley et al. (1994), Xu and Croft show in a series of experiments that local context analysis achieves consistently higher mean average precision than local analysis query expansion. As the addition of passages does not contribute much to the performance, they conclude that the improvement achieved through local context analysis is mainly due to using a better term selection metric.

### 3.3.4 Standard language modelling approach to automatic query expansion

The standard approach for query expansion used in the framework of language modelling (see Section 2.4.3) is to use the model comparison method, where the document model $\theta_{t,d}$ biased on term $t$ is calculated for each document $d$ in the local set $R'$ (see for instance Larkey et al. (2002)). Each term in that set is then ranked according to the sum of divergences between its prevalence in each document it occurs in and the importance of the term in the whole collection (see Table 2.8 for language modelling notation):

$$\begin{aligned} score_t &= \sum_{d \in R'} \left( \theta_{t,d} \log \frac{\theta_{t,d}}{\theta_{t,C}} \right) \\ &= \sum_{d \in R'} \left( P(t|d) \log \frac{P(t|d)}{P(t|C)} \right) \\ &= \sum_{d \in R'} \left( \left( \lambda \frac{f_{d,t}}{|d|} + (1-\lambda) \frac{F_t}{F} \right) \log \frac{\lambda \frac{f_{d,t}}{|d|} + (1-\lambda) \frac{F_t}{F}}{\frac{F_t}{F}} \right) \\ &= \sum_{d \in R'} \left( \left( \lambda \frac{f_{d,t}}{|d|} + (1-\lambda) \frac{F_t}{F} \right) \log \left( \lambda \frac{f_{d,t}}{|d|} \frac{F}{F_t} + 1 - \lambda \right) \right) \end{aligned}$$

where the probabilities $P(t|d)$ and $P(t|C)$ are calculated as derived for Equation 2.18.

Seeing that $\frac{\theta_{t,d}}{\theta_{t,C}}$ leads to very small scores for terms that occur frequently in the collection, the Kullback-Leibler divergence seems to be a good candidate for the ranking of possible expansion terms. This is similar to the *tf.idf* weighting measure described in Section 3.3.2.

This approach – albeit unsmoothed – is also employed by Carpineto et al. (2001), using classical probability formulations, rather than the language modelling framework. Zhai and Lafferty (2001*a*) also use the Kullback-Leibler divergence. Further, they propose a second approach that calculates the language model of an improved query for use with the query likelihood, or model comparisons, approach:

$$\log P(\mathcal{F}|\theta_q) = \sum_{d \in \mathcal{F}} \sum_{t \in V} \left( f_{d,t} \times \log \left( (1-\lambda) P(t|\theta_q) + \lambda P(t|C) \right) \right)$$

where $\theta_q$ is the language model of the query that is estimated from the feedback documents $\mathcal{F}$. They achieve good improvements using this method, and consistently outperform the Rocchio method, detailed in Section 3.1.2.

There are also several other methods for expanding queries in this framework. Berger and Lafferty (1999) for instance propose *statistical translation* for the purpose of matching the models of a query and target documents; their method could easily be extended to incorporate query expansion.

## 3.4 Summary

In this chapter, we presented an overview of query reformulation schemes, such as relevance feedback and interactive query expansion, and examined automatic query expansion in detail.

In particular, we explained that a query is optimal if it ranks all relevant documents before all non-relevant documents, and how such an ideal query could be constructed within the framework of the vector space model. Since full knowledge of the relevance of all documents is needed, this is only of limited practical usefulness; in real retrieval environments, such knowledge is not available. We therefore discussed how query optimality can be approached using relevance feedback, where users indicate the relevance of documents that they have see so far. Incrementally refining a query using such a technique (for instance using the Rocchio method) will, after some iterations, lead to a query that is close to optimal, depending on whether an optimal query is possible at all (if the set of relevant documents is not sufficiently distinct from the set of non-relevant documents). We also discussed relevance feedback techniques other than the Rocchio method, and considered how effectiveness measurements can take into account the fact that actual relevance judgements influence the document rankings, and therefore skew results.

We then discussed interactive query expansion, where the user's interaction with the system – in contrast to relevance feedback – is limited to judging the usefulness of candidate terms that are suggested by the search system. We explained the most cited algorithms used for ranking candidate terms.

Finally we discussed the concept of blind or automatic relevance feedback, where the user does not interact with the system at all, apart from submitting an initial query, and eventually viewing the results. No interaction is required during the expansion stage. Automatic query expansion can be carried out using knowledge structures, or it can be based on initial search results. When drawing expansion terms from knowledge structures, collection-independent term sources such as dictionaries, or manually crafted domain specific thesauri, can be used. When using collection dependent structures, terms are drawn from automatically constructed thesauri.

The method analysed and used for experiments in this thesis is local analysis query expansion, where a set of results is ranked from the initial query. Candidate expansion terms are then ranked based on some heuristics, and the highest ranked terms are selected and added to the original query. Finally we considered some of the alternative term selection measures that can be employed.

Chapter 4

# Local Analysis: Effects of Varying Parameters

Search systems are the principal mechanism used for finding documents on the web (Schwartz, 1998). These engines use information retrieval techniques to match queries, typically expressed as a series of terms, to the documents that are judged the most likely to answer the users' needs. When queries are well formulated, typically consisting of topic-specific keywords that together specify the information need with low ambiguity, search engines can return good matches in the top-ranked documents.

However, queries are resultant from a complex thought process undertaken by a user (as described in Section 2.1) and are often not well-formulated (as discussed in Chapter 3). They may be ambiguous, insufficiently precise, or use terminology that is specific to a country or dialect – recall the example used in Chapter 1: whereas the term “wrench” is commonly used in the United States, a “spanner” is the UK equivalent. The majority of queries posed to search engines are brief, with about 60% of queries containing only one or two key terms, while the average query length in 2001 was 2.6 terms (Spink et al., 2002). These problems are more acute when the user wishes to find large numbers of relevant documents: all reviews of a particular movie, for example, or all commentary on a particular topic. Many relevant documents may not contain the terms used in the query.

In this chapter, we investigate the performance of one successful approach to query expansion, as used in the Okapi/Keenbow system (Robertson and Walker, 1999, 2000). In common with all local analysis query expansion methods, this approach requires two parameters, namely the number of documents in the initial ranking and the number of expansion terms. Settings for these parameters were determined through experiments on a particular test collection, and have since then been used in many experiments without variation.

Our results show that it is far from clear that these parameter choices are optimal. Using comprehensive experiments on a test collection, we have investigated both average effectiveness and per-query effectiveness for a wide range of parameter choices. These results show that other choices of values can give higher effectiveness, but that no fixed choice is robust: entirely different values are preferable for other collections. Worse, the best choices vary wildly per query. Current approaches to query expansion are not well founded.

However, our results also show that the performance of query expansion has significant scope for improvement: individually tuning parameters to queries can give much better performance than the use of fixed values. We hope to be able to develop a method for predicting parameter values, and thus obtain greater effectiveness than is available with current methods.

Preliminary versions of this chapter appeared as Billerbeck and Zobel (2003, 2004*a*).

## 4.1 Experimental setup

To ensure that the measurements of effectiveness and efficiency – the latter being important for the work in later chapters – were realistic, we used *Lucy* (experiments in Chapter 5 were also based on Lucy, but experiments in Chapters 6 and 7 were conducted using *Zettair*, a later version of the same software; differences between Lucy and Zettair are described in Section 6.3). Lucy/Zettair is an open source search engine being developed at RMIT University by the Search Engine Group. The primary aim in developing Lucy/Zettair is to test techniques for efficient information retrieval. It was used at TREC the first time in 2004 (Billerbeck et al., 2004) and is available from http://www.seg.rmit.edu.au/.

We used the Okapi BM25 measure as discussed in Section 2.4.2 and the Robertson and Walker (1999, 2000) method of expanding queries (see Section 3.3.2). We confirmed the suitability of the dampening factor mentioned in Section 3.3.2 in limited experiments (see Figure 4.1).

Differences in details such as parsers, stopping and stemming, can have a marked impact on effectiveness, making it difficult to exactly reproduce reported experimental results. Preliminary experiments (not reported here) suggested that using passages in place of whole documents does not improve effectiveness, however it might be an interesting topic to revisit, as other work has indicated that passages can increase effectiveness (Kaszkiel and Zobel, 2001, Xu and Croft, 2000). Some systems stem terms aggressively, or some index the content of HTML tags (or of selected tags) while others do not; and there are numerous other variations.

In our experiments, we use case folding, but do not make use of stemming. We also stop queries, using a stoplist of 477 terms, including all single alphabetical characters. Our indexes contain term offsets for the work in this and the next chapter, but not for Chapters 6 and 7 (this and other variations to the experimental setup for those two chapters are discussed in Section 6.3). Inverted lists are stored on disk in a variable-byte compressed format (Scholer et al., 2002), where document identifiers are stored using d-gapping (Witten et al., 1999, page 115). The vocabulary is held in memory during query evaluation and we use the *continue* strategy (see Section 2.4.4) for efficient query evaluation.

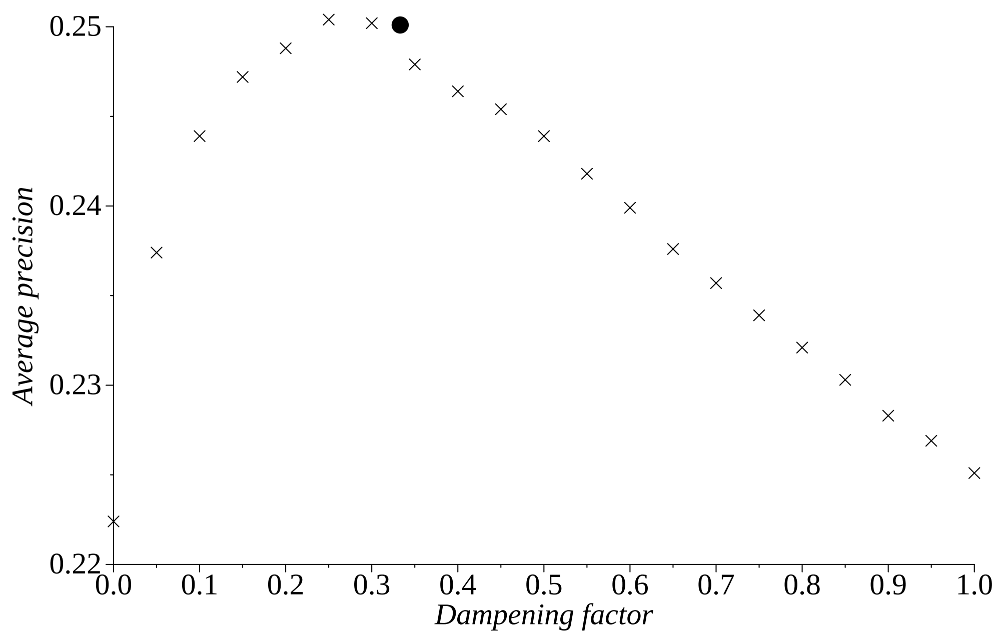


*Figure 4.1: The influence of the dampening factor on effectiveness is shown for TREC 8 data. It can be seen that the specially marked default factor is an acceptable choice.*

Even though all these different settings and optional use of techniques may have a considerable impact on the effectiveness (and efficiency) of experiments, our results with Lucy and Zettair are highly consistent with those reported by Robertson and Walker (1999, 2000), and we are therefore confident of our implementation.

For all our experiments, unless stated otherwise, we use the TREC testbed (see Section 2.6.2) and run only the title field for the initial query, as this is most representative of a typical web information retrieval task.

## 4.2 Where query expansion fails

Using local analysis, Robertson and Walker (1999, 2000) have shown that the approach as outlined above improves effectiveness by about 10% over an already high Okapi baseline. A fixed number of $25$ terms is appended to the query, and although thresholds for the TSV (see Section 3.3.2) have been trialled (Robertson and Walker, 2000), these were only of marginal effectiveness on a small number of data sets.

In the Okapi work, fixed values were used for key parameters. In some of our experiments (described in the following), we also used fixed values for the number of documents in the initial ranking ($|R'|$) and the number of expansion terms ($|E|$). These values (10 and 25 respectively) were chosen by experiments on a particular collection, and were observed to give an overall improvement in effectiveness. In other experiments, a fixed value was used for $|R'|$ and a fixed upper bound was imposed for TSV.

In our experiments, we have primarily used TREC disks 4 and 5 (see Section 2.6.2 for details of TREC collections) and the title field only of queries 401–450, which is the data used at TREC 8 in 1999. In some additional experiments, we have used the TREC 9 10-gigabyte web track with queries 451–500.

Using the standard parameters of $|R'| = 10$ and $|E| = 25$, query 405 *cosmic events* is expanded to:

```
cosmic events asteroid asteroids astronomers astronomical astronomy
cobe comet comets cosmological data dust earth issledovaniya jovian
kosmicheskiye nasa orbits particles planet scientists solar space
spaceguard sunless telescope
```

Average precision for this query increases from 0.061 to 0.216, and the recall (for the first 1,000 ranked documents) increases from 13 to 31 documents. Most of the expansion terms are specific to that topic and narrow the search; the original query is fairly general. Due to the nature of local analysis, a particular topic is selected from all possible topics one could deduce from the original query. In this case the correct topic was expanded and the query did not drift (see Chapter 3). From this example it is apparent that we did not use stemming. An explanation of why stemming might not be helpful for query expansion is that the expansion process automatically identifies variant forms of terms, and simplistically introducing further variants is unlikely to be helpful.

In contrast, consider query 440, *child labor*, for which the expansion terms are:

```
child labor age ballboys batboys clac detrimental dol employed
employers employment flsa hazardous hours minors nonagricultural
nonschool occupational occupations olds permissible reg school
subpart wecep workers young
```

Average precision for this query falls from 0.093 to 0.003. In this case, query expansion degrades the query, by introducing low-relevance and general terms. In absolute terms, the decline in average precision is not large, but it is certainly clear that for these parameters query expansion is unhelpful.

## 4.3 Parameterisation of query expansion

As explained in Section 3.3.2, local analysis query expansion is a highly parameter-dependent process. Local analysis can be split into two steps: the first is to identify relevant documents from which expansion terms are drawn, and the second is to rank possible terms by their perceived usefulness. Hence, parameters for the number of documents used for the local set $|R'|$ and the number of terms appended to the initial query $|E|$ need to be chosen (as in the previous chapter, we use $R'$ for the local set to indicate that we are using pseudo relevance feedback; that is, the relevance of the documents in $R'$ is only *assumed*).

In previous work, these parameters have often been chosen arbitrarily, and, for different experiments, different parameters were used. In many experiments, only a particular parameter pair is used, while the choice of expansion parameters is seemingly peripheral to the experiments performed. Buckley et al. (1994) for instance use 30 documents to expand with 500 terms and an additional 10 phrases for a TREC run. In different experiments Xu and Croft (2000) make use of 30 and 70 concepts for expansion using local context analysis (described in Section 3.3.3). Voorhees and Harman (1998) state that each group participating at TREC use different parameter settings for a particular ad-hoc track. Harman (1992*b*) uses 20 terms for her experiments, and Haines and Croft (1993) use around 150 terms. Kwok (2002) measures the term co-occurrence of candidate terms in small text windows for re-ranking and query expansion. For this successful method (achieving about 10% improvement in conjunction with re-ranking) he uses a fixed number of 40 terms for every query.

In other work, one parameter is held constant, while different settings are explored for the other parameter. For example, Harman (1988) reports that the number of expansion terms is dependent on specific collections. Magennis and van Rijsbergen (1997) use different numbers of expansion terms for interactive query expansion while keeping the number of documents to 20. Although not optimal, this provides a more stable framework for expansion; see Ruthven and Lalmas (2003, page 35). Conversely, Mitra et al. (1998) vary the size of the local set from 50 to 500 documents.

Others try different settings for both parameters using a trial and error approach: Lam-Adesina and Jones (2001) establish various baselines for their expansion method based on human-readable document summaries, using a combination of 5, 10, and 20 documents and terms. Once they establish their baseline, they try their various algorithms with both 5 and 20 documents as the local set size. Lundquist et al. (1997) experiment with varying the number of phrases and terms used for expansion and – after tuning – conclude that 20 phrases and 50 terms are optimal for the TIPSTER collection (Harman, 1992*a*), which is in contrast to our results presented in the following.

The problem of choosing good parameters is already evident in the earlier work on interactive query expansion, described in Section 3.2, where test subjects were asked to add a certain number of query terms to the query; see for instance Magennis and van Rijsbergen (1997).

## 4.4 Exploring query expansion

As local analysis query expansion is highly dependent on two parameters $|R'|$ and $|E|$, it is interesting to find ways of making local analysis more robust. As an initial step, in this chapter we explore the choice of values for those parameters.

### 4.4.1 Collection dependency

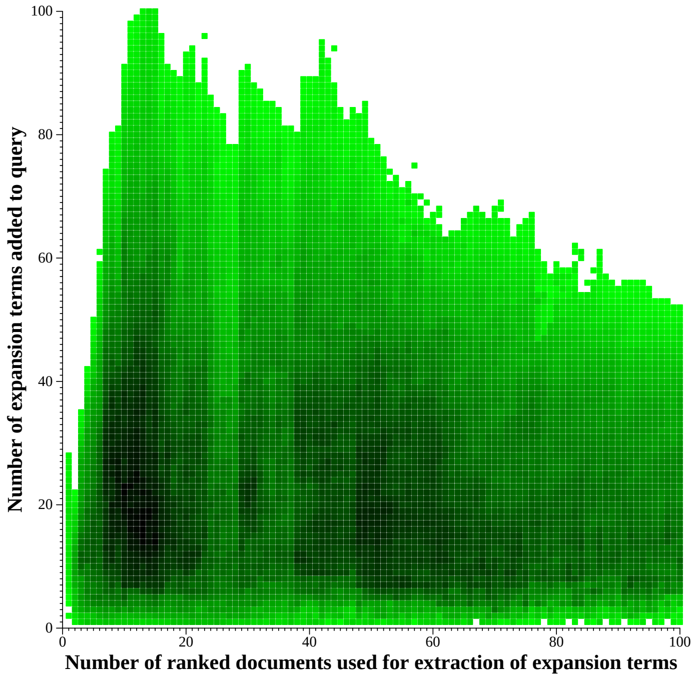
In our first experiment, we explore all combinations of $|R'|$ and $|E|$ from 1 to 100, that is, we explore the effect of varying the number of documents used for expansion terms and the number of expansion terms chosen. For each of the 10,001 combinations (including the baseline of no expansion), we run all 50 queries from TREC 8 and measure mean average precision.

Results are shown in the top graph in Figure 4.2. The graphs show all 10,000 data points (the baseline of no expansion at coordinate (0,0) is not assigned a colour). White areas indicate parameter pairs for which expansion degraded effectiveness on average for this query set. The darker the data point, the greater the increase in mean average precision compared to the baseline. Thus, roughly, the greatest improvement can be seen for $|R'|$ between 8 and 16 documents, and for $|E|$ between 7 and 42 terms. Choosing $|R'|$ of around 50 documents and keeping the number of expansion terms lower than 40 also gives good results. The vertical stripes correspond to areas where mean average precision for a single query was dramatically improved (or degraded) by retrieval of an additional document with excellent (or awful) expansion terms. On the other hand, sensitivity to the number of expansion terms is comparatively low.

The original expansion parameters of $|R'| = 10$ and $|E| = 25$ are just within the dark “best” area. The mean average precision at this point is 0.254, up from 0.216 with no expansion. These are not quite the best choices; parameters of $|R'| = 13$ and $|E| = 15$ give slightly better overall results of 0.260. However, the original values (both the parameter settings, as well as the result of mean average precision) are impressively close to these settings. Contrasting results are shown in the bottom graph in Figure 4.2, for the TREC 9 WT10g collection, comprising of 10 Gb of web data. Expansion is on average much less successful, with little overall improvement observed. The best effectiveness is at $|R'| = 98$ and $|E| = 4$ (which is not in the neighbourhood of other successful points), and was only a slight improvement on effectiveness without expansion. Indeed, even at this point most queries perform better without expansion.

A possible explanation for why local analysis query expansion works with the TREC 8 collection, but not with the TREC 9 collection, is that TREC 8 consists of newswire articles, whereas TREC 9 is made up of web data. The former consists of carefully reviewed text with a relatively controlled vocabulary on specific topics, whereas web data is usually interspersed with links and other information that is more diverse. The language used in web data is often erratic and text commonly serves only to describe or otherwise accompany images, tables, or other items that are infrequent in newswire data.

Although this approach to query expansion does not lead to higher effectiveness on TREC 9 data, other methods are more successful. In Chapter 5 we show that using query associations as a source of expansion terms leads to greater accuracy. On the other hand, this method is not applicable in an environment like that of TREC 8, for which there is no suitable large collection of queries from which query associations could be constructed. This seems to suggest that no one particular method of query expansion can be used to reliably expand queries on all possible collections.



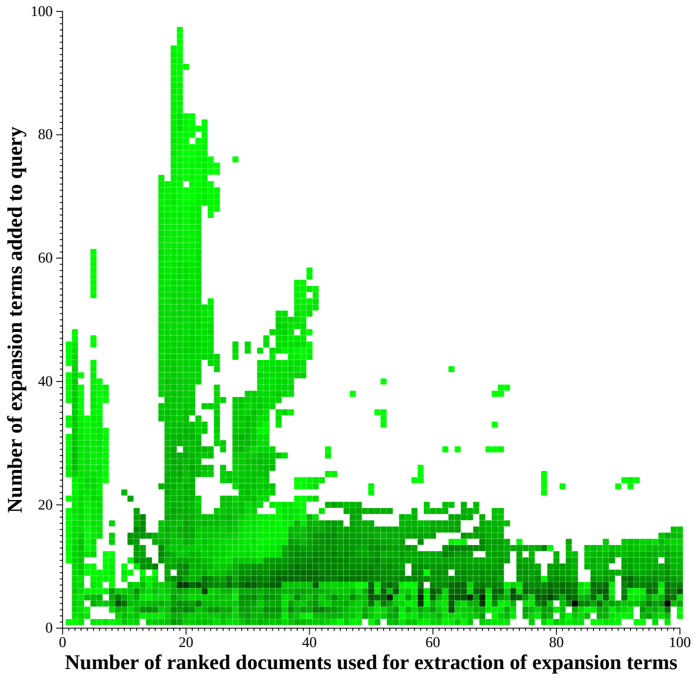


*Figure 4.2: Mean average precision for each combination of number of documents ($|R'|$) and number of expansion terms ($|E|$). Top: TREC 8 data, disks 4 and 5. Bottom: TREC 9 10-gigabyte web data. Dark areas: high mean average precision. Light areas: low mean average precision. White: mean average precision worse than or equal to no expansion.*

### 4.4.2 Query variability

Another problem with query expansion is highlighted in Figure 4.3, which shows the parameters at which expansion is beneficial for two queries. For query 405, it can be seen that, in terms of average precision, expansion outperforms the unexpanded query for almost any parameter pair (in the region of the graph). For query 440, however, expansion is only beneficial at very limited numbers of parameter pairs; for almost all combinations of those two parameters, the query is degraded.

However, most of the difficult-to-expand queries do have sets of parameters at which expansion is beneficial. Although one of the ad-hoc queries from TREC 9 does not benefit from expansion for any combination of parameters, for all other queries at least some parameter pairs exist, for which average precision is improved.

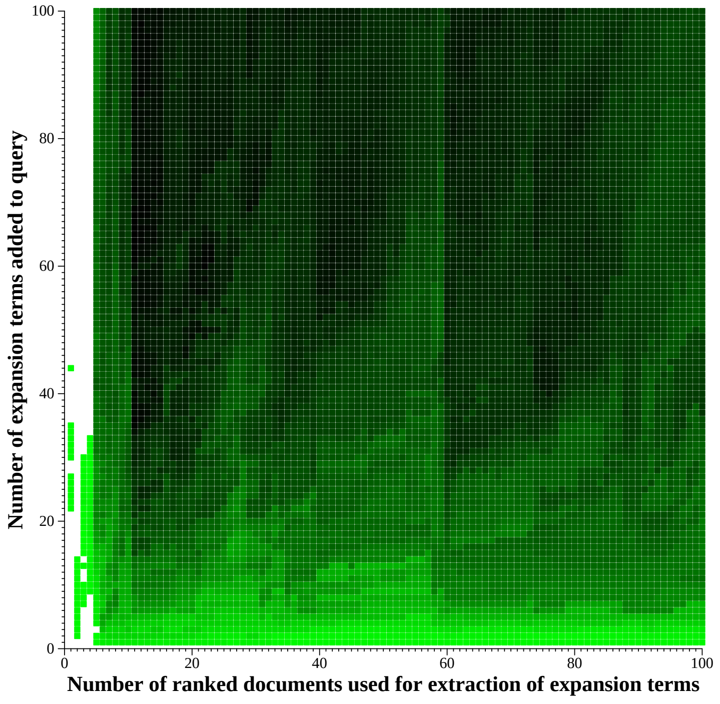
### 4.4.3 Optimal parameters for individual queries

Interestingly, neither the default ($|R'| = 10$,$|E| = 25$) nor the optimal parameter settings (13,15) are particularly effective for either of these two queries. It can also be seen that parameter pairs that work well for one query don't work anywhere near as well for another. Query 405 is best expanded with the combination of (11,65), with large increase in average precision from the baseline of 0.061 to 0.317, while for query 440 the best parameter pair is (61,4), which results in a moderate increase from 0.093 to 0.124. Using the optimal pairs on the respective other query does not work well, as can be seen from Figure 4.3; using the pair (61,4) on query 405 gives average precision of 0.119, worse by almost a factor of three; using the parameters (11,65) for query 440 gives 0.015, which is worse by a factor of eight.

More generally, the best expansion parameters vary wildly between queries, as illustrated in Figure 4.4, which on the top shows the optimal parameter pair for each of the 50 queries. If expansion was applied with the optimal parameters for each query, mean average precision would increase from 0.260 to 0.330, where 0.260 is the best result observed when the same parameters are used for all queries.

Whereas this graph shows that no two queries have the same optimal parameter pairs, the graph on the bottom of Figure 4.4 shows that some queries can achieve an average precision that is quite close to optimality, when using the best parameter pairs of a specific different query. In particular, this graph shows the 100 coordinates with the highest average precision with descending point sizes for each of the 50 queries. It can be seen that some of the marks are covering each other. Most top coordinates for each query are clustered around the parameter pair that achieves best mean average precision when using a single parameter pair for all queries (this can also be seen on the left hand side). Coordinates associated with an individual query are clustered in one of three possible ways:

- For some queries the coordinates are clustered along the x-axis, building a "horizontal band", which means that a particularly important term is found (often in a specific document and therefore the corresponding streak starts at a certain x position).



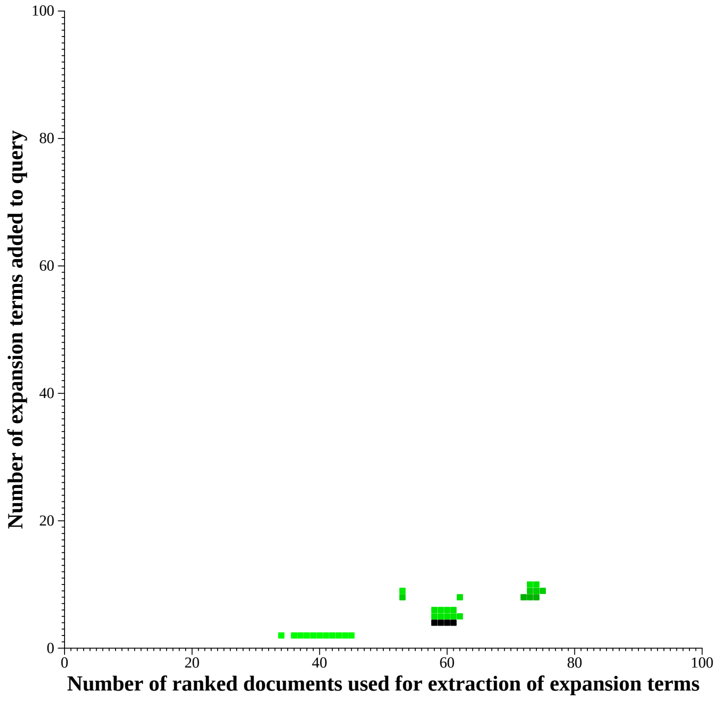
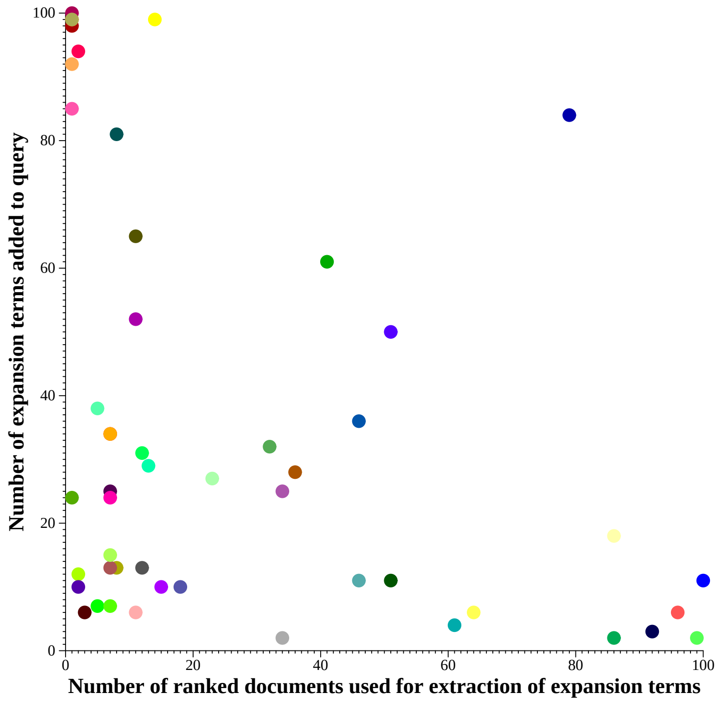


*Figure 4.3: Parameter pairs for which expansion of TREC 8 query 405 (on the top) and 440 (on the bottom) achieves higher average precision than non-expansion. Dark spots mark high average precision values, as before.*



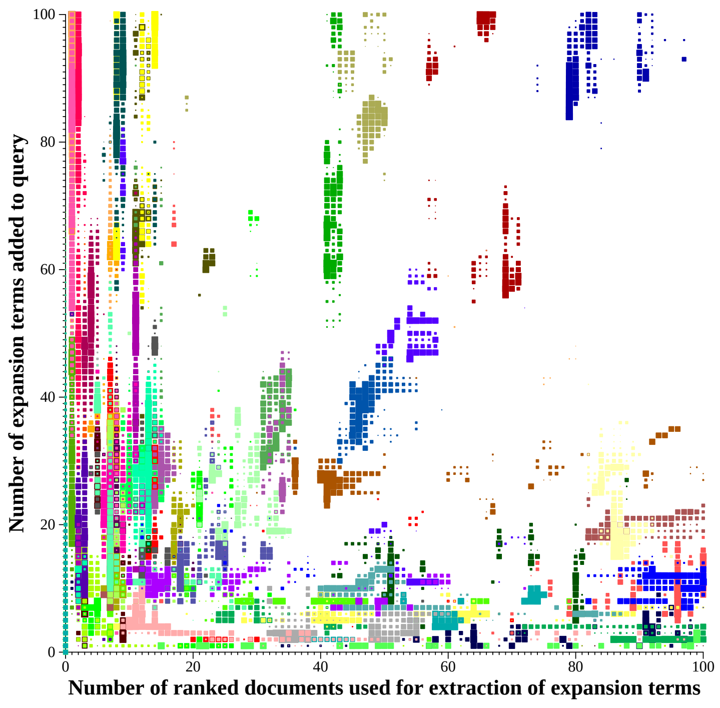


*Figure 4.4: The top figure shows parameter pairs for each query where expansion achieves maximum average precision. The bottom figure shows the best 100 parameter pairs in descending point size for each query.*

- Conversely to the first case, for other queries the cluster is stretched along the y-axis and is therefore organised in a "vertical band". This means a document that has particularly good expansion terms was added to the local set.
- The last possibility is a combination of both, where coordinates are more closely centred on a particular coordinate (or rather the cluster stretches out along both axis from a certain coordinate onward), which means, that as long as a certain document has entered the local set and a particular term is added to the list of expansion terms, the query performs well.

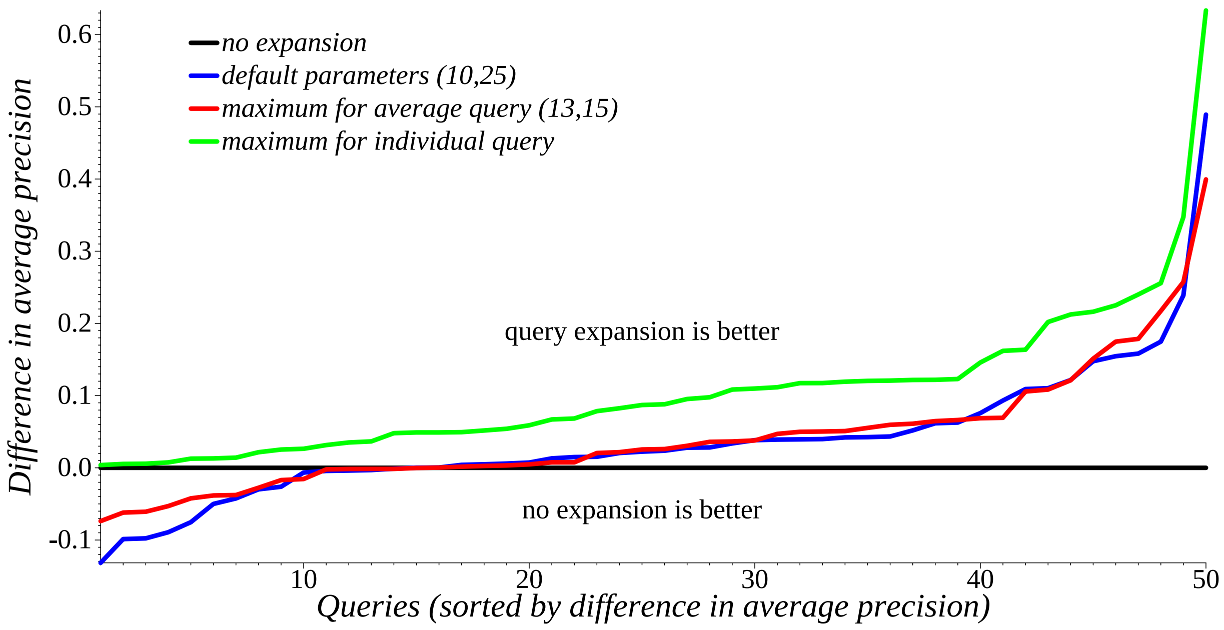
The coordinates of a query where average precision has greatly changed (either improved or decreased) are likely to be arranged in one of the first two ways. Interestingly, for all queries the top 100 pairs were clustered together in some way.

### 4.4.4 Lack of robustness

Per-query improvements in average precision due to query expansion are shown in Figure 4.5. The baseline, shown as a line at 0.0, is the performance without expansion achieved by Lucy. Also shown are the change in effectiveness from Lucy using the standard Robertson-Walker parameters, and the change using the best parameters for that collection and query set. Finally, the performance with individually tuned optimal parameters for each query is shown.

For the TREC 8 data, the standard parameters and best parameters are little different ($|R'| = 10$ and $|E| = 25$ versus 13 and 15, respectively). However, choosing optimal parameters per query gives much greater effectiveness. For TREC 9, the standard parameters are very poor, with around half of the queries degraded and less than a third improved. The best parameters of 98 documents and 4 expansion terms give much better performance; a small number of queries are degraded, but only slightly. With choice of the best parameters per query, all queries improve to some degree, or are not degraded, at a minimum.

In Figure 4.6 the top graph of Figure 4.5 is repeated, but here queries are sorted by their ID (again shown from 1 to 50 for simplicity) rather than by their change in effectiveness from the baseline. Using this view, it can be seen that the result achieved when using the best fixed parameters pair for the query set overall is in most cases fairly close to that achieved by using the default parameter, except for queries at positions 30 and 35, where the former parameter set significantly improves or degrades performance, respectively. For these two queries, this change is so pronounced that where a query performs worse through expansion with the default parameter set, it is improved by using the best average parameter pair, and vice versa. Overall, however, the best average parameter set results in a fairly robust improvement of expansion – only a small minority of queries performs worse. It is also interesting to note that the best individual parameter settings can sometimes achieve large improvements, even for some queries that do not improve through expansion with any of the other parameter settings, such as for queries at position 9 and 41.



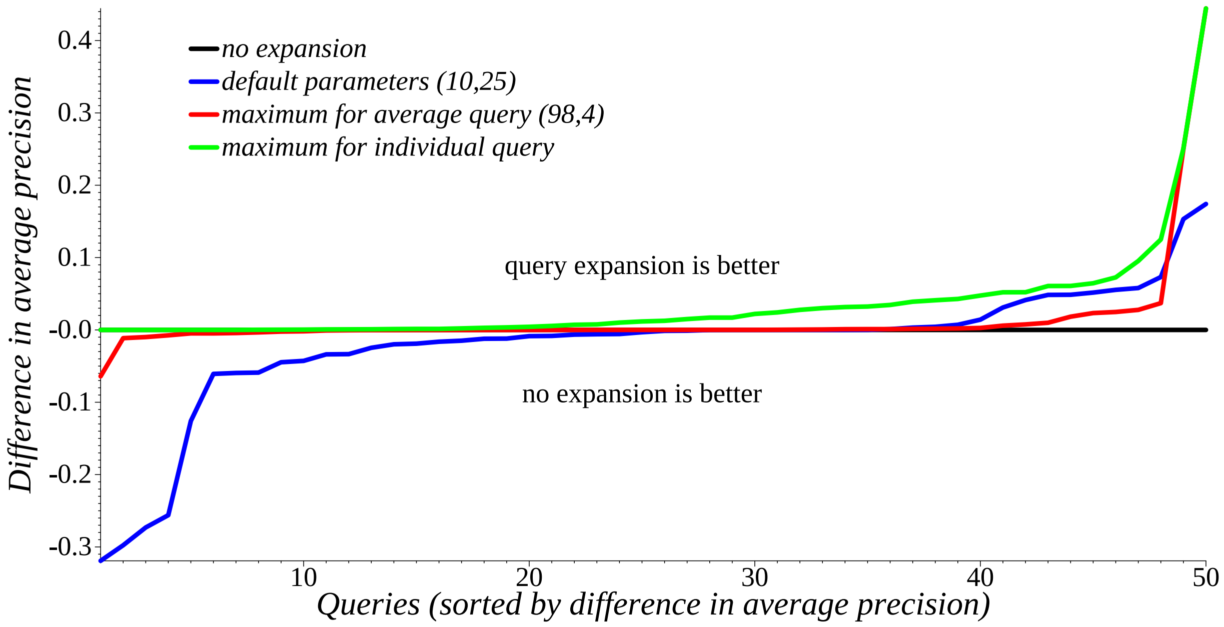


*Figure 4.5: Average precision at 1,000 documents ranked with different parameter settings shown. Each curve is sorted individually by the magnitude of the difference in average precision. The line that shows the results of individually optimal parameter setting provides an upper bound for the effectiveness of local analysis query expansion. TREC 8 results are shown on the top and TREC 9 are displayed on the bottom.*

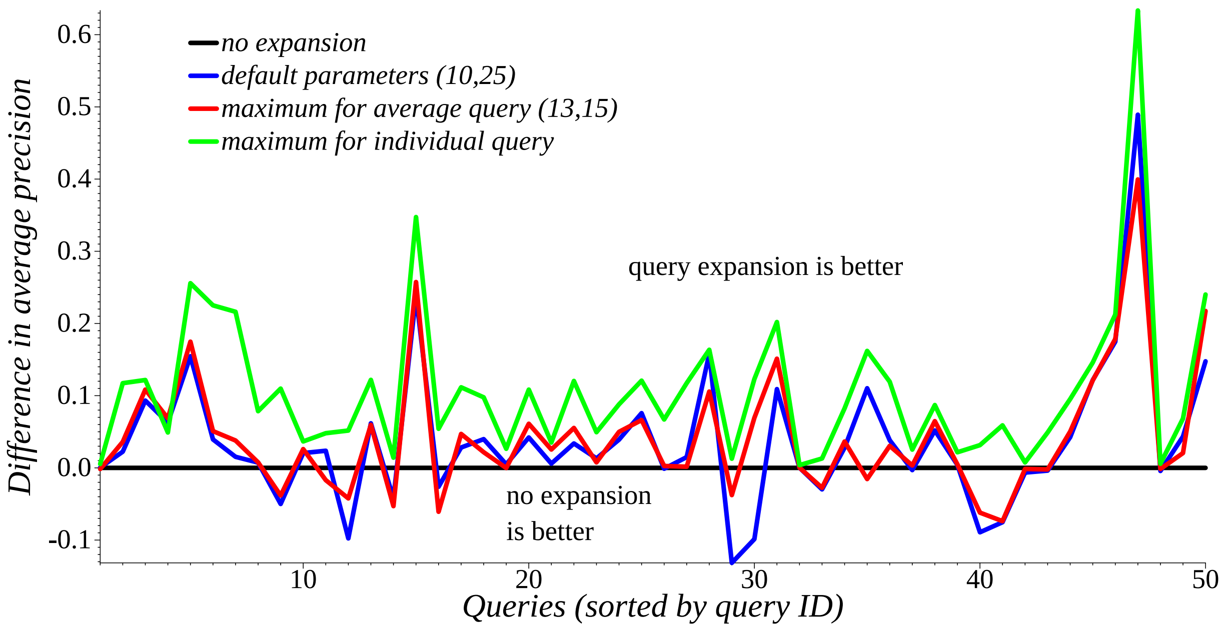


*Figure 4.6: The same (TREC 8) information is conveyed as in the top graph of Figure 4.5, except that here the data producing the individual lines are not sorted by average precision, but by query ID. This view makes it possible to see how the different parameter settings affect each individual query.*

## 4.5 Attempts at making expansion more robust

Intuition suggests that queries that are effective prior to expansion should be good candidates for query expansion, since many relevant documents with well-suited expansion terms are used as sources in the query expansion process. This intuition is strengthened by the observation that query expansion that is based only on relevant documents in the top $|R'|$ is superior to query expansion based on all documents, as we have seen in our experiments and as reported for example by Mano and Ogawa (2001). However, we found that there is no relationship between the average precision that the original query achieves and by how much query expansion improves average precision. Using the Pearson product moment correlation (Gravetter and Wallnau, 1991, pages 471–481), no correlation was found between the improvement of average precision through query expansion and the average precision of the original query. This is illustrated in Figure 4.7, for which the Pearson product moment correlation rejects the null hypothesis of correlation between the two axes of the graph. Note that for generating this particular graph we used the optimal parameter pairs for each individual query, which is why all changes in average precision are positive. In other unreported experiments, we did not observe a correlation between the default parameter pair and the improvements in average precision for each query, either.

There is also an argument to be made for the converse: queries that did not perform well to begin with have greater scope for improvement through expansion. This also is not supported by the Pearson product moment correlation.

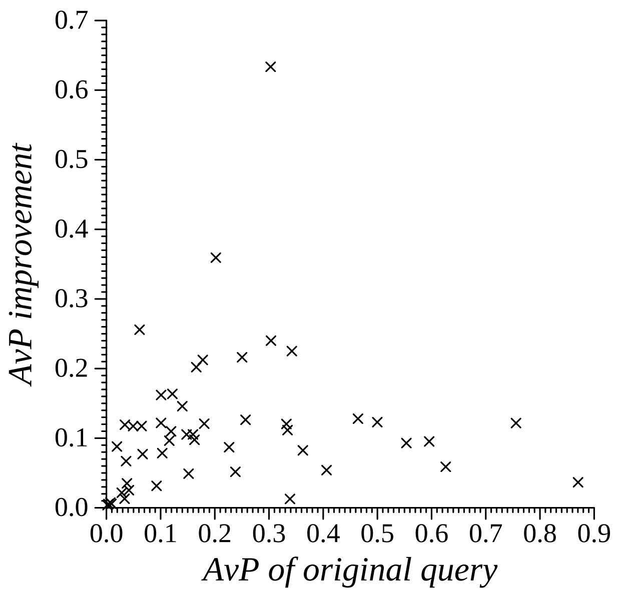


*Figure 4.7: Improvement through query expansion versus average precision of original query. No correlation is observable between those two quantities.*

An interesting question is then whether some property of the original query can be used to predict whether expansion will be effective. We explored a range of query metrics, but without clear success. These included the similarity score of the documents fetched in the original ranking; a measure of how distinct these documents were from the rest of the collection; specificity of the query terms; and an approximation to *query clarity* (Cronen-Townsend et al., 2002), where the language of a query is compared to that of the background collection, from which it is deduced whether the query will find good matches in the collection (see also Section 2.4.3). None of these were effective. However, since this research was conducted, Amati et al. (2004) found – albeit using a single collection for their experiments – that there is a relationship between the change of effectiveness and query clarity. They measure the clarity score of the initial and that of the expanded query, and then issue either the initial or the expanded query, depending on the difference in scores. Cronen-Townsend et al. (2004*a*) also applied a variation of query clarity to selectively expand some queries with some success (see Cronen-Townsend et al. (2004*b*) for the full paper).

Classification of queries – such as that by Broder (2002) into navigational, informational, and transactional – and using the hypothesis that only informational queries are expandable, might lead to improvements. Employing an automatic query classification scheme, such as that proposed by Kang and Kim (2003), makes selective expansion feasible and warrants further investigation.

Sakai and Robertson (2001) have suggested varying the parameters per query by classifying queries into one of 10 bins, according to measures such as the similarity score of the highly-ranked documents. As we did not observe any correlation between such scores and improvements due to expansion, we are not convinced that such a strategy is likely to be successful.

Carpineto et al. (2001) experimented with varying $|R'|$ for some fixed $|E|$, and with varying $|E|$ for some fixed $|R'|$, considering the impact on average effectiveness for two data sets. They conclude that some limit on the number of expansion terms is warranted, but did not observe that the various settings had different impact on different queries. Our work generalises their results.

## 4.6 Reliable information access workshop

Our findings above – which date from early 2003 – were later validated through the *reliable information access workshop* held in late 2003, where the National Institute of Standards and Technology of the United States (NIST) brought together a number of different research organisations and ran a set of experiments on each of seven systems (Harman and Buckley, 2004). The systems used ranged from traditional vector space models (such as the SMART system (Salton, 1971), see also Section 2.4.1) to those based on language models (see Section 2.4.3).

The main findings were as follows:

- **Sensitivity to $\mathbf{R'}$:** Systems are very sensitive to the initial set of documents in the local analysis process, with results varying between 10% and 50% for different systems, when running each system with the initial set from the other systems.
  - Montgomery et al. (2004) show how retrieval effectiveness varies with the size of the local set of documents, which follows a different curve for each system, but generally the best results are achieved at around 10 to 15 documents and then decreases as more documents are added. They also find that there is no correlation between either the query length or the number of relevant documents and the optimal number of of feedback documents.
  - Warren and Liu (2004) find that all systems achieved good improvements by using about 6 documents in the initial set, and only marginally improving retrieval by adding any more documents.
  - The mismatch of initial retrieval stages and expansion phases of different system is examined by Lynam et al. (2004). They demonstrate that systems can perform better when using a set of initial documents ranked by a different system, even if that system would not have achieved improvements using a local set of documents ranked by itself. Furthermore, the combination of multiple local sets of documents can lead to improvements. Lynam et al. report a 13% improvement over the best individual system when fusing the initial sets of all systems.
- **Variable $|\mathbf{E}|$:** Choosing the best number of expansion terms can increase results by 30% over a fixed number of terms added.
- **Choice of E:** Finally, even with the same set of initial documents, only 15% to 25% of expansion terms were the same when selected by different systems.

The original report of this workshop was written by Buckley and Harman (2004). The large volume of data resulting from this workshop has since been analysed in detail by various research groups (some of which has been shown above).

Buckley (2004) groups topics into difficulty classes. Sorting the cause of the failure in terms of their natural language understanding (Allen, 1995) needed to overcome the various problems, some of the categories were: general technical failure (such as stemming); one aspect of topic was emphasised, while another was missed; emphasising a non-relevant aspect of the topic, while missing the point of the topic; not performing question answering type analysis on the query to find what terms of the topic are of interest; or not expanding a topic correctly that would need human intervention, such as "new methods" in the topic "what are new methods of producing steel".

Gu and Luo (2004) compared blind relevance feedback from documents to that of passages and found that retrieval becomes generally more stable when passages are retrieved for the local set. Interestingly, the improvement offered by passage retrieval over that achieved by document retrieval seems to be dependent on the length of relevant documents. In particular, passage retrieval performs best on short documents with less than 500 terms and on long documents that have more than 2000 terms.

## 4.7 Summary

Query expansion is a successful method for improving the average effectiveness of an information retrieval system, particularly for cases where there is a vocabulary mismatch between query and relevant document. Expansion is most successful when the documents that match the original query include topic-specific terms that can be automatically identified and then used to fetch further documents. For some queries, however, automatic expansion can introduce non-relevant terms that degrade effectiveness.

Furthermore, the precision of a certain proportion of queries is greatly increased or decreased, to the point where differences approaching 50% in absolute terms are observed. Surprisingly, the success or failure is often determined by a single expansion term, while most expansion terms have almost no effect on the query at all.

We have quantified the performance of a successful query expansion technique, by exploring behaviour as parameters are varied. This exploration has identified an upper bound on the improvement available via the Okapi approach to query expansion on two test collections, and showed that the use of fixed parameters for all queries can be significantly improved upon.

We have identified that query expansion is much less reliable than previously suggested in the relevant literature. Despite the positive results reported in many previous papers, in our experiments query expansion failed on many queries and behaviour is highly inconsistent from collection to collection.

What is not clear is how the parameters should be chosen. We have explored a range of options, but have not identified a metric that provides a method for guiding expansion. We are exploring how to use the results in this chapter to develop new techniques for robust query expansion, as well as techniques for predicting whether expansion will be of value. Nonetheless, with appropriate parameter choices query expansion is a successful way of enhancing the effectiveness of queries, particularly on collections with typically consistent documents.

# Chapter 5

# Query Expansion using Associated Queries

In web retrieval, a user's information need is typically expressed as a query consisting of a small number of terms (Spink et al., 2001), and answer documents are chosen based on the statistical similarity of the query to the individual documents in the collection. Much research over several decades has led to development of statistical similarity measures that are reasonably effective at finding answers for even the shortest queries (see Chapter 2).

In the previous chapter we found not only that the standard parameters for local analysis query expansion are inappropriate for web data, but that, even with the best parameters (found by tuning to that data and query set), the performance gains are small. However, we also found that the addition of good expansion terms to the original query in a more controlled environment – such as newswire data – can lead to significant improvements in effectiveness. It is a reasonable proposition therefore, if we could get a more controlled vocabulary view of web data, we should be able to achieve good improvements here, too.

In this chapter we propose an alternative approach to query expansion. The general approach we consider is that the source of expansion terms need not be the collection itself, but could be any set of documents whose topic coverage is similar to that of the collection, and may thus suggest additional query terms. We explore a novel way of expanding queries, where – instead of using the top ranked documents as sources for expansion terms – we select terms from past user queries that are associated with documents in the collection. Since user queries are typically carefully constructed to retrieve information, the vocabulary employed is therefore more controlled than that of web pages, which often consist of unrelated terms that for instance co-occur in tables (see also Williams and Zobel, 2005).

Specifically, we investigate whether *query association* can be used for query expansion (Scholer, 2004). Given a query log containing a large number of queries, it is straightforward to build a surrogate for each document in a collection, consisting of the queries that were a close match to that document. Scholer and Williams (2002) have shown in earlier work that query associations can provide a useful document summary; that is, the queries that match a document are a fair description of its content. Here, we investigate whether query associations can play a role in query expansion.

| | |
|---|---|
| earthquakes seismicity | weekly seismic reports for northern california |
| richter | seismograph |
| seismograph | nevada earthquake seismological seismicity usgs |
| earthquakes neic | volcanic activity recent |
| richter scale | maps of recent major earthquakes |
| kobe earthquake | magnitude and body and wave |
| san andreas fault | past earthquakes earthquake magnitude |
| predicting earthquakes | university of washington earthquake center |
| tectonic plates | earthquake epicenter |

*Figure 5.1: Sample query log queries on the topic "earthquakes". Queries shown make a good pool of candidate terms on a query on that topic.*

Consider the query "earthquakes", where a user wants to find out more about this topic. When using documents in a web collection as a source for expansion terms, local analysis is designed to find terms that are related to this topic from the top ranked documents. If other minor topics are present in some of the top ranked documents, it is plausible that terms not related to earthquakes would be used for expansion. However, consider the rich source of highly relevant terms in the past queries shown in Figure 5.1. These queries are very concise and all focus on the topic at hand. A difficulty of finding relevant queries from a log is that many of the queries that would be a useful addition to the pool of queries from which expansion terms are drawn cannot be found in the first place, if they do not contain the original query term (only about half the queries in Figure 5.1 contain the term "earthquakes"). Query associations provide a way to make use of topical queries. With our proposed method we are targeting these pools of queries as a source of expansion terms. Although these queries could also be on minor related topics, this is unlikely, since they have been associated with the main topic of the document. As we will explain later, they otherwise would not have been associated with the document, since the document would not have been ranked highly against this query.

As discussed in Chapter 3, ranking can be broken up into three phases: first, the original query is ranked against the collection; second, additional query terms are extracted from highly ranked documents; and third, the new query is ranked against the collection. We show that query associations are a highly effective source of expansion terms. Our scheme is effective for query expansion for web retrieval: our results show relative improvements over unexpanded full text retrieval of 26%–29%, and 18%–20% over an optimised, conventional expansion approach.

A preliminary version of this chapter appeared as Billerbeck et al. (2003).

## 5.1 Background

This chapter explores refinements to automatic query expansion by considering alternative ways in which candidate expansion terms can be chosen. In this section, we consider related background work in the areas of query expansion and the use of past user queries.

### 5.1.1 Past queries

Past queries have been shown to be useful for increasing retrieval effectiveness (Fitzpatrick and Dent, 1997, Furnas, 1985, Raghavan and Sever, 1995). Fitzpatrick and Dent investigated the use of past queries to improve automatic query expansion, by using the results of those queries to form *affinity pools*, from which expansion terms are then selected. The process works as follows: for a query that is to be expanded, up to three past queries that are highly similar to the current query are identified. The top 100 documents that were returned for each of these past queries are then merged, forming the affinity pool. Candidate expansion terms are identified by running the original query against this pool. Next, individual terms are selected from the top-ranked documents using a *tf.idf* term-scoring algorithm. Fitzpatrick and Dent experiment with the TREC-5 ad-hoc collection, using past queries from the ad-hoc tracks from TRECs 3 and 4, and demonstrate that their technique results in a 15% relative improvement in average precision over a non-expanded baseline run.

In our work, we propose a different approach the use of past queries. We also choose expansion terms from past queries directly, rather than using them to construct sets of full text documents from which terms are then selected.

### 5.1.2 Query expansion based on past user queries

A detailed description of query expansion is given in Section 3.3.2, and is not repeated here. In the context of this chapter, however, it is noteworthy that Komarjaya et al. (2004) make use of past queries as well as local analysis by linking document headings to query terms, especially since there is not much previous work on using past queries for query expansion, with the exception of the work described in the following section. In the context of online public access catalogs, Komarjaya et al. choose subject headings depending on how many query terms of the past queries they are associated with and the cohesion of associated titles. After a thesaurus of past queries and subject headings is built, queries are then processed and expanded based on the degree of co-occurrence between the query and thesaurus entries:

$$g(h_d, q) = \prod_{t \in q} (\delta + \textit{co-degree})$$

Similar to the work by Qiu and Frei (1993), which we describe in Section 3.3.3, the *co-degree* is calculated as follows:

$$\textit{co-degree}(h_d, t) = f(h_d) \times \ln \frac{m}{f_{h,t}}$$

where $\delta$ is a smoothing factor used to avoid the zero frequency problem described in Section 2.4.3 and is set to $0.001$ in the work at hand; $m$ is the number of distinct query terms in query $q$; $f(h_d)$ is the frequency with which the query term $t$ co-occurs with the subject heading $h_d$; and $f_{h,t}$ is the number of distinct terms that co-occur with term $t$ in the thesaurus.

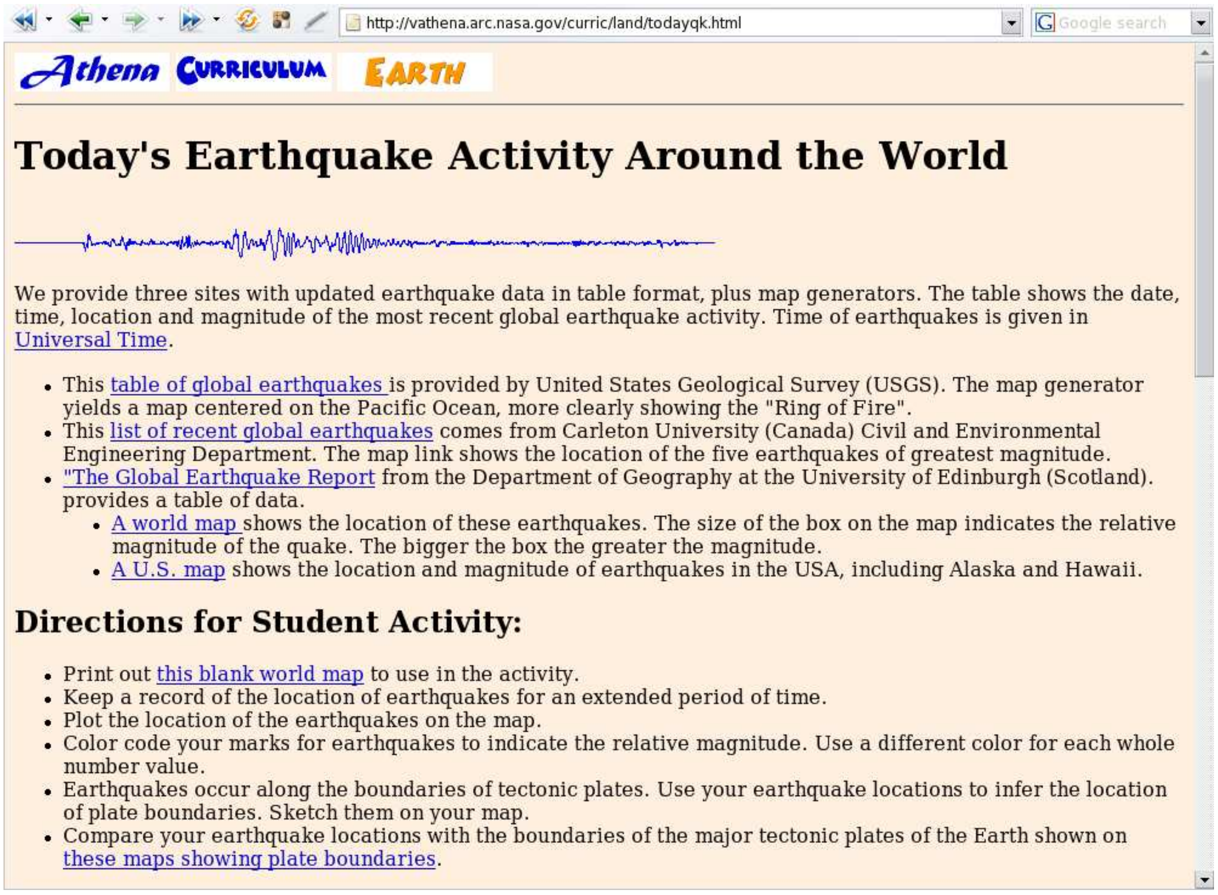
Komarjaya et al. argue that using a thesaurus made up of past queries is a good way of expanding queries, because subject headings are less ambiguous and are edited by experts of the subject matter. Their work demonstrates a relative improvement in average precision by 31%, in a relatively small-scale experiment. Our work differs from the method described here in that we do not make use of a thesaurus, but rather use past queries more directly, as explained later. We also rely only on past queries, rather than on subject headings or any other data, which are only indirectly connected to the past queries.

### 5.1.3 Query association

*Query association* is a technique whereby user queries become associated with a document if they share a high statistical similarity with the document. This technique was proposed for the creation of document summaries, to aid users in judging the relevance of answers returned by a search system. It has been successfully used to increase the weighting of terms that encapsulate the "aboutness" of a document (Scholer and Williams, 2002, Scholer, 2004).

The association process works as follows: a query is 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 (for example, using the Okapi BM25 ranking function described in Section 2.4.2). The query then becomes associated with the top $A$ documents that are returned. For efficiency, an upper bound $B$ is imposed on the number of queries that can become associated with a single document. Once a document has a full set of $B$ associations, the least similar associated query can be dynamically replaced with a new, more similar query.

Consider the following brief example, where we start with no stored associations, and use association parameter settings of $A = 5$ and $B = 2$. A user runs an initial query $q_1$. This query becomes associated with the top five answer documents returned by the search system. Suppose that a second query $q_2$, retrieves a further five documents, one of which was also retrieved by the first query. Then this document now has two associations, while eight other documents in the collection have one. A final query $q_3$ also retrieves the document that already has two associations as one of its answers. If the similarity scores between queries and the common document are ordered such that $q_1 < q_2 < q_3$, then $q_1$ will be replaced with query $q_3$ as an association for that document.


http://vathena.arc.nasa.gov/curric/land/todayqk.html

Athena CURRICULUM EARTH

Today's Earthquake Activity Around the World

We provide three sites with updated earthquake data in table format, plus map generators. The table shows the date, time, location and magnitude of the most recent global earthquake activity. Time of earthquakes is given in Universal Time.

- This table of global earthquakes is provided by United States Geological Survey (USGS). The map generator yields a map centered on the Pacific Ocean, more clearly showing the "Ring of Fire".
- This list of recent global earthquakes comes from Carleton University (Canada) Civil and Environmental Engineering Department. The map link shows the location of the five earthquakes of greatest magnitude.
- "The Global Earthquake Report from the Department of Geography at the University of Edinburgh (Scotland). provides a table of data.
  - A world map shows the location of these earthquakes. The size of the box on the map indicates the relative magnitude of the quake. The bigger the box the greater the magnitude.
  - A U.S. map shows the location and magnitude of earthquakes in the USA, including Alaska and Hawaii.

Directions for Student Activity:

- Print out this blank world map to use in the activity.
- Keep a record of the location of earthquakes for an extended period of time.
- Plot the location of the earthquakes on the map.
- Color code your marks for earthquakes to indicate the relative magnitude. Use a different color for each whole number value.
- Earthquakes occur along the boundaries of tectonic plates. Use your earthquake locations to infer the location of plate boundaries. Sketch them on your map.
- Compare your earthquake locations with the boundaries of the major tectonic plates of the Earth shown on these maps showing plate boundaries.

*Figure 5.2: Example web page for which associated queries are shown in Figure 5.3.*

Scholer et al. (2004) report that appropriate parameter settings for this task are $B = 5$ and $A = 3$, leading to small summaries composed of high-quality associations. Keeping the summaries small was important for reducing cognitive processing costs for the user. In this chapter, we use associated queries as a source of terms for query expansion. It is therefore not imperative that the number of associated queries be kept low. We discuss the choice of parameters further in the Section 5.3.1.

A web page that was found to be relevant to the query "earthquakes" by relevance assessors in the 2001 TREC web track using ad-hoc queries is partially shown in Figure 5.2. Figure 5.3 contains the queries that were associated by our system with that web page.

## 5.2 Generalised expansion

In this section, we describe our approach to query expansion and, in particular, focus on the novel use of query associations in the expansion process. Our generalised method for query expansion proceeds as follows: first, a query is submitted to our search system, and the top $|R'|$ answer documents are obtained,

```
tectonic plates               earthquakes in canada earth
earthquakes earthquake        recent earthquake in california
earthquake on hawaii          map of usa virtual locations maps
earthquakes seismograph       united states map showing states
earthquake epicenter          university of washington earthquake center
united states earthquakes     earthquake earthquakes
earthquakes nevada            earthquakes in canada earthquake
recent earthquakes            maps of recent major earthquakes
earthquake alaska             earthquakes in california
california earthquakes
```

*Figure 5.3: Associated queries for document shown in Figure 5.2.*

based on a particular collection of documents or surrogates thereof (notation used in this chapter – in line with that used in previous chapters – is summarised in Table 5.1). From this initial retrieval run, it is possible to identify a set of candidate expansion terms; these may be based on the top $|R'|$ documents, or surrogates corresponding to these documents. Then the top $|E|$ expansion terms are selected, using Robertson and Walker's term selection value formula (see Section 3.3.2). Finally, selected terms are appended to the original query, which is then run against the target text collection.

Within this general framework, if we use a single collection of documents for all steps, then query expansion is of the standard form, as for example proposed by Robertson and Walker (1999, 2000). We call this scheme FULL-FULL, as steps one and two of the expansion process are based on the full text of documents in the collection.

Instead of initially searching or choosing expansion terms from the full text of a document, another possibility is to use surrogate documents constructed from query associations. In this approach, queries are associated with documents as described earlier, then the set of associated queries for a document is used to represent the document. These can be incorporated into the expansion framework in either the first step (ranking), the second step (term selection), or both. In detail, these three options and the conventional approach are as follows (see also Figure 5.4 for a graphical representation of the different approaches to expansion).

- FULL-FULL
  The original query is ranked on the full text collection, after which the top $|E|$ expansion terms are selected from the top ranked $|R'|$ documents returned from running the original query. This option is labelled FULL-FULL and gives a reference point to compare to for the following expansion runs of our experiments.

| | |
|---|---|
| $R'$, $\|R'\|$ | The set and number of documents assumed relevant that are used for local analysis |
| $E$, $\|E\|$ | The set and number of expansion terms added to a query |
| $A$ | The number of documents returned for each query during the association process |
| $B$ | The number of queries maximally associated with a document |

*Table 5.1: Notation used for query expansion based on associated queries.*

- FULL-ASSOC
  The original query ranks documents of the full text collection, and expansion terms are selected from the set of queries that have previously become associated with the top documents returned from running the original query. We call this scheme FULL-ASSOC, as step one of expansion is based on the full text of documents in the collection, while step two is based on query associations.

- ASSOC-FULL
  In this scheme, surrogates that have been built from associations are used for the initial ranking, while expansion terms are chosen from the original documents. We call this scheme ASSOC-FULL.

- ASSOC-ASSOC
  The document surrogates are used for both the initial ranking and for the term extraction, where the $|E|$ expansion terms are selected from the top $|R'|$ ranked surrogates. We call this scheme ASSOC-ASSOC, as associations are used for both steps 1 and 2 of expansion.

An alternative way to find candidate terms for query expansion from past user queries is to treat the individual queries as documents. We can then source expansion terms by initially ranking the individual queries, and selecting $|E|$ terms from the top $|R'|$ past queries returned. We call this scheme QUERY-QUERY. Note that, as the individual queries have no direct relation with any particular full text document in the collection (in contrast to the association case above), it does not make sense to have a FULL-QUERY or QUERY-FULL scheme.

The use of past queries for expansion is attractive for several reasons. One is that it means that the additional terms have already been chosen by users as descriptors of topics. This is particularly important in the context of web retrieval, where the vocabulary is less controlled than in a collection of newswire data for example; the associated queries are arguably of higher quality than the average web document, in terms of the selection of vocabulary. Another reason, particularly relevant for associations (as opposed to single queries as described in the QUERY-QUERY scheme), is that there is more evidence of relevance: a surrogate document constructed from associations has many more terms than an individual query. The fact that the queries have become associated with the document means that, in some sense, the terms have topical relationships with each other.

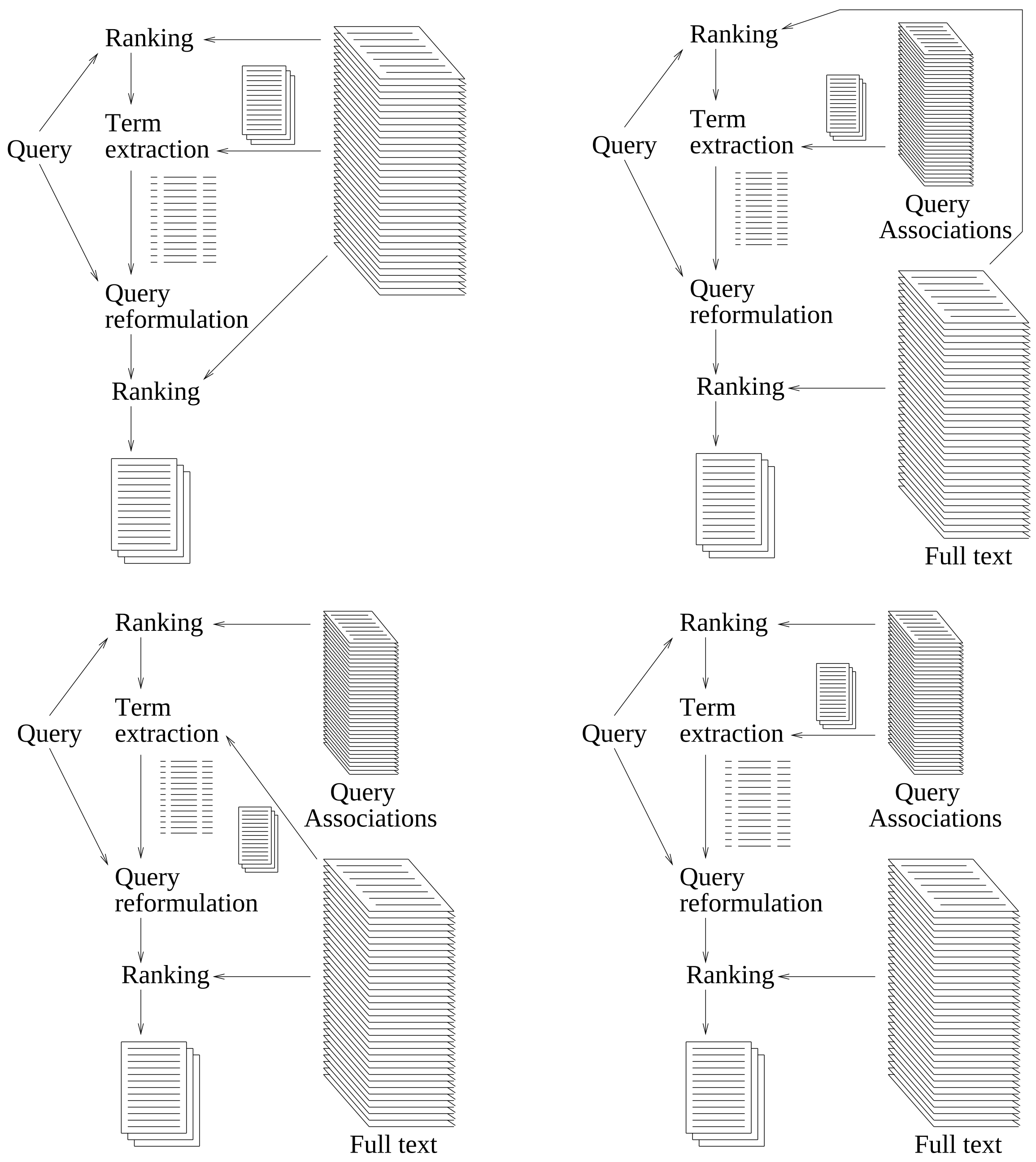


*Figure 5.4: Using query associations for query expansion.* Top left*: the* FULL-FULL *scheme is shown, which is the conventional approach to query expansion and provides us with a baseline for our experiments.* Top right*: the* FULL-ASSOC *scheme ranks documents initially from the full text collection, but sources expansion terms from associated queries.* Bottom left*: the* ASSOC-FULL *scheme ranks associations initially and extracts candidate terms from the full documents.* Bottom right*: associations are used for both, the initial ranking and the term extraction in the* ASSOC-ASSOC *scheme.*

Unlike other methods that make use of the top-ranked documents, such as expansion from document summarisations by Lam-Adesina and Jones (2001), ASSOC-ASSOC does not rely directly on the document collection that is searched. The second and third expansion variations (schemes ASSOC-FULL and ASSOC-ASSOC) do not rely on ranking the documents in the collection to find relevant associations, but treat the associations themselves as documents that are ranked and used as sources for expansion terms. Thus, in contrast to using thesauri (see Section 3.3.1), the "aboutness" of the individual documents is captured and made use of.

Another source of terms for query expansion that we have experimented with is anchor text (see also Section 2.5.1). Inlinks (text from anchor tags in other documents that point to a document) have a direct relationship with documents in the collection. We consider one approach using anchor text, where we select $|E|$ expansion terms from the top $|R'|$ anchor text surrogates, and then search the surrogates again using the expanded query. We call this LINK-LINK.

Most of the schemes described above have parameters that need to be determined, in particular $|R'|$ and $|E|$. Rather than make arbitrary choices of values, in most cases we used the TREC-9 queries and relevance judgements described below to find good parameter settings, using the average precision measure, and then report only these settings on TREC-10. Thus the TREC-9 results are the best possible for that method on that data, with post hoc tuning, while the TREC-10 results are blind runs.

## 5.3 Experiments

In this section, we describe our experimental environment, discuss the methods used to validate our results, and present the results of our experiments with query expansion techniques for web collections.

### 5.3.1 Setup

For our experiments, we use the same setup as detailed in Section 4.1. We used 50 queries for our training and test sets from TRECs 9 and 10 respectively (see Section 2.6.2). We made this choice of training and test set because the two query sets are quite different in terms of their amenability to expansion (as explored in Chapter 4). In Chapter 6, we reversed the order of training and test sets intentionally, in order to avoid any possible collection-dependent effect on our experiments.

The approach used for creating associations is that described in Section 5.1.1. The query associations were built using two logs from the Excite search engine, each taken from a single day in 1997 and 1999 (Spink et al. (2002) provide a comprehensive analysis of the properties of these query logs). After filtering the logs to eliminate profanities, and removing duplicates and punctuation, we were left with 917,455 queries to associate with the collection. The average number of associations per document after processing was 5.4, and just under 25% of documents in the collection had zero associations. While we built our associations as a batch job, in a production system associations would be made in real time as

| Type | MAP | P@10 | P@20 | P@30 | R-Pr. | $\lvert R'\rvert$ | $\lvert E\rvert$ |
|---|---|---|---|---|---|---|---|
| Base | 0.1487 | 0.2714 | 0.2235 | 0.2000 | 0.1710 | — | — |
| ASSOC-ASSOC | 0.1893‡ | 0.3429‡ | 0.2888‡ | 0.2503‡ | 0.2204‡ | 06 | 17 |
| ASSOC-FULL(I) | 0.1820‡ | 0.3184† | 0.2796‡ | 0.2497‡ | 0.2222‡ | 06 | 17 |
| ASSOC-FULL(II) | 0.1618‡ | 0.3041† | 0.2510‡ | 0.2231‡ | 0.1969‡ | 98 | 04 |
| FULL-FULL(I) | 0.1584 | 0.2796 | 0.2571‡ | 0.2333‡ | 0.1809 | 10 | 25 |
| QUERY-QUERY | 0.1567 | 0.2755 | 0.2357‡ | 0.2116† | 0.1861‡ | 65 | 02 |
| FULL-FULL(II) | 0.1553 | 0.2857 | 0.2388 | 0.2184† | 0.1867† | 98 | 04 |
| FULL-ASSOC | 0.1549 | 0.2571 | 0.2276 | 0.2068 | 0.1786 | 06 | 17 |
| LINK-LINK | 0.1454† | 0.2653 | 0.2153 | 0.1905 | 0.1685 | 37 | 02 |

*Table 5.2: Results for query expansion by using query associations. Performance of expansion techniques of TREC-10 queries on the TREC WT10g collection, based on mean average precision (MAP), precision at 10 (P@10), precision at 20 (P@20), precision at 30 (P@30), and R-Precision (R-Pr.). Schemes are ordered by decreasing MAP. Results that show a significant difference from the baseline using the Wilcoxon signed rank test at the 0.05 and 0.10 levels are indicated by ‡ and † respectively.*

each query is submitted to the system, or after a number of queries have been accumulated. The costs of this process are explored by Scholer (2004, Chapter 6).

Effective settings for query association were established by Scholer et al. (2004). It was found that association parameters of $B = 19$ and $A = 39$ worked well, that is, each document has a maximum of 19 associated queries and the top 39 documents returned in response to a query are associated with that query. This was determined by creating document surrogates from the associated queries based on different parameter combinations, and testing retrieval effectiveness by evaluating searches on these surrogates. We use these parameter settings for our results reported in the following.

For our experiments where expansion terms are chosen directly from queries (with no association), we also use the 917,455 filtered entries from the Excite 1997 and 1999 Excite logs. The anchor text for our experiments was obtained by identifying anchor tags within the WT10g collection, and collating the text of each anchor that points in to a particular document into a surrogate for that document.

We evaluate the significance of our results using the Wilcoxon signed rank test as described in Section 2.6.3.

### 5.3.2 Results

In our experiments, we have compared a baseline full text retrieval run with the expansion variants we have described in Section 5.2. Our results are presented in Table 5.2, which shows retrieval performance

| Type | Q1 | Median | Q3 | Variance |
|---|---|---|---|---|
| ASSOC-ASSOC | -0.0239 | 0.0040 | 0.0814 | 0.7327 |
| ASSOC-FULL(I) | -0.0073 | 0.0116 | 0.0738 | 0.4286 |
| ASSOC-FULL(II) | -0.0119 | 0.0030 | 0.0308 | 0.1201 |
| FULL-FULL(I) | -0.0265 | 0.0007 | 0.0302 | 0.2633 |
| QUERY-QUERY | -0.0150 | -0.0000 | 0.0082 | 0.1108 |
| FULL-FULL(II) | -0.0211 | -0.0004 | 0.0177 | 0.1319 |
| FULL-ASSOC | -0.0370 | -0.0007 | 0.0306 | 0.2077 |
| LINK-LINK | -0.0049 | -0.0001 | 0.0002 | 0.0410 |

*Table 5.3: Quartiles and variance of different expansion methods on the TREC WT10g collection (TREC-10). Each number is the effectiveness relative to the baseline of no expansion.*

based on five precision metrics: precision at 10 returned documents (P@10), precision at 20 (P@20), precision at 30 (P@30), mean average precision (MAP), and R-precision (R-Pr.).

In these results, the FULL-FULL scheme is the conventional approach to query expansion. We show results with two parameter settings: first FULL-FULL(I), the parameter settings used by Robertson and Walker (1999, 2000) of $|R'| = 10$ and $|E| = 25$; and, second, FULL-FULL(II), the optimal parameters we found in exhaustive tests (we discuss this further below). Perhaps surprisingly – but in agreement with our work in Chapter 4 and recent observations by Carpineto et al. (2001) – the FULL-FULL schemes do not offer significantly better results than no expansion, except in the R-Precision measure for our optimal parameter settings (where the relative improvement is 9%, corresponding to an absolute increase of 0.016).

Our novel association-based schemes are effective for query expansion. In relative terms, the ASSOC-ASSOC scheme is 18%–20% better in all three measures than FULL-FULL(I) expansion (an absolute difference of 0.03–0.05), and 26%–29% better than the baseline no-expansion case (an absolute difference of 0.04–0.07). The ASSOC-FULL schemes are even more effective in the R-Precision measure; all results are significant at least at the 0.10 level. We conclude that query association is an effective tool in the initial querying stage prior to expansion; this is a particularly useful result since query associations are compact and can be efficiently searched.

Another perspective on the results is shown in Table 5.3. All of the methods improve median performance to approximately the same extent – that is, not at all. At the lower quartile, all the methods have degraded performance somewhat; the extent of degradation has little relationship to average effectiveness. At the upper quartile, however, the differences between the methods are clear. However, note that it is often the case that a query improved by one method is not improved by another.

| Type | MAP | P@10 | P@20 | P@30 | R-Pr. | $\lvert R'\rvert$ | $\lvert E\rvert$ |
|---|---|---|---|---|---|---|---|
| Base | 0.1895 | 0.2708 | 0.2042 | 0.1806 | 0.2290 | — | — |
| ASSOC-ASSOC | 0.2231‡ | 0.3104† | 0.2323† | 0.2132‡ | 0.2398 | 06 | 17 |
| QUERY-QUERY | 0.1996 | 0.2958† | 0.2094 | 0.1875 | 0.2402 | 65 | 02 |
| ASSOC-FULL(I) | 0.1966‡ | 0.2604 | 0.2115 | 0.1944 | 0.2167 | 06 | 17 |
| LINK-LINK | 0.1939 | 0.2708 | 0.2177 | 0.1799 | 0.2201 | 37 | 02 |
| FULL-FULL(II) | 0.1923 | 0.2813 | 0.2042 | 0.1819 | 0.2287 | 98 | 04 |
| ASSOC-FULL(II) | 0.1856‡ | 0.2875 | 0.2229† | 0.1924 | 0.2176 | 98 | 04 |
| FULL-ASSOC | 0.1804 | 0.2417† | 0.2052 | 0.1910 | 0.1954‡ | 06 | 17 |
| FULL-FULL(I) | 0.1607 | 0.2729† | 0.2083 | 0.1854 | 0.1806† | 10 | 25 |

*Table 5.4: Training results for query expansion using query associations. Performance of expansion techniques on the training data (TREC-9). We used these queries to determine parameter settings and included them for reference. Schemes are again ordered by decreasing MAP.*

An example of the behaviour of the ASSOC-ASSOC scheme illustrates its utility over the FULL-FULL approach. For the query “earthquakes” (TREC query 513), the average precision for the ASSOC-ASSOC scheme is 0.171, compared to 0.134 for no expansion and 0.116 for the FULL-FULL approach. This is a direct result of the choice of terms for expansion. For the ASSOC-ASSOC scheme, the expanded query is large and appears to contain only useful terms:

> earthquakes earthquake recent nevada seismograph tectonic faults perpetual 1812 kobe magnitude california volcanic activity plates past motion seismological

In contrast, for the FULL-FULL(II) scheme the query is the more narrow:

> earthquakes tectonics earthquake geology geological

These trends in the success of expansion with ASSOC-ASSOC are consistent with our empirical inspections of other queries.

The QUERY-QUERY, FULL-ASSOC, and LINK-LINK schemes offer limited benefit for expansion. Without association to documents, the queries are ineffective: this is probably due to the median length of the queries being two words, that is, the queries have very little content when not grouped together as associations. The FULL-ASSOC scheme is ineffective for similar reasons: query associations are an excellent source of expansion terms, but in order to identify which associations to retrieve candidate terms from, they need to be ranked in the initial step also. The LINK-LINK scheme is significantly worse than no expansion for the average precision measure; this is perhaps unsurprising, since anchor text has been shown to be of utility in home or named page finding tasks, while the queries we use are topic finding

| Type | Q1 | Median | Q3 | Variance |
|---|---|---|---|---|
| ASSOC-ASSOC | -0.0057 | 0.0040 | 0.0604 | 0.5814 |
| QUERY-QUERY | -0.0022 | 0.0000 | 0.0049 | 0.0688 |
| ASSOC-FULL(I) | -0.0044 | 0.0006 | 0.0458 | 0.8446 |
| LINK-LINK | -0.0067 | -0.0001 | 0.0019 | 0.0938 |
| FULL-FULL(II) | -0.0093 | 0.0000 | 0.0075 | 0.1272 |
| ASSOC-FULL(II) | -0.0028 | 0.0020 | 0.0295 | 0.4789 |
| FULL-ASSOC | -0.0259 | -0.0040 | 0.0197 | 0.3155 |
| FULL-FULL(I) | -0.0423 | -0.0072 | 0.0086 | 0.8487 |

*Table 5.5: Quartiles and variance of different expansion methods on the training data (TREC-9).*

tasks (see Broder (2002) for a classification of web queries). For these reasons we did not experiment further with expansion based anchor text, that is, we did not evaluate schemes such as FULL-LINK or LINK-FULL.

As discussed earlier, we tuned parameters prior to running experiments. Specifically, $|R'|$ and $|E|$ were identified through an exhaustive search on the same collection, but using TREC-9 queries. The results of this process with the TREC-9 queries are shown for reference in Tables 5.4 and 5.5.

We have tried similar experiments on TREC disks 4 and 5, which consist of newswire and similar data. These were unsuccessful. The problem appears to be that the query logs, drawn from the web, are inappropriate for this data: based on a small sample, it seems that many of the queries in the log do not have relevant documents. Thus the process of creating newswire associations from search-engine logs is unlikely to be successful. Query association based expansion is therefore only of utility if queries are available that are appropriate for the collection being searched.

## 5.4 Summary

Conventional wisdom has held that query expansion is an effective technique for information retrieval. However, recent experiments have contradicted this in some circumstances and shown that parameter settings that work well for one set of queries may be ineffective on another. In this chapter, we have investigated alternative techniques for obtaining query expansion terms, with the aim of identifying techniques that are robust for different query sets.

We have identified a successful expansion source for web retrieval. This source is query associations, that is, past queries that have been stored as document surrogates for the documents that are statistically similar to the query. In experiments with almost one million prior query associations, we found that expanding TREC-10 web track topic finding queries using query associations and then searching the full text is 26% – 29% more effective than no expansion, and 18% – 20% better than an optimised conventional expansion approach. Moreover, our results are significant under statistical tests. We conclude that query associations are a powerful new expansion technique for web retrieval.

# Chapter 6

# Efficient Local Analysis with Auxiliary Data Structures

As we have shown in Chapter 5, local analysis query expansion can significantly improve the effectiveness of retrieval tasks. However, since local analysis depends on a dynamically determined set of documents, these need to be retrieved from disk, which is costly.

It is therefore useful to consider whether it is possible to improve efficiency of local analysis algorithms and associated data structures. Although there has been a great deal of research on the efficient evaluation of ranked queries (see Section 2.4.4), there is no prior work on efficient local analysis for text retrieval. Query expansion through knowledge structures, such as the use of thesauri for global analysis (see Section 3.3.1), is typically very efficient, as only limited work is required during query evaluation. However, local analysis has been found to be superior in terms of effectiveness (Xu and Croft, 2000) and remains the focus of our efforts, although recently it was found that constructing a thesaurus from external sources can be beneficial for queries that are typically hard to answer (Voorhees, 2004). We use a conventional approach to local analysis as our baseline (a detailed explanation of local analysis is provided in Section 3.3.2).

In the previous chapter, we presented a technique for expanding queries using a synchronised side corpus consisting of query associations. The associations consisted of queries that were taken from a query log and run against the document collection. The resulting associated queries were grouped into documents and formed surrogates from which candidate terms were retrieved, instead of using the documents in the collection as a source of terms.

Since we placed the restriction on queries that all query terms must be present in the document a query was associated with, we have, in effect, created a summary of each document.

In this chapter we take this approach one step further, and propose that brief summaries of documents can be held in memory. The local set for query expansion is conventionally determined by ranking the document corpus, but here summary terms are directly accessed from a much smaller corpus that may fit in main memory, avoiding expensive disk accesses. Some of these terms are then chosen for expansion, using the heuristics described in Section 3.3.2. Using this approach, it is possible to approximate the effectiveness of the conventional approach to expansion, while significantly reducing the time required for query expansion by a factor of five to nine.

We also explore other alternatives for reducing the costs of local analysis, but these approaches compromise effectiveness so severely that they are not of practical benefit.

The remainder of the chapter is structured as follows. The steps involved in expanding queries using local analysis techniques are examined in terms of the effect on query throughput in Section 6.1. We then propose several approaches that are aimed at increasing efficiency while addressing the investigated practicalities in Section 6.2. Finally, we describe the experiments undertaken and analyse the performance of the new approaches in Section 6.3.

A preliminary version of this chapter appeared as Billerbeck and Zobel (2004*b*) and has been published – in a more extensive form – in Billerbeck and Zobel (to appear).

## 6.1 Local analysis practicalities

In most expansion methods that make use of local analysis, there are five key stages:

- **Initial ranking:** First, the original query is used to rank an initial set of documents.
- **Fetching documents:** This set is then retrieved from disk.
- **Extracting candidate terms:** All terms are extracted from the initial set of documents.
- **Selecting expansion terms:** Terms are ranked in order of their potential contribution to the query.
- **Final ranking:** The top-ranked terms are appended to the query, and finally the reformulated query is re-issued and a final set of documents is ranked.

We will investigate the impact on evaluation time each of these stages has in greater detail in this section.

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 also explore these options, providing a focus for the experiments reported later in this chapter. Some of the concepts introduced here – in particular, associations and surrogates – are described in more detail in Section 6.2.

### 6.1.1 Initial ranking

During the first stage, documents are ranked according to the original query. For each query term the inverted list is retrieved (unless it has already been cached from a previous access) and processed. As explained in Section 2.2, an inverted list specifies which documents a query 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 (Moffat and Zobel, 1996). 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 (see for instance Ramakrishnan and Gehrke, 2003, page 308).

Before examining these three quantities in more detail, we briefly describe the structure of a simple magnetic disk (Hennessy and Patterson, 2002, pages 680–684). A hard disk is made up of one or more platters. Logically the data is organised in concentric circles around the centre of the platter. These circles are also called *tracks*, or *cylinders* when mentioned in the context of multiple platters. The outer tracks can hold more data than the inner ones, since the circumference of those is larger and – at constant data density – more data can be stored. Each track is divided up into *sectors* that hold typically between half and four kilobytes of data (the average sector size is increasing as disk and main memory sizes increase over time). For each platter there is a *disk head* that moves over the platter, from the innermost to the outermost track.

- **Seek time:** The time taken for a disk head to move to the track where the desired data is stored, or where the first part of that data is stored. Currently, a typical hard disk has an average seek time of around 9 ms (any performance data of hard disks cited here are obtained from Patterson and Hennessy (2005, page 573)). 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:** The time taken for the disk to spin until the first sector that contains the data to be read is located underneath the disk head. Depending on the rotational velocity of the disk, the average rotational delay in modern disks is about 4 ms. Since the rotational delay is a reasonably small component of the total access time and it is difficult to reduce, we do not consider it further.

- **Transfer time:** The *transfer time* is the time required to read a file from the hard drive into memory once the head is positioned correctly. It 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 of the order of 50 MB per second. The transfer time for an inverted list of query terms would usually be smaller than the 9 ms cited above, since inverted lists in most systems are stored in compressed form, and the size of a typical inverted list for a query term is less than 1 MB for the collections we use. As an example, for the WT10g collection (see Section 2.6.2 for details) the compressed inverted list of a common term (such as "the") is up to 45 MB in size. This size shrinks to 2 MB if the list does not contain term offsets, but only the document numbers 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 term offsets). Term offsets are needed in order to evaluate phrase queries. We do not use term offsets in our experiments as we only use TREC title queries, as explained in Section 6.3.

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. For example, Anh and Moffat (2002) proposed a technique where documents are not stored in the order they are encountered during indexing, but in order of the impact that a term has in a particular document, as discussed in detail in Section 2.4.4. For instance, a term has a greater impact in a document in which it occurs twice than in another document of the same length in which it occurs once only. Using this ordering means that either the processing of lists can be stopped once a threshold is reached, or that the lists are truncated at an estimated threshold 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 impact ordering can provide are expected to be in addition to the gains that we achieve with our methods. Efficiency should be increased even more than can be expected from using impact-ordered indexes, since the expanded queries are – by definition – longer than original queries and therefore stand to gain considerably by using such indexes.

Another way to reduce list length, discussed in more detail in Section 6.2.2, is to index only a fraction of the document collection for the initial ranking. In all previous work – except for our work on query associations detailed in the last chapter and explained in the context of this chapter in Section 6.2.1 – ranking is performed on the document collection at hand, but there is no particular reason why other, smaller, collections should not be used. 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 less efficient, since in this case the index, including the in-memory dictionary and the inverted lists (stored on disk), is large. This means that disk seek and transfer times for the inverted lists are increased, while the likelihood of an inverted list being cached is decreased, given that memory volume is fixed and that the dictionary as well as all inverted lists cannot be held in memory simultaneously when dealing with large collections.

### 6.1.2 Fetching documents

Having identified the highly ranked documents, these need to be fetched. Assuming a typical memory size, these documents are not cached from a previous expansion or retrieval process 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 several orders of magnitudes

(Hennessy and Patterson, 2002, pages 390–391), and 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, this is the most expensive stage of expansion and therefore the area where the greatest gains are available. We have shown in the last chapter that surrogates – which are a fraction of the size of the documents – can be more effective than full-text documents. Using surrogates such as query associations is more efficient, provided that those surrogates can be pre-computed, as discussed in Section 6.2.1.

Another approach is to limit the number of documents available for the extraction of terms, which should result in higher efficiency, due to reduced cache misses when retrieving the remaining documents and otherwise smaller seek times, if 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 (Garcia et al., 2004).

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.

### 6.1.3 Extracting candidate terms

In the second phase of the expansion process, candidate terms are extracted from the previously fetched documents. 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. (We do not use stemming in our experiments.) 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 roughly proportional to 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.

### 6.1.4 Selecting expansion terms

In this phase, all candidate terms are considered, using the heuristics from whichever expansion method is being 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 term selection value (see Section 3.3.2) 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 much of it is typically cached in main memory. 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. As we use pointers to terms in the in-memory dictionary rather than storing the terms themselves (as explained in the previous stage), no mapping of terms to dictionary entries is needed for this approach.

### 6.1.5 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 in the previous chapter 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 in Section 2.4.4.

## 6.2 Methods of increasing efficiency for local analysis

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

### 6.2.1 Query associations

Query associations (Scholer and Williams, 2002) 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 in the query log is submitted to a search system in turn, 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 ranked 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 can be dynamically replaced with a new, more similar query. Associations are described in more detail in Chapter 5.

In the previous chapter, we have shown that associations are effective when appropriate query logs are available. 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 newswire collection used in TRECs 5 to 8, associations take up 73 MB. For the 10 GB WT10g collection from TREC 9 and 10, associations occupy 233 MB. The advantages of using associations are that they 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 contiguously on disk. The cost of the term selection phase is also lower, since fewer candidate terms need to be considered.

Rather than indexing the associations, it would in principle be possible to rank using the standard index, then fetch and expand from the associations, but in our previous chapter we found that it was by far more effective to rank against the associations themselves (see Tables 5.2 and 5.4).

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, which may incur additional disk accesses if associations are too large to be held in memory.

### 6.2.2 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 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 $n$th document, for $n$ of 2, 4, and 8. 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 Brin and Page (1998)), which we have not tested.

### 6.2.3 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 (Goldstein et al., 1999), and in particular Lam-Adesina and Jones (2001) have used summarisation for query expansion. Lam-Adesina and Jones use human-readable document summaries as term pools for document expansion. They experiment with query-biased summaries (which need to be generated at query time, requiring retrieval and parsing of documents; see for instance Tombros and Sanderson (1998)), as well as context-independent summaries, basing their work on Luhn (1958), who automatically created abstracts of technical papers and magazine articles. In their work, representative sentences are selected by attributing

**Pyongyang**'s response to the latest South **Korean** proposal for **resumption** of **inter-Korean talks** seems to score **polemical** points against the South and its preference for negotiating with Washington over **Seoul** on the **nuclear** issue.
**Pyongyang** countered **Seoul**'s proposal for a 1 March **working-level** contact with a later date, a counterproposal that seemed aimed mainly at gaining some perceived **polemical** advantage over **Seoul**. **Pyongyang**'s attempt to delay the North-South meeting for two days seemed to violate an agreement it had reportedly reached with Washington for **resumption** of **high-level** DPRK-U.S. **talks**. **Pyongyang** has been showing its concern about the issue of **nuclear** inspections and its continued **reluctance** to deal with anyone but Washington on the **nuclear** issue.
The party paper **Nodong Sinmun ridiculed** President **Kim** Yong-sam for allegedly acting "as if he receives U.S. reports on DPRK-U.S. **talks**".

*Figure 6.1: Example document (modified TREC document FBIS-3) with* **highly ranked *tf.idf* terms**. *Please refer to Table 6.1 for a ranking of those terms.*

significance scores to sentences that contain a high density of closely related terms, giving an abbreviated human-readable document. Lam-Adesina and Jones demonstrate relative improvements of roughly 10% over a standard local analysis approach (using the relevance weight described in Section 3.3.2 for selecting terms). Not surprisingly, perhaps, their best method is based on query-biased summaries, which cannot be pre-computed, and are therefore less efficient.

However, summaries to be used for query expansion do not need to be suitable for human consumption. Instead, we propose summaries that 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. We use the following *tf.idf* formulation, (as derived in Section 3.3.2):

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

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:

- Fixed number ($S$) of terms per summary.
  The first option is to use a fixed number of terms per document. The $S$ highest ranked terms for each document are used in their summaries.

| Summary term | Summarisation method | | |
|---|---|---|---|
| | Fixed number ($S$) | Global threshold ($C$) | Percentage of document ($P$) |
| inter-korean | 1 | 0.47 | 1.6% |
| pyongyang | 2 | 0.49 | 3.2% |
| resumption | 3 | 0.56 | 4.8% |
| polemical | 4 | 0.56 | 6.5% |
| seoul | 5 | 0.57 | 8.1% |
| sinmun | 6 | 0.59 | 9.7% |
| korean | 7 | 0.63 | 11.3% |
| high-level | 8 | 0.63 | 12.9% |
| ridiculed | 9 | 0.65 | 14.6% |
| reluctance | 10 | 0.70 | 16.1% |
| nodong | 11 | 0.70 | 17.7% |
| nuclear | 12 | 0.71 | 19.4% |
| working-level | 13 | 0.72 | 21.0% |
| kim | 14 | 0.75 | 22.6% |
| talks | 15 | 0.77 | 24.2% |

*Table 6.1: The highest ranked terms from the document shown in Figure 6.1, listed in order of descending* tf.idf *value. Any of three methods can be used in order to determine how many of the terms are included in the summary.*

- Terms with a weight lower than a global threshold ($C$).
  The second option is to choose a global threshold $C$, where each summary consists of all the terms appearing in the document at hand whose inverted *tf.idf* value (1/*tf.idf*) is lower than $C$.

- Have the summary size dictated by a percentage of the document size ($P$).
  The third option to compose summaries is to limit the volume of terms used in the summary to $P$% of the underlying document, where this percentage is limited to (the somewhat arbitrarily chosen number of) 100 terms so that extraordinarily large documents do not take up too much memory. The length of the document is measured in stopped, unique terms. Again, the terms with the highest *tf.idf* scores are chosen.

In each of these cases, terms for the document summary are chosen from a ranked list of *tf.idf* terms. See Table 6.1 as an example, where only the highest ranked terms are shown, as displayed for the example document in Figure 6.1. How queries are expanded from disk-based documents or in-memory summaries is also shown in Figure 6.2.

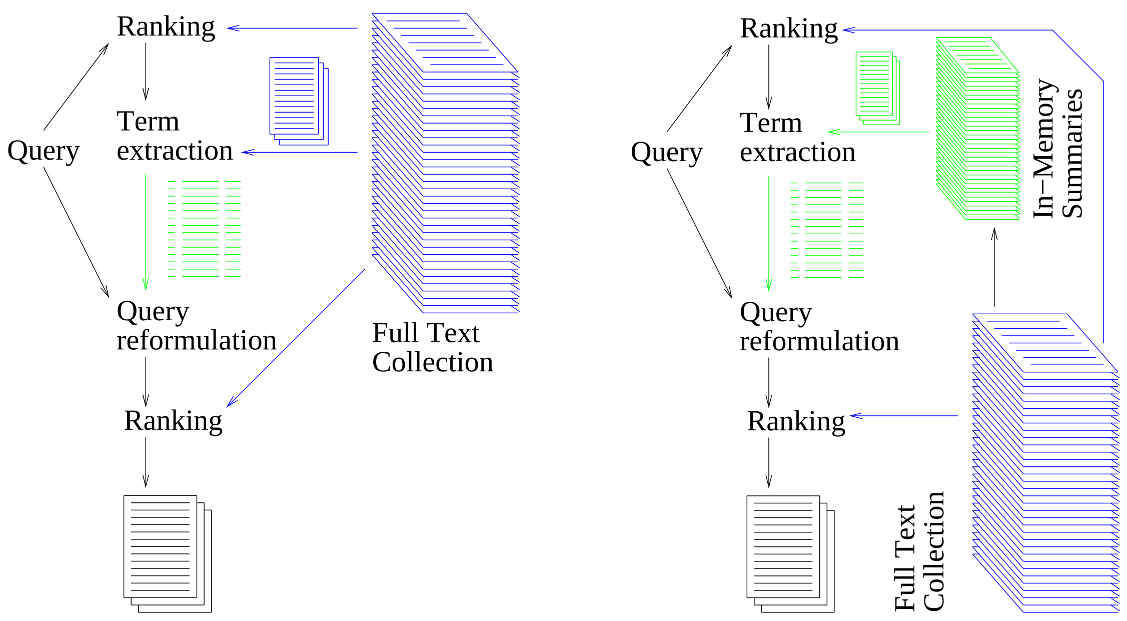


*Figure 6.2: Document and summary based local analysis query expansion. In both figures data that is stored on disk and held in memory is shown. The left figure depicts how queries are expanded through terms that are extracted from documents, whereas expansion terms are taken from in-memory summaries on the right hand side. The summaries are created at indexing time and are held in memory during query evaluation, limiting the need for disk accesses.*

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 $TSV$ (see Section 3.3.2). 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 means that 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 query-biased summaries could be more effective as a source of expansion terms, such a method cannot be applied in the context of efficient local analysis, as query-biased summaries cannot be pre-computed.

### 6.2.4 Other approaches

Since the original query terms effectively get processed twice during the ranking processes (once for the initial ranking of documents, and once more for the final ranking), it would be plausible to process the original query terms during the initial ranking, and subsequently process the expansion terms without clearing the accumulator table that was used for the initial ranking.

However, Moffat and Zobel (1996) demonstrated that limiting the number of accumulators may aid efficiency and effectiveness (we make use of this optimisation in all our experiments). 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 in Section 2.4.4.) This means that the accumulator table must be cleared after the initial ranking and 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 predominantly choose the documents in order to prevent query drift and give the highest impact to user-chosen terms. However, we found that it is essential to process the original terms a second time, potentially after having processed some of the expansion terms. For instance, we found that standard expansion improves MAP for TREC 7 queries from 0.191 to 0.229, whereas not clearing the accumulator and ranking using only expansion terms in the final ranking gives lower results of 0.222. The corresponding figures are 0.221, 0.248, and 0.242 respectively for TREC 8.

Processing only expansion terms in the second phase reduced costs, but leads to poor effectiveness. Even if we could use the existing accumulators and therefore not rank the original query terms again, the time savings achieved would be relatively small, because the original query terms are only a small minority of the overall number of terms in the expanded query and the inverted lists of the original terms are most likely cached from the initial ranking at any rate and would not need to be retrieved from disk.

Other strategies could also lead to reduced costs. Only some documents, perhaps chosen by frequency of access (Garcia et al., 2004) 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 (Witten et al., 1999, page 115) and a variable-byte compression scheme (Scholer et al., 2002). We leave the evaluation of these strategies for future work.

Note that our summaries have no contextual or structural information, and therefore cannot be used – without major modifications – in conjunction with methods that use such information, such as the local context analysis method of Xu and Croft (2000) or the summarisation method of Goldstein et al. (1999).

## 6.3 Experimental setup

We conducted a variety of experiments to evaluate the impact of these approaches on overall effectiveness, and whether they lead to reduced query evaluation time.

The search engine that we used for our experiments is described in Section 4.1. However, for the experiments detailed in this chapter and in Chapter 7, we made several changes to the search engine:

in contrast to the experiments run in Chapters 4 and 5, any indexes used do not contain term offsets; the vocabulary is initially stored on disk, but term information is cached permanently, once it has been retrieved for the first time. Finally, we indexed only terms that contain no more than four non-alphabetical characters (Williams and Zobel, 2005). This reduces the total number of unique terms by between 30% to 40% (depending on which collection is being indexed), and leads to a similar decrease in the combined size of the inverted lists. In addition, it increases the likelihood that in-memory summaries contain useful candidate terms, as “junk terms” – such as terms that only appear once in the whole collection – are also likely to have a high inverse document frequency component of the *tf.idf* score but are typically not useful for expansion.

The test data is drawn from the TREC conferences, outlined in Section 2.6.2. We used two collections: the first is composed of newswire data, used at TRECs 7 and 8. The second is the WT10g collection, consisting of 10 gigabytes of web data crawled in 1997 (Bailey et al., 2003), used at TRECs 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 (see Figure 2.2). We use the Wilcoxon signed rank test to evaluate the significance of the effectiveness results, as described in Section 2.6.3.

For timings, we used 10,000 stopped queries taken from two query logs collected for the Excite search engine (Spink et al., 2002); these are web queries and thus suitable for the WT10g experiments. 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 (Billerbeck and Zobel, 2004*b*).

## 6.4 Results

We used the TREC 8 and 10 query sets to test parameter settings. Results are summarised in Table 6.2 and details are shown in Tables 6.3 and 6.4. We applied the best methods found in these tables to the TREC 7 and 9 query sets, as shown in Table 6.5, without any further tuning of our approach. 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 local analysis improves over the baseline, but in both cases query evaluation takes around nine times as long.

### 6.4.1 Expansion using query associations

Several of the methods proposed do not succeed in our aims. Expanding queries using associations takes nearly as long as standard local analysis, and effectiveness is only marginally increased over the baseline.

| Data set | Expansion Method | Query Time (ms) | MAP | P@10 | R-Pr. | 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 |

*Table 6.2: Summarising the best effectiveness results from tuning collections TRECs 8 and 10 shown in Tables 6.3 and 6.4, respectively.*

(Due to developments in the software (described in Section 6.3) some of the baselines have changed from those detailed in Chapter 5. We spent considerable time to run follow-up experiments explicitly seeking the cause for this discrepancy. Although we were not able to identify the cause we are confident that the experiments detailed in both chapters are valid.) For TREC 8 the surrogates are arguably inappropriate, as the web queries used to build associations may not be pertinent to the newswire data; however, this issue highlights the fact that associations cannot be used without an appropriate query log.

### 6.4.2 Expansion through collection subsets

Using halves ($n = 2$), quarters ($n = 4$), or eighths ($n = 8$) 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. Results for different halves and quarters are shown as Quarter 1, Quarter 2, and so on in Tables 6.3 and 6.4. Larger $n$ led to smaller improvements in query expansion; in experiments with $n = 8$, expansion gave no improvements. Reducing the local set $R$ to roughly a quarter of its original size in order to cater for a smaller number of relevant documents – as intuition might suggest – only further degrades results. These results are consistent with previous work which shows that retrieval effectiveness, especially in the top-ranked documents, is greater for larger collections than for sub-collections (Hawking and Robertson, 2003). This means that there is a higher likelihood of sourcing expansion terms from relevant documents when using local analysis if the largest collection is used. It was also found that expansion works best when terms are sourced from collections that are a superset of documents of the one targeted (Kwok and Chan, 1998).

| Expansion Method | Query Time (ms) | MAP | P@10 | R-Pr. | 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 |

*Table 6.3: Effectiveness of expansion of TREC 8 queries on the TREC newswire data. Results shown are mean average precision (MAP), precision at 10 (P@10), and R-Precision (R-Pr.). 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. Figures in* ***bold*** *are the best results for the summary based technique.*

| Expansion Method | Query Time (ms) | MAP | P@10 | R-Pr. | 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 6.4: Effectiveness of expansion of TREC 10 queries on the WT10g collection. See Table 6.3 for an explanation of methods and symbols used.*

| Data set | Expansion Method | Query Time (ms) | MAP | P@10 | R-Pr. | 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 |

*Table 6.5: As in Table 6.2, but showing results for test collections TRECs 7 and 9.*

### 6.4.3 Expansion using in-memory summaries

Simple *tf.idf* summaries work well. Even one-term ($S = 1$) summaries yield significantly improved average precision on TREC 8, for a memory overhead of only a few megabytes. The best results were obtained for $S = 60$ on TREC 10 and $S = 76$ on TREC 8; total processing costs were less than one-third those of standard local analysis, making this method of expansion between five (summary based expansion time overhead / conventional expansion time overhead = (83ms - 30ms) / (296ms - 30ms) = 19.9%) and nine times (9 ms / 83 ms = 10.8%) more efficient than standard local analysis. These gains are similar to those achieved by Lam-Adesina and Jones (2001) with summaries of between six and nine sentences each, but our summaries are considerably more compact, showing the advantage of a form of summary intended only for query expansion. 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 using the summaries on the test set for the newswire data (TREC 7) are equally satisfactory, with good effectiveness and low overheads. Results on the test set of queries for the web data (TREC 9) are, however, disappointing. We had already discovered in our previous chapter that expansion on TREC 9 does not improve effectiveness, and Table 6.5 shows that the standard local analysis approach leads to worse results than no expansion; in that light, our results here are unsurprising. The principal observation is that query expansion based on summaries is still of similar effectiveness to that based on full documents, while offering efficiency benefits.

Figure 6.3 shows how much time during query evaluation is spent on the different phases necessary to expand queries as described in Section 6.1. These graphs show that fetching of documents is

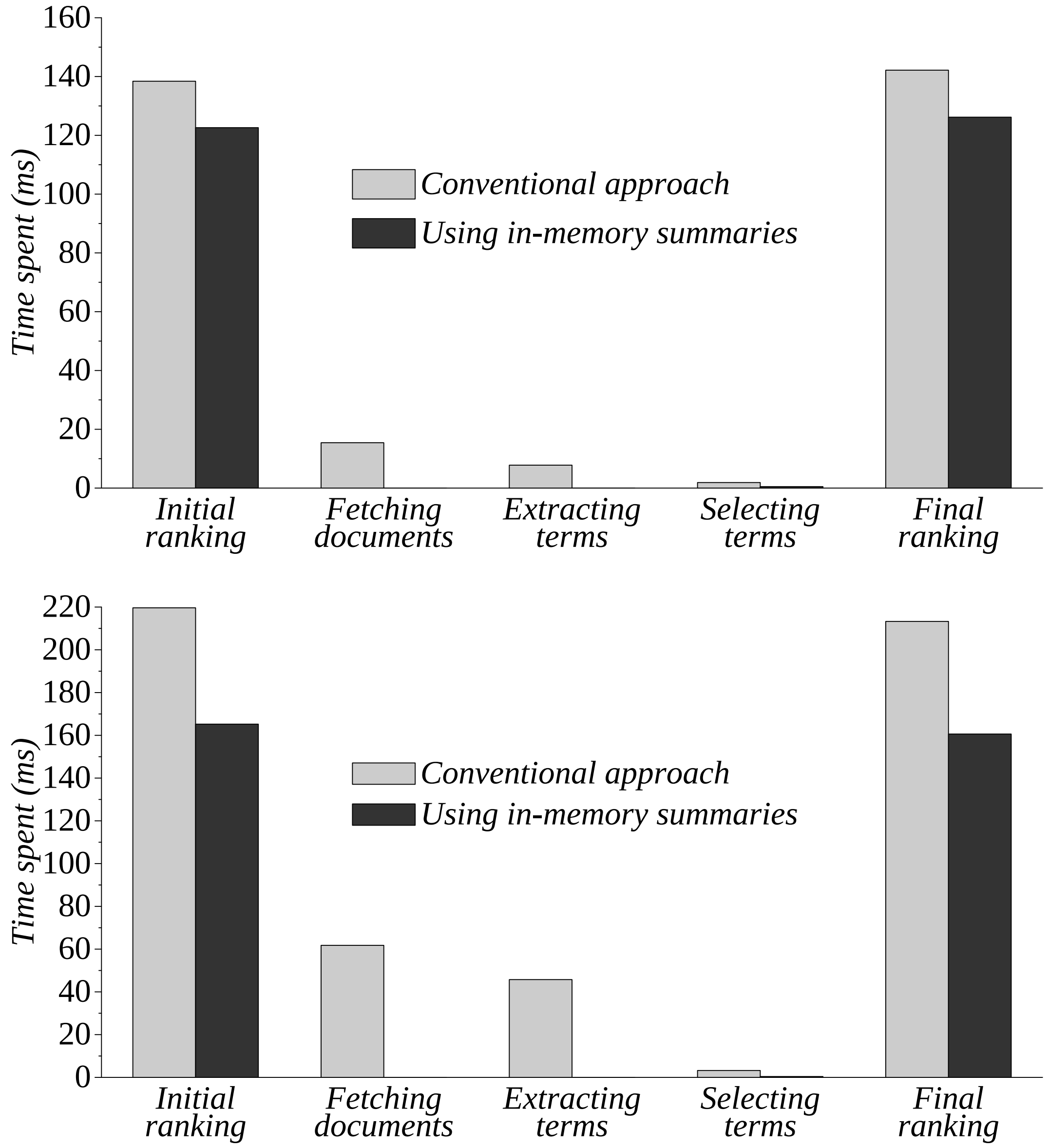


*Figure 6.3: The amount of time spent in the different stages necessary for local analysis during query expansion. The top graph shows the timings for the newswire collection, whereas the bottom is based on the WT10g collection. In each case, the $S$ approach was used to create summaries, with the best values (that is 60 and 76 respectively used). Interestingly, the time need to rank documents is reduced due to improved caching of lists.*

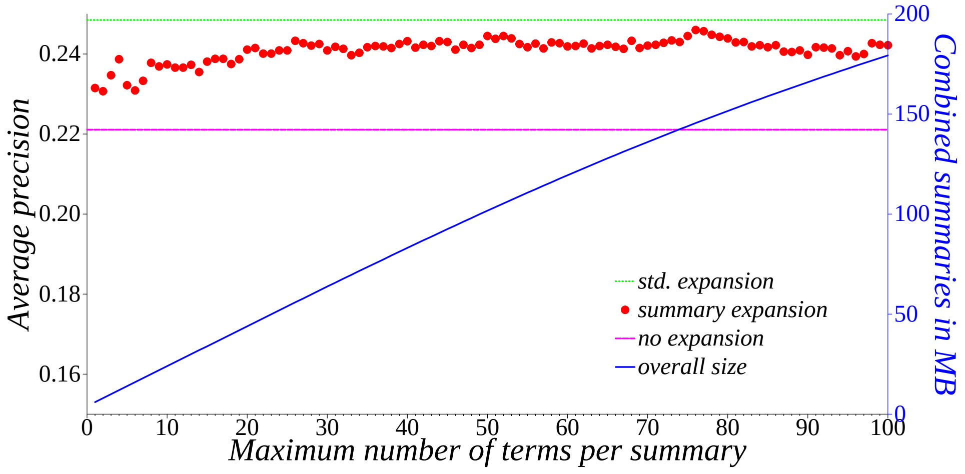
a relatively large component of the query expansion process and is not needed when using in-memory summaries. They also confirm that time spent accessing terms from the in-memory summaries is negligible when compared to time spent extracting terms from documents that need to be parsed after having been retrieved from disk. Furthermore, term selection costs are drastically reduced as well, since much fewer terms are available for selection using the summaries. Although the graphs in Figure 6.3 seem to indicate this time is not reduced to 0, the term selection time is roughly a quarter of the time needed for the conventional approach.

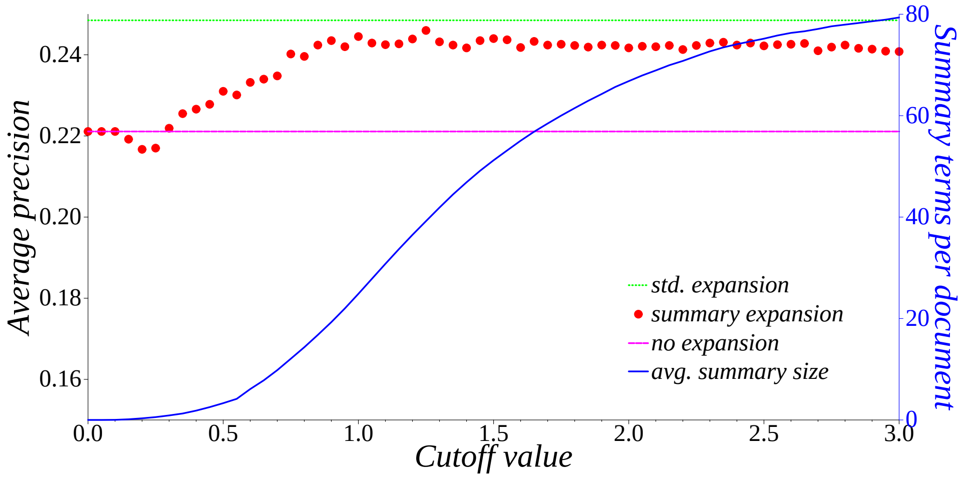
Unexpectedly, the times needed to rank the documents against the unexpanded and the expanded queries are reduced (this is particularly pronounced with the larger WT10g collection). The explanation for this is that – although a large amount of memory is used for the summaries that are held in memory – overall the ability to cache inverted lists is improved since no lists are swapped out of memory in order to make room for documents that are retrieved. Although the documents loaded take up a relatively small amount of memory, the random nature of loading up many documents destroys locality of the cache. This reduction in ranking time is even greater than the combined savings achieved by not having to fetch documents from disk or parsing those. To ensure that the effects of caching can be properly observed, we averaged the query times that were used to produce the graphs in Figure 6.3 over 100,000 queries (that is 10 times the amount of queries used for other experiments detailed in this chapter).

### 6.4.4 Sensitivity of expansion to summary parameters

We show results for several parameter settings. These lead to comparable effectiveness for similar memory overhead. The choice of $S$, $C$, and $P$ for determining the size of summaries is further examined in Figures 6.4 and 6.5 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 local analysis. Note that the total memory overhead for the in-memory summaries are not linear in the top graphs of Figures 6.4 and 6.5, as it may appear to be, but rather the amount of memory used increases more slowly than the number of terms held. This is the case as we include the total size of all associated data structures needed and a certain amount of space is a fixed overhead for holding the in-memory summaries and associated pointers.

The middle graph in Figure 6.4 shows a drop in average precision if summaries consist only of terms with extremely low inverted *tf.idf* values, of below $0.25$. This would suggest that average precision could be optimised by using only terms that have an inverted *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 6.3 and 6.4, this is not the case, which leads to the conclusion that some terms with a low inverted *tf.idf* value are chosen as expansion terms from a small pool of candidate terms to the detriment of queries.





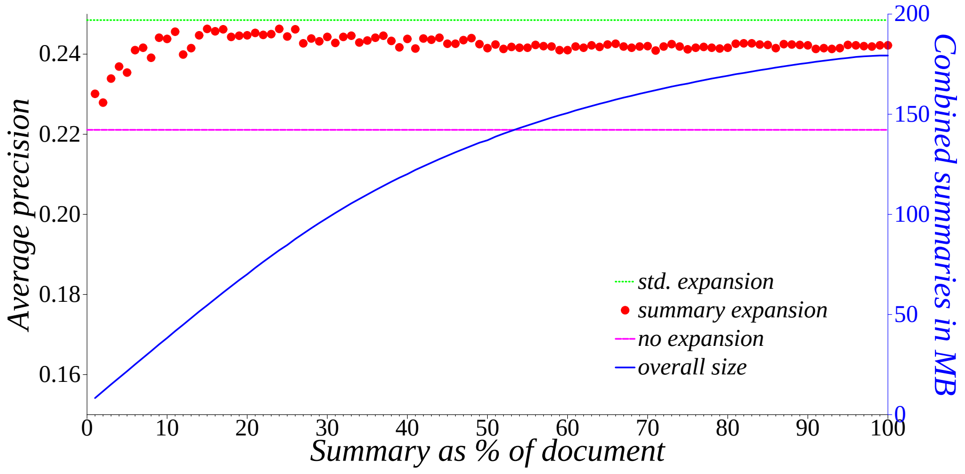


*Figure 6.4: Varying average precision and associated memory cost with the number, cutoff value of summary terms, and percentage of document used for summaries respectively. Results were obtained using the TREC 8 data set.*

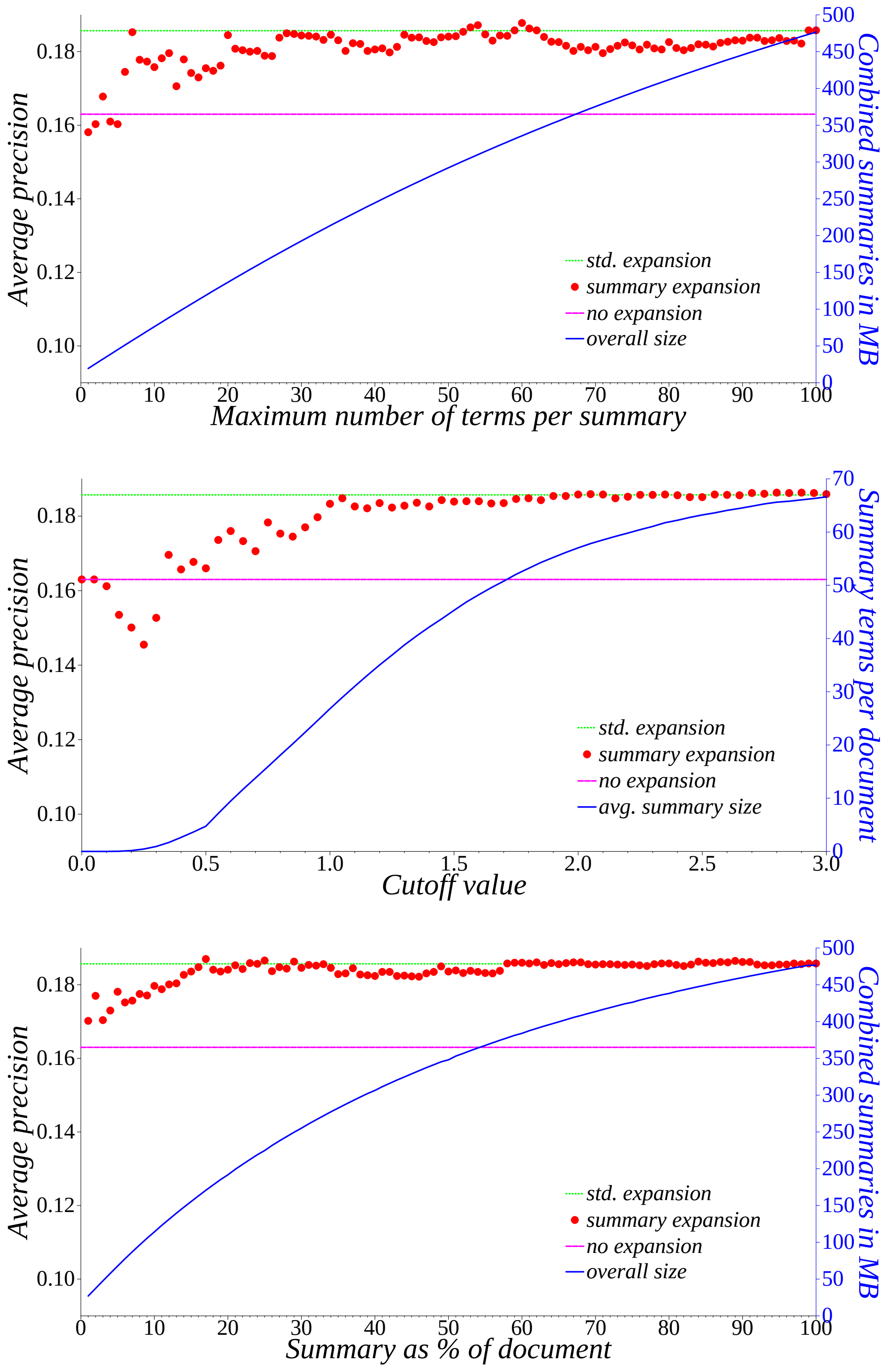


*Figure 6.5: As in previous figure, except that the TREC 10 collection and queries are used.*

### 6.4.5 Reliability of expansion methods

Figure 6.6 compares the robustness of the different expansion methods on the training and test data shown in Tables 6.2 and 6.5. As can be seen, the effectiveness of our proposed summaries is comparable to that of the standard local analysis approach. However, improvements through summary-based query expansion are generally more moderate and a minority of queries are degraded more severely. This is most noticeable for the TREC 9 query set.

## 6.5 Summary

We have identified the main costs of query expansion 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 use of *tf.idf* summaries. These are much smaller than the original collections, yet are able to provide effectiveness close to that of standard local analysis. 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 fixed 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 the selection and extraction of candidate terms is trivial.

Many of the methods we explored were unsuccessful. Associations can yield good effectiveness if a query 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 local analysis query expansion while reducing query evaluation time by a factor of five to nine.

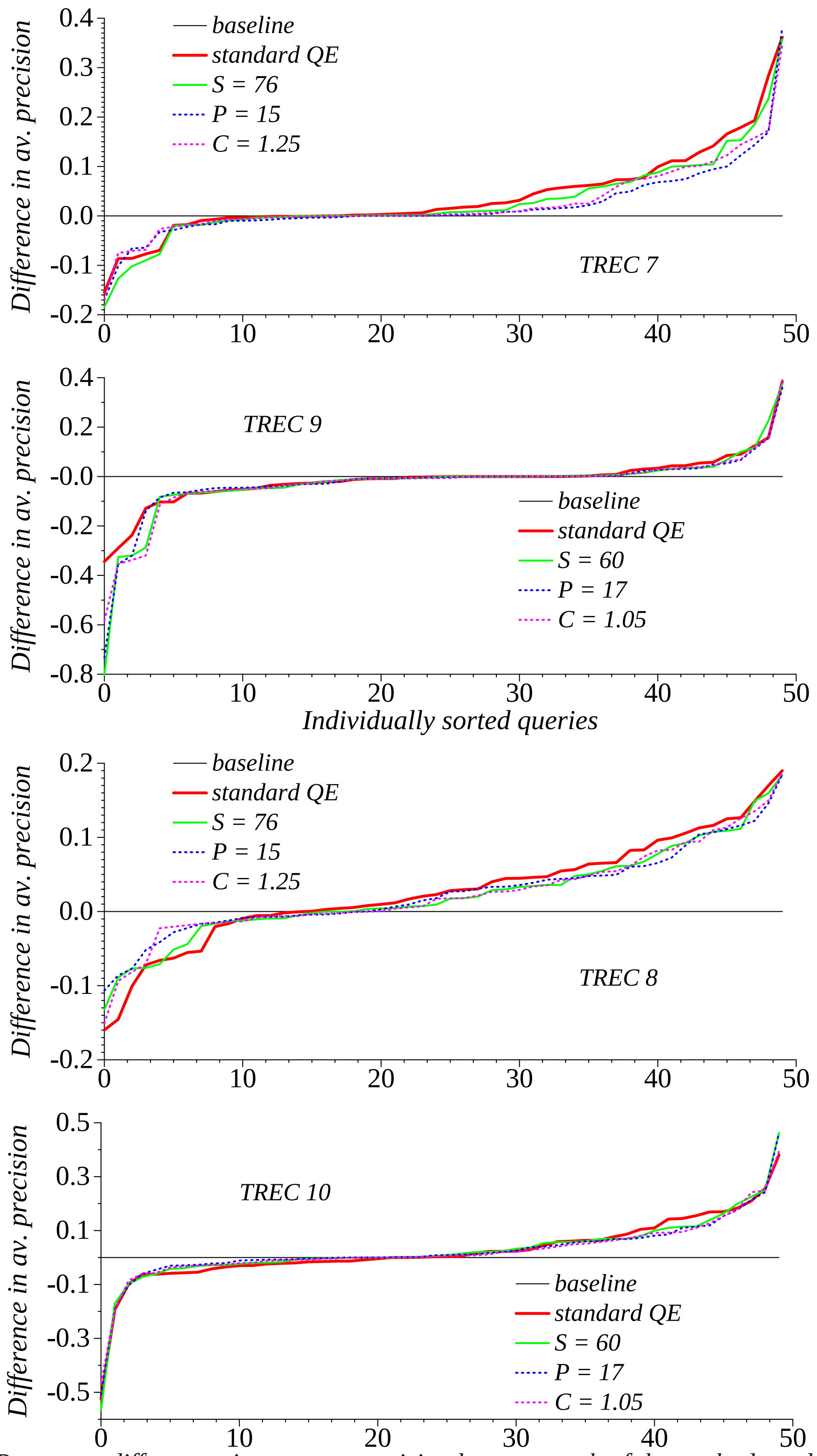


*Figure 6.6: Per-query-differences in average precision between each of the methods and the respective baselines on test and training data. Queries are individually sorted by increasing MAP values (normalised by the baseline).*

# Chapter 7

# Document Expansion for Ad-hoc Retrieval

As has been discussed in the previous chapters, query expansion can improve queries and thus make the search experience for a user more satisfying. However, a disadvantage of query expansion is the inherent inefficiency of reformulating a query. These inefficiencies have largely not been investigated, with the exception of our work reported in Chapter 6 and comments on the effect of larger number of documents used in the local set by Gu and Luo (2004). In Chapter 6 we proposed improvements to the efficiency of query expansion by keeping a brief summary of each document in the collection in memory, so that during the expansion process no time-consuming disk accesses need to be made. While some of the methods that we proposed maintain effectiveness, expanding queries using the best of these methods still takes between 30% to 150% longer than evaluating queries without expansion, depending on collection size and system hardware.

The best possible result for efficient query expansion – the principal topic of this thesis – is to not have to expand queries at query time at all, but still benefit from alleviation of term mismatch. A promising avenue to achieve this is *document expansion*, which we explore in this chapter as an alternative to query expansion. In document expansion, documents are enriched with related terms. When expanding documents all costs are incurred at indexing time, and there is only a marginal cost at retrieval time.

Most retrieval systems match documents and queries on a syntactic level, that is, the underlying assumption is that relevant documents contain exactly those terms that a user chooses for the query (as explained in Chapters 2 and 3). Typical search engines therefore only return documents that contain at least one of the terms in the query. However, Furnas et al. (1987) found that two users, who are asked to describe a certain topic with particular keywords, choose the same keyword with a likelihood of less than 20%; and we have observed that many documents that have been judged relevant in the TREC context (see Section 2.6.2) do not contain query terms. Figure 1.1 in Chapter 1 shows that as many as 25% of documents that are judged to be relevant do not contain *any* terms that appear in a concise query. The actual proportion might be much larger than this figure suggests, since in the TREC framework – from which the graph was produced – the relevance of only a relatively small number of documents is judged for each query. Furthermore, the documents that are judged are generally found through direct matching of query terms against document terms. Thus term mismatch (the problem that the graph is illustrating) may result in a much larger number of relevant documents not being identified. Document expansion, as proposed in this chapter, adds related terms to a document, so that those documents will be included in the ranking.

We propose two new corpus-based methods for document expansion. The first method is based on adding terms to documents in a process that is analogous to query expansion: each document is run as a query and is subsequently augmented with expansion terms. The second method is based on regarding each term in the vocabulary as a query, which is expanded using local analysis query expansion and used to rank documents. The original query term is then added to this new set of top-ranked documents.

Our experiments measure the efficiency and effectiveness of query expansion and document expansion on several collections and query sets. We find that, on balance, document expansion leads to improvements in effectiveness, but few of the measured gains are statistically significant. The computational cost at query time is small, as one would expect. In contrast, both standard query expansion and the efficient query expansion that we proposed in Chapter 6 lead to gains in most cases, many of them statistically significant. Our efficient query expansion technique incurs less than twice the cost of querying without expansion.

Our experiments were, within the constraints of our resources, reasonably exhaustive. We experimentally evaluated several alternative configurations of document expansion and explored the parameters, but did not observe useful gains in effectiveness. We conclude that corpus-based document expansion is unpromising. We did not explore query expansion to the same extent, yet found effectiveness to consistently improve, and thus believe that further gains in performance may be available.

A preliminary version of this chapter appeared as Billerbeck and Zobel (2005).

## 7.1 Document expansion background

While document expansion has recently been applied in various areas of information retrieval, it has not been used as an alternative to query expansion in order to improve ranking effectiveness (with one exception, detailed below). In this section, we discuss previous work on document expansion and show the areas where it has been applied.

Ide and Salton (1971) propose a technique for updating vector representations. Unlike the document expansion techniques considered in this chapter, they use relevance feedback, relying on the help of the user (see Section 3.1). In their work, the query representation is changed to obtain a query vector that is closer to that of the relevant documents (for instance, by adding those terms to the query that appear in many of the relevant documents); this is similar to the Rocchio method (see Section 3.1.2). They also propose a second method, where the document vector space is changed so that relevant documents are closer to the query vector. One of the approaches adds query terms to the vectors of relevant documents (or increase the weighting of those terms). Another approach is to interchange the vector space representation of two documents – one relevant and one non-relevant – with respect to a query. Thereby the vector of a document that was found to be relevant to the query is moved closer to the vector space representation of that query, while the vector of another document, that was found to be non-relevant, but was originally in closer proximity of the query vector, is moved further away, by respectively adding to or subtracting from the weights of query terms in those vectors. In either case the document vectors are normalised to retain their original vector length; that is, the weights of terms already in documents are reduced. Using these methods, they achieve effectiveness improvements of 10% to 15%.

True document expansion (that is, not just manipulating document vectors, but actually adding terms to documents) was first used by Singhal and Pereira (1999) in the context of speech retrieval. Although speech recognition has since improved, at the time of publication of their work, speech recognition was unreliable with an error rate of up to 60%. Singhal and Pereira expand transcribed documents with related terms from a side corpus (not unlike the methods we considered in Section 3.3.1). This method achieves a relative increase in MAP of 12% over a baseline that was established employing pseudo relevance feedback based on the technique proposed by Rocchio (1971).

Li and Meng (2003) use document expansion for *spoken document retrieval*, where they expand documents by augmenting them with highly valued *tf.idf* terms that have been retrieved from a side corpus. Their method is very similar to that of Singhal and Pereira. Li and Meng found a 56% relative improvement in Cantonese monolingual retrieval and 14% relative improvement in Mandarin cross-language retrieval.

Both Lester and Williams (2002) and Levow and Oard (2002) have used document expansion for topic tracking. Whereas Lester and Williams use document expansion to enrich topic profiles and do not specify whether it bears any benefit, Levow and Oard obtain consistent improvements in Mandarin cross-lingual retrieval by expanding the documents to be tracked.

*Document-self expansion* was used by Tseng and Juang (2003) for text categorisation. They propose two techniques. The first is summary-based expansion, where top-ranked sentences from each category are combined to form new documents in their respective categories. The second is term-based expansion, where representative terms for categories that only contain a few training documents are used to form documents, consisting of only that term. Document expansion here is somewhat of a misnomer since Tseng and Juang produce new documents rather than expanding existing ones. A better name might have been *collection expansion*.

In the context of latent semantic indexing (see Section 3.3.1), Cristianini et al. (2002) consider "a kind of document expansion" in order to link documents that share related terms. To this end they briefly consider expanding documents by adding all synonyms of terms contained within that document; however, they do not describe any experiments making use of document expansion.

Scholer et al. (2004) augment documents by associating queries (see Section 5.1.3) obtained from a query log in order to increase retrieval effectiveness. However, they do not reduce the problems of vocabulary mismatch, as they only add queries to documents where all query terms are already part of the document. Instead, they emphasise terms that are central to a document. The differences between our method explained in this chapter and query association are discussed in more detail in Section 7.2.2.

With the exception of Lester and Williams (who use expansion only on translated documents) and Cristianini et al., all previous work mentioned above uses document expansion in the context of enriching possibly incorrectly translated documents. Additional work on document expansion has been in the area of speech recognition and speech retrieval (particularly at TREC); as this work is not directly related to our proposed document expansion approaches, we do not survey it here.

The only direct reference to document expansion for document retrieval was made by van Rijsbergen (2000), who pondered whether document expansion could be used in this context. However, no experiments are reported in his paper.

## 7.2 Methods of document expansion for document retrieval

Rather than expanding a query from an initially retrieved set of documents, which is time-consuming, document expansion modifies documents by adding potential query terms that occur in similar documents. While this expansion process is reasonably costly, it is done at indexing time. Query times are only slightly increased (compared to conventional query expansion), since inverted lists are on average slightly longer (10%, say), depending on which document expansion method is chosen.

There are several ways to expand documents. All methods aim to eliminate inefficient run-time query expansion, while benefiting from the effectiveness of a local analysis mechanism. Each of the following proposed methods makes use of local analysis at indexing time and expands the original documents with additional terms.

There are some parallels between document expansion and smoothing, as employed by language modelling approaches to information retrieval (see Section 2.4.3). However, in contrast to smoothing, document expansion leads to a document being augmented with terms that are related to the topics covered in the respective documents, which tend to be relatively rare across the collection. Smoothing can be seen as enriching a document mainly with high frequency, and, to a lesser extent, rare terms.

### 7.2.1 Expansion via documents as queries

For this approach to document expansion, each complete document is run as a query. Standard local analysis query expansion (see Section 3.3.2 and Figure 7.1a) is performed and a set $E$ of expansion terms for this document is defined. The top $|E|$ candidate terms are then simply appended to the document.

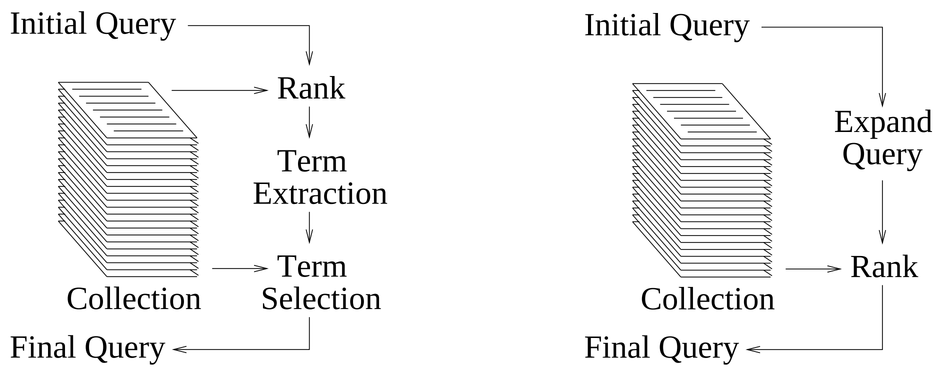


*Figure 7.1: The figure on the left (a) shows the central part of any expansion process proposed in this chapter. The right hand figure (b) shows how this is conventionally used for query expansion.*

This method is conceptually similar to conventional query expansion (see Figure 7.2a). We refer to this approach also as *document centric expansion*.

Although the process of expanding documents in this way is reasonably time consuming, it can be sped up considerably by, for instance, using only the top *tf.idf* terms from each document to formulate the query that will determine the local set from which expansion terms are sourced, or by making use of impact ordered indexes (see Section 2.4.4). For the experiments detailed in this chapter we did not make use of impact ordering. Both approaches are left as future work.

In Section 7.4.1 we give further details (such as choice of document expansion parameters) that have arisen from tuning with a training data set and explain other practicalities.

### 7.2.2 Expansion via vocabulary terms as queries

Our other approach to document expansion mimics more closely a reversal of the conventional local analysis algorithm. Imagine a query that consists of only one term. The role of query expansion is to identify documents that are *about* this term, but do not necessarily include this term. This is done by adding terms to the query that co-occur with the query term within the local set of documents. After expansion, we may therefore retrieve documents that do not contain the query term itself but that do contain expansion terms. Document expansion inverts this scenario: it puts the query term into those documents that contain the expansion terms. This has the effect of adding potential query terms that are on the same topic as the document, but are missing from that document. In other words, we aim to add those terms to a document that would have lead to the document being ranked highly if that term had been run as the only original term in an expanded query. Our hypothesis is that this technique will be a good proxy for expansion of queries consisting of single terms. However, it is by the same reasoning a less good match for the case of multi-term queries, as we perform the document expansion based on single vocabulary terms only.

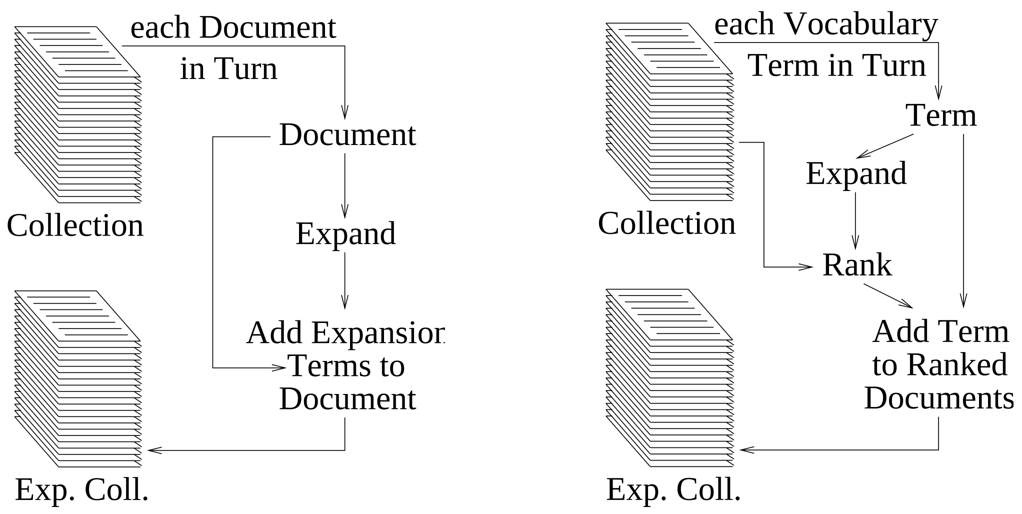


*Figure 7.2: The left hand figure (a) shows how documents can be expanded by running each document as a query and adding the expansion terms back into the document. In the right figure (b) documents are expanded by running vocabulary terms as a queries and adding the terms to the top ranked documents. "Expand" in either graph refers in either case to the conventional expansion process shown in Figure 7.1a.*

Formally, the algorithm of this approach is to run each single vocabulary term $t$ in turn as a query $q$. This one-term query is then used to rank a local set $R'$, consisting of $10$ documents. From this set, the standard local analysis algorithm (see Section 3.3.2) is used to obtain a set $E$ of $25$ expansion terms that are added to the single original query term $t$, building $q'$. The new query $q'$ is run against the collection. For each of the top 100 documents that are returned, the triplet of $t, d, s$ is stored, where $s$ is the similarity score between query $q'$ and document $d$.

Once all terms in the vocabulary have been processed in this way, terms from the list of triplets are added to the current document. We choose those terms with the highest similarity score $s$, and add terms until the size of the document has been increased by 10%. See Figure 7.2b for an illustration of this process, which we refer to as *term centric expansion*. Algorithm 1 also gives a formal definition of this approach to document expansion.

Term-centric expansion is considerably faster than the document-centric expansion method previously described. However, a problem that does not occur with the previous method arises in a setting where the collection grows, as frequently occurs in a web setting, for instance. Since the basis for selecting terms with which documents are enriched, that is the collection as a whole, is changing with the addition of new documents, the expansion terms chosen for a particular document might not be the best suited any more. (This is not a problem with the document-centric expansion method to the same extent, since each new document can still be expanded individually against the old collection without a large deviation from the best possible expansion terms, until the collection has changed to a large de-

**Algorithm 1** Term-centric document expansion

```
1:  for all terms ($t$) in a vocabulary ($V$) do
2:      run $t$ as a query $q$
3:      rank documents $d$ against $q$
4:      select top $10$ documents $R'$ as the local set
5:      rank candidate terms using TSV
6:      select top $|E| = 25$ terms and append to $q$, forming $q'$
7:      rank $q'$ against collection and take top $100$ documents ($X$)
8:      for all documents $d$ in $X$ do
9:          save $t, d, s$ triplets ($s$ is the similarity score of $q'$ to $d$)
10:     end for
11: end for
12: for all documents $d$ in collection $D$ do
13:     determine length $l = |d|$ of document $d$
14:     select $0.1 \times l$ of terms with the highest $s$ and add to $d$
15: end for
```

gree.) Furthermore, it is difficult to determine the best expansion terms for the added documents, as those documents did not exist when documents were originally ranked against terms. An – admittedly very expensive – solution would be to re-run the document expansion process every time a certain number of documents has been added. Optimisations may be possible but are outside the scope of this thesis. These could be revisited in future work.

Our proposed document expansion techniques involve a number of parameters that need to be chosen; we discuss these in Section 7.4.1.

Query association, as described in Section 5.1.3, differs from the above techniques in several ways. First, our techniques have the extra step of expanding a query with terms before ranking documents that a query becomes associated with. Second, associations are based on external information in the form of query logs, whereas document expansion relies on within-collection data and statistics only. More importantly, the query association results (Scholer et al., 2004) show that augmenting documents with associated queries works best when placing the restriction on queries to be associated with a document that all query terms must be present in the document. The effect of this restriction is that pertinent terms in a document are emphasised (their term count is increased and therefore the ranking of those documents is improved subsequently when query terms are used that appear in past queries that have been associated with that document), rather than new terms – which address the problem of vocabulary mismatch – are added to the document. Adding new terms makes a document retrievable to queries that originally would not have ranked this document, even though it may be on the same topic.

### 7.2.3 Expansion via phrases as queries

An extension to the document expansion approach described above is to use phrases rather than individual vocabulary terms for term-centric document expansion. This addresses a potential shortfall of the method above, which is a good match for queries consisting of single terms only.

Phrases are extracted as suggested by for instance Kim and Wilbur (2001). A phrase consists of two or more contiguous terms occurring in any document that are not separated by either a stop word (our stoplist contains 477 items), an HTML or TREC tag, or any of the following characters: ?!,;:(){}[]. Note that this is a much cruder attempt at identifying phrases than the approaches recently used by Joho and Sanderson (2000) and Liu et al. (2004). We leave using a more sophisticated approach of identifying phrases for future work.

In separate experiments, we use maximal-length phrases, and overlapping two-term phrases that were derived from the maximal length phrases. The phrases are then added to documents, using the term-centric document expansion method.

## 7.3 Experimental setup

We evaluate the proposed document expansion approaches using the Zettair search engine, described in Section 6.3. Although the local analysis parameters $|E|$ and $|R'|$ are collection dependent, we did not tune those for each collection. Instead we use the default parameters of $25$ and $10$, respectively, in all cases, since these are more easily comparable with other approaches detailed in the research literature and secondly, they have been found to be reasonably good for some of the target collections (see Section 4) whereas for others no tuning information is at hand.

### 7.3.1 Test collections

All our test data is drawn from the TREC conferences (see Section 2.6.2). To tune document expansion parameters and choose appropriate selection measures we used the Wall Street Journal articles from TREC disk 2, which consists of articles from 1990–1992. We refer to this subset as WSJ2. With this collection we used the TREC 3 topics 151–200 and associated relevance judgements. In all our experiments we used the title field only as queries.

We used several collections to evaluate our techniques. These include the Associated Press data (AP) from disks 1 and 2 to match the TREC 3 topics and relevance judgements. We also used the newswire collection (which we call NW) from TREC 7 and 8. This collection is drawn from TREC disks 4 and 5, without the congressional record collection. NW was used as a whole, and also as several sub-collections, namely the Financial Times 1991–1994 (FT), the Foreign Broadcast Information Service (FBIS) and the LA Times (LA). We tested these collections against topic sets of TRECs 6, 7, and 8.

| **Expans. method** | **Time (in milliseconds)** | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **WJS2** | | **AP** | | **NW** | | **FBIS** | | **FT** | | **LA** | |
| | Lrg | Ltl | Lrg | Ltl | Lrg | Ltl | Lrg | Ltl | Lrg | Ltl | Lrg | Ltl |
| None | 4.7 | 7.4 | 6.9 | 11.7 | 11.4 | 22.8 | 5.0 | 8.0 | 6.8 | 12.1 | 6.8 | 11.3 |
| QE | 25.4 | 47.1 | 29.0 | 52.5 | 145.9 | 211.2 | 49.4 | 123.5 | 41.2 | 87.6 | 32.6 | 62.3 |
| $S = 40$ | 7.5 | 14.8 | 11.4 | 22.5 | 20.9 | 52.0 | 8.1 | 16.3 | 11.8 | 24.4 | 10.6 | 20.6 |
| $Q = D$ | 5.3 | 8.6 | 7.7 | 13.3 | 12.1 | 24.8 | 5.5 | 9.4 | 7.6 | 13.7 | 7.3 | 13.3 |
| $Q \in V$ | 4.9 | 7.4 | 7.0 | 11.8 | 11.7 | 22.9 | 5.1 | 8.2 | 6.9 | 12.1 | 6.9 | 11.3 |
| $Q = P$ | 4.9 | 7.6 | | | | | | | | | | |
| $Q = B$ | 4.8 | 7.6 | | | | | | | | | | |

*Table 7.1: Efficiency of expansion techniques. The table shows the average query time over 100,000 queries on a machine with a large amount of main memory (Lrg) and one with little (Ltl). The expansion method denoted as* None *specifies the respective baselines,* QE *shows the standard local analysis results, and $S = 40$ shows the results for a summarisation technique. $Q = D$ is a document expansion technique where documents are expanded by treating them as queries at indexing time (*document centric expansion*). $Q \in V$ (or* term centric expansion*), $Q = P$, and $Q = B$ are document expansion techniques where terms, phrases, or bigrams respectively are added to documents.*

### 7.3.2 Timings

For timings, we used 100,000 stopped queries taken from two query logs from the Excite search engine (Spink et al., 2002). Although these queries are web queries and not ideally suited to match the newswire data (we were not able to obtain a more suitable query log), these queries are adequate for testing the throughput – rather than effectiveness – of the system. We only used queries that do not contain phrases.

Timing experiments were run on two machines. The first is an Intel Pentium IV 2.8 GHz machine with hyper-threading and 2 GB of main memory, running GNU/Linux kernel 2.6. The second is a dual Intel Pentium III 866 MHz with 768 MB of main memory, running kernel 2.2. In Table 7.1 these are denoted as *Lrg* and *Ltl*, respectively.

*Lrg* has ample amount of memory that easily fits – at least for the experiments with small collections – the whole document collection as well as the inverted indexes and any major auxiliary data structures, such as document summaries, if applicable. Even though the main memory was flushed before timings were commenced, thus eliminating any influence of caching from any previous timed runs, as all 100,000 queries are processed, all data is eventually cached. The effect of this to the conventional query expansion method is that the additional time requirement over the baseline is purely that of parsing documents, evaluating terms and processing a greater number of inverted lists, rather than the main cost associated with expanding queries from a local set in a typical environment, which is retrieving documents from disk. In practice, for larger collections, this scenario is unrealistic. We therefore also show timings for a machine that can fit only part of the document collection and inverted lists in main memory (*Ltl*).

## 7.4 Results

Results for timings are shown in Table 7.1 and effectiveness performance is shown in Table 7.2. We specify mean average precision (MAP), precision at 10, and R-Precision values (see also Section 2.6.3) for the baseline and various expansion measures. Only methods that were successful on our training data are reported. The $S = 40$ rows in Tables 7.1 and 7.2 give results for one of the most successful methods we previously explored when using in-memory document summaries in Chapter 6. For this method each document is summarised by the top $40$ *tf.idf* terms of that document. During query time, the summaries for all documents are kept in memory (although we did not prevent the operating system from swapping sections of this data structure out of memory). The parameter of $S = 40$ was not tuned for WSJ2 or any other collection. The memory overheads for this method are as follows: WSJ2: 11.7 MB, AP: 26.1 MB, NW: 84.0 MB, FBIS: 20.8 MB, FT: 33.2 MB, and LA: 20.4 MB.

Since the average and R-Precision figures for the experiments with phrases are no better than the other document expansion runs, while resource requirements for phrase experiments are significantly greater than the other techniques, we did not experiment with phrases further.

### 7.4.1 Tuning of document expansion approaches

From the discussion of our document expansion methods in Sections 7.2.1 and 7.2.2, it can be seen that both the document-centric and term-centric approaches involve several parameters. We now discuss how these parameters should be set.

#### Document centric expansion

As for our other experiments, we used the Okapi BM25 similarity function for the first document expansion scheme, where whole documents are run as queries. As explained in Section 2.4.2, generally we set the Okapi parameter $k_3$ to a value of $0$, meaning that each query term contributes only once to the ranking of documents, even though the same term might appear multiple times in a query. This is reasonable for short queries such as those that might be expected in a web search environment. The document expansion scheme, however, adds a whole document to the query; it is therefore likely that many terms appear multiple times. It stands to reason that the expansion technique would be improved if $k_3$ was set to a large value for this experiment, so that terms can contribute more to the similarity calculation if they appear more frequently. A large value of $k_3$ would give those terms that are rare across the document collection as a whole, but appear frequently in the document at hand (that is, those terms with a high

| **Coll.** | **Exp. method** | | **MAP** | **P@10** | **R-Pr.** | | **MAP** | **P@10** | **R-Pr.** | | **MAP** | **P@10** | **R-Pr.** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **WSJ2** | None | **TREC 3** | 0.251 | 0.363 | 0.275 | | | | | | | | |
| | QE | | 0.325‡ | 0.388 | 0.324‡ | | | | | | | | |
| | $S = 40$ | | 0.286‡ | 0.380 | 0.287† | | | | | | | | |
| | $Q = D$ | | 0.265‡ | 0.361 | 0.280 | | | | | | | | |
| | $Q \in V$ | | 0.264 | 0.378 | 0.283 | | | | | | | | |
| | $Q = P$ | | 0.259 | 0.371 | 0.276 | | | | | | | | |
| | $Q = B$ | | 0.260 | 0.380 | 0.268† | | | | | | | | |
| **AP** | None | **TREC 3** | 0.243 | 0.430 | 0.262 | | | | | | | | |
| | QE | | 0.327‡ | 0.468‡ | 0.333‡ | | | | | | | | |
| | $S = 40$ | | 0.290‡ | 0.454‡ | 0.301‡ | | | | | | | | |
| | $Q = D$ | | 0.251 | 0.416 | 0.286‡ | | | | | | | | |
| | $Q \in V$ | | 0.248 | 0.420 | 0.276‡ | | | | | | | | |
| **NW** | None | | | | | **TREC 7** | 0.195 | 0.458 | 0.251 | **TREC 8** | 0.222 | 0.438 | 0.262 |
| | QE | | | | | | 0.232‡ | 0.452 | 0.285‡ | | 0.250‡ | 0.464 | 0.289‡ |
| | $S = 40$ | | | | | | 0.208 | 0.438 | 0.263 | | 0.234 | 0.434 | 0.269 |
| | $Q = D$ | | | | | | 0.199 | 0.476 | 0.259 | | 0.213 | 0.444 | 0.263 |
| | $Q \in V$ | | | | | | 0.195† | 0.444 | 0.243 | | 0.220 | 0.434 | 0.261 |
| **FBIS** | None | **TREC 6** | 0.223 | 0.260 | 0.232 | **TREC 7** | 0.208 | 0.318 | 0.218 | **TREC 8** | 0.269 | 0.319 | 0.281 |
| | QE | | 0.237‡ | 0.266 | 0.226 | | 0.222 | 0.292† | 0.243‡ | | 0.270 | 0.305 | 0.256 |
| | $S = 40$ | | 0.231 | 0.274 | 0.226 | | 0.217 | 0.308 | 0.224 | | 0.268 | 0.309 | 0.274 |
| | $Q = D$ | | 0.220‡ | 0.257 | 0.235 | | 0.205 | 0.300‡ | 0.218 | | 0.264‡ | 0.312 | 0.279 |
| | $Q \in V$ | | 0.233 | 0.260 | 0.237 | | 0.228 | 0.318 | 0.239‡ | | 0.278 | 0.321 | 0.284 |
| **FT** | None | **TREC 6** | 0.214 | 0.250 | 0.244 | **TREC 7** | 0.224 | 0.271 | 0.241 | **TREC 8** | 0.290 | 0.331 | 0.298 |
| | QE | | 0.209 | 0.261 | 0.220 | | 0.233 | 0.287 | 0.234 | | 0.298 | 0.361† | 0.282 |
| | $S = 40$ | | 0.217 | 0.276‡ | 0.221 | | 0.216 | 0.269 | 0.229 | | 0.261 | 0.341 | 0.249 |
| | $Q = D$ | | 0.211† | 0.243 | 0.235 | | 0.229 | 0.277 | 0.242 | | 0.299 | 0.316 | 0.312 |
| | $Q \in V$ | | 0.206 | 0.237 | 0.229 | | 0.212 | 0.287‡ | 0.221 | | 0.295 | 0.325 | 0.304 |
| **LA** | None | **TREC 6** | 0.198 | 0.231 | 0.232 | **TREC 7** | 0.211 | 0.300 | 0.234 | **TREC 8** | 0.233 | 0.260 | 0.238 |
| | QE | | 0.226‡ | 0.254‡ | 0.218 | | 0.251‡ | 0.316 | 0.269† | | 0.207 | 0.256 | 0.223 |
| | $S = 40$ | | 0.213‡ | 0.244‡ | 0.222 | | 0.240† | 0.306 | 0.263† | | 0.216 | 0.262 | 0.237 |
| | $Q = D$ | | 0.209 | 0.227 | 0.221 | | 0.225 | 0.304 | 0.242 | | 0.237 | 0.256 | 0.248 |
| | $Q \in V$ | | 0.216 | 0.237† | 0.237 | | 0.224 | 0.288 | 0.250 | | 0.235 | 0.256 | 0.242 |

*Table 7.2: Effectiveness of expansion techniques. The WSJ2 data was used for tuning. Effectiveness results are averaged over 50 queries. Shown are mean average precision (MAP), precision at 10 (P@10), and R-Precision (R-Pr.). The expansion method denoted as* None *specifies the respective baselines,* QE *shows the local analysis results and* $S = 40$ *shows the results for a summarisation technique.* $Q = D$ *is a document expansion technique where documents are expanded by treating them as queries at indexing time (*document centric expansion*).* $Q \in V$ *(or* term centric expansion*),* $Q = P$ *and* $Q = B$ *are document expansion techniques where terms, phrases or bigrams, respectively, are added to documents. Results that are statistically significant different to the baseline at the 0.10 and 0.05 levels are marked with † and ‡ respectively.*

*tf.idf* value), a large impact on the ranking of documents for the local set. Arguably (and local analysis is based on this assumption), those terms are most defining the topic of the document.

However, we found in preliminary experiments, not reported here, that using a large value for $k_3$ or, alternatively, using the the vector space model (see Section 2.4.1) chosen with the same consideration, degrades results considerably.

Furthermore, using our training data, we found that selecting terms with the *KLD* (Kullback-Leibler divergence measure, discussed in Section 3.3.2) worked consistently better than selecting terms based on their *tf.idf* value or *TSV* (see Section 3.3.2). Interestingly, it became clear in initial experiments that allowing terms which are already contained in a document to be appended to this document, decreases effectiveness compared to restricting additions to new terms. We also found that augmenting a document with around $10\%$ of the number of tokens in a document works best, rather than adding a fixed number of terms or using a global threshold value for the selection value of each candidate term. That is, a document that contains $100$ words is augmented with $10$ more words. As mentioned earlier, a side effect of document expansion is therefore that document collections and associated indexes are roughly $10\%$ longer than the original collection and indexes after expansion.

A potential problem with document expansion is that terms that are used for augmenting documents tend to be quite rare across the collection. Adding rare terms to documents means that, after expansion, those terms will be less rare, which will have an effect on retrieval behaviour. An analysis of this effect of “diluting” collection frequencies is left as future work.

#### Term centric expansion

In our experiments, we rank $100$ documents against an expanded term, although we found that ranking any number between $90$ and up to $110$ documents works equally well. Contrary to the document centric expansion method, we found that using the *TSV* to select terms worked better than choosing candidate terms based on their *KLD* values, possibly because this more closely mimics the local analysis algorithm. We allowed terms to be added to documents even though they might already appear in that document. Surprisingly, we found that excluding terms on this basis leads to lower increases in effectiveness, unlike with the document centric expansion method.

### 7.4.2 Effectiveness

In the following discussion we treat any sub-collection as a full collection and neglect a change of 0.005 or less in the respective effectiveness measurements. Out of 13 collections, the standard approach to query expansion lead to an increase in MAP for ten collections, while decreasing it for two. The document-centric expansion technique ($Q = D$) improved MAP for five collections and degraded the results of one. For the term-centric document expansion method ($Q \in V$), these figures are six and two respectively. Query expansion improved precision-at-10 in nine instances and decreased it in three cases. Using either document expansion technique increased P@10 three times and decreased it five times. Ignoring for a moment the magnitude of changes in effectiveness and using only those terms as outlined above for comparison, the document summarisation technique $S = 40$ performs the same as query expansion, with the exception of the FT collection where it is mostly worse than query expansion.

Increases in effectiveness for document expansion methods are typically very small compared to those of query expansion. Furthermore, improvements achieved by query expansion are mostly statistically significant whereas document expansion improvements are not.

### 7.4.3 Efficiency

The term-centric document expansion technique ($Q \in V$) slows down retrieval by around only 2% in most cases, but the document-centric method ($Q = D$) adds roughly 10% to retrieval cost. These figures do not vary from one machine to the other, as there is enough main memory on either machine for all inverted lists to be cached.

On the machine with large amounts of memory (*Lrg*), query expansion slows retrieval down by a factor of five to seven. Caching does not work well at all, since from one query to the next many inverted lists have to be purged in order to load new lists and to make room for documents to be fetched from disk. This problem is exacerbated on the machine with less main memory, where the overhead increases from five to fifteen-fold, depending on the size of the collection.

For *Lrg*, the additional data needed for the technique involving summaries ($S = 40$) fits well into memory, while leaving adequate room for inverted lists to be cached. This is why query times are increased only by around 50%. On *Ltl*, the in-memory summaries need to be swapped in and out of memory more often and the penalty is relatively high, leading to a decrease in query throughput to roughly half that of the baseline.

### 7.4.4 Robustness

Figures 7.3 and 7.4 show the robustness of different retrieval techniques, illustrating how many queries are degraded or improved in respect to the baseline, and by how much. The baseline is constructed by running queries in their original form against the non-modified collection. All lines intersect the x-axis at roughly the same point, which means that all methods examined in this chapter exhibit similar robustness for each collection.

## 7.5 Analysis

A possible explanation for the relatively poor improvements of document expansion is that – as for query drift (see Page 45) – the topic of the expanded documents is changed too much from the original topic by

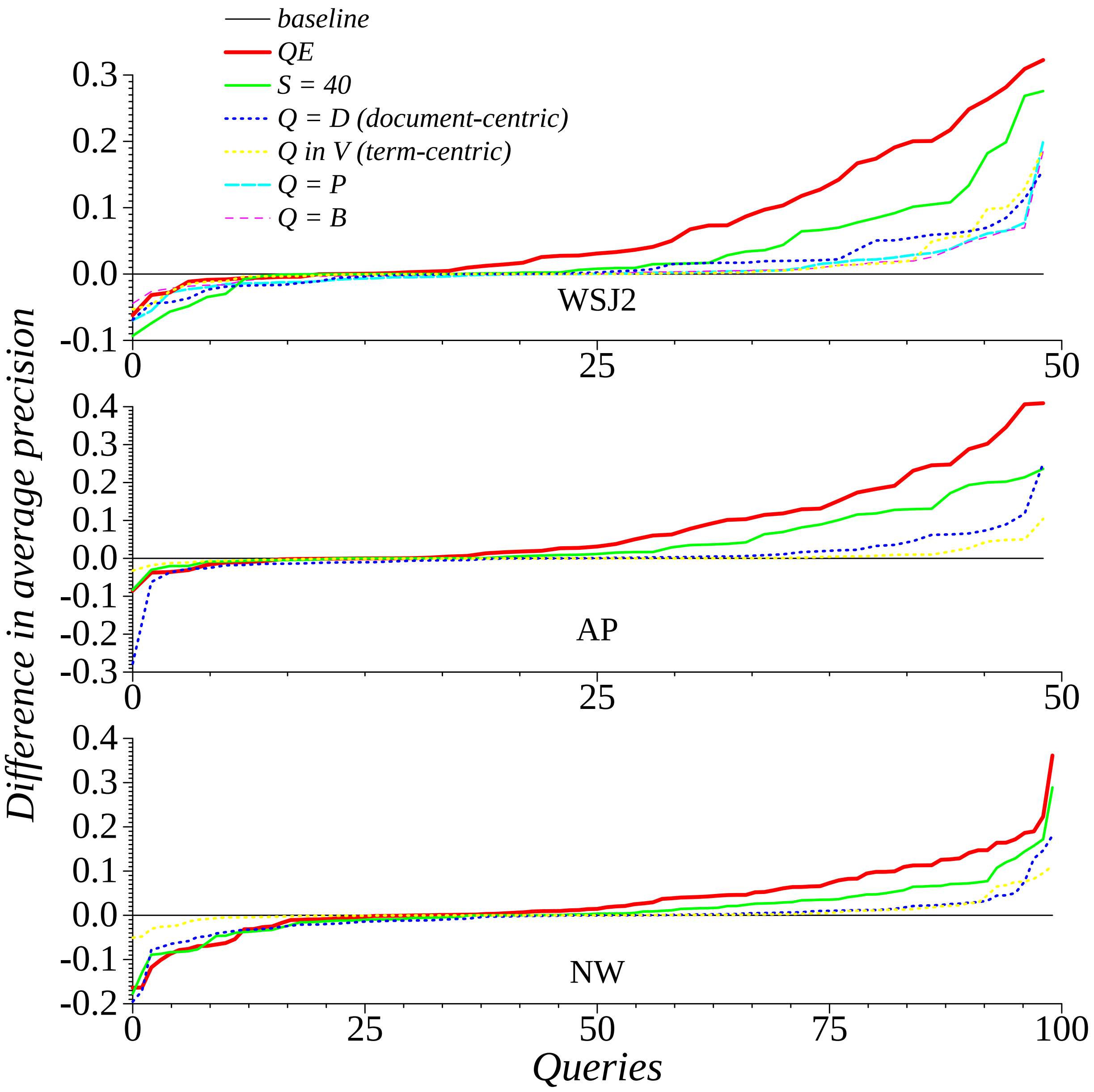


*Figure 7.3: Per-query-differences in average precision between each of the methods and the respective baselines for the WSJ2, AP and NW data. The same notation is used as in Table 7.2. Each curve is sorted individually by the magnitude of the difference in average precision. Data for phrases and bigrams is only shown for the WSJ2 data.*

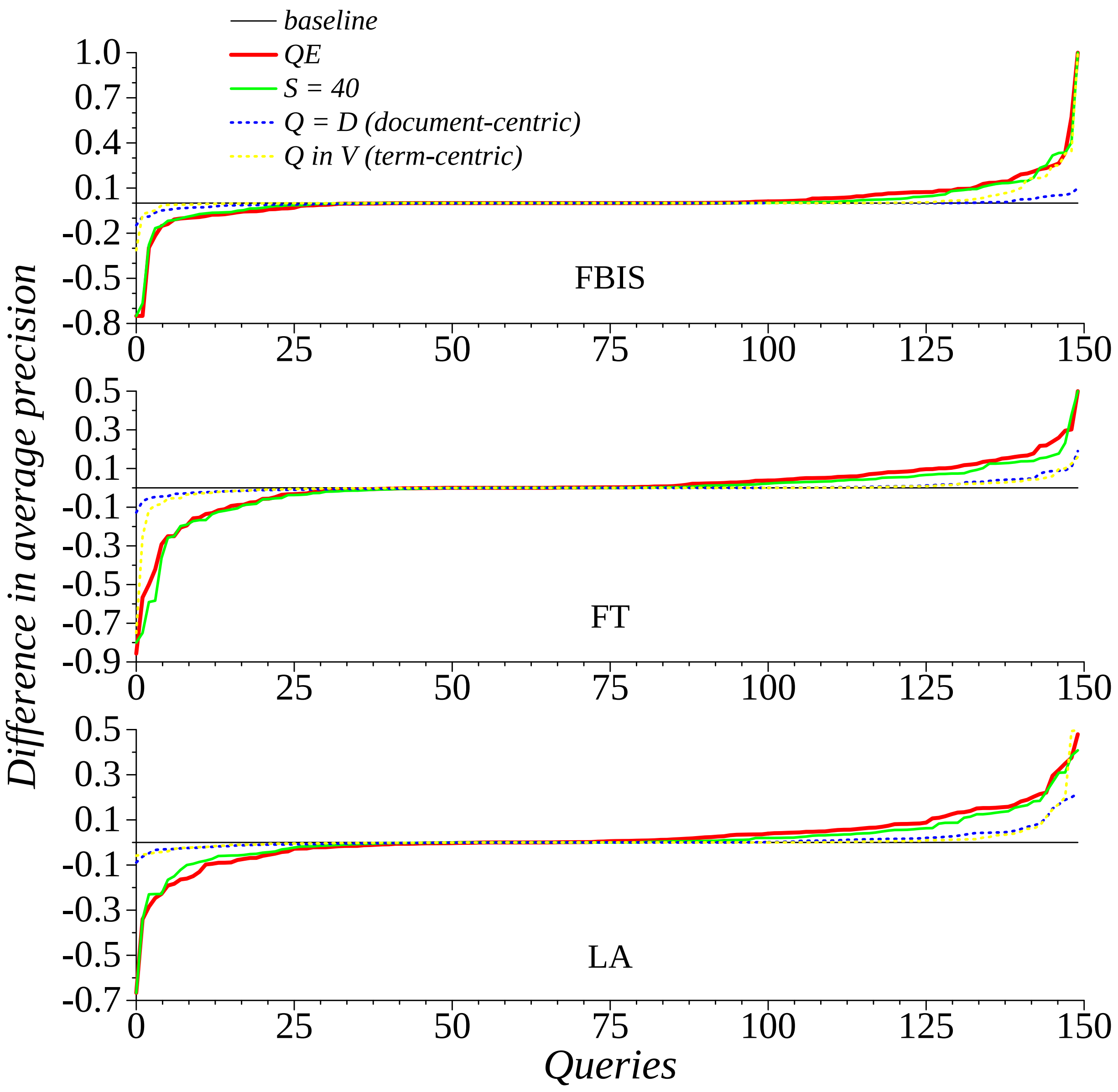


*Figure 7.4: As for Figure 7.3, except effectiveness is shown for FBIS, FT and LA data.*

terms that are not sufficiently centred on the original topic. This problem could be overcome by reducing the weight of document expansion terms, which would lead to a reduced impact of the appended terms. In our implementation we have not modified weights of terms that are appended to documents and leave this for future work.

A further explanation, and a more substantial one, could be that the lack of context during the expansion of documents is unhelpful; whereas, during conventional local analysis, several query terms (usually more than one) set a particular context that determines the intersection of documents in the local set. No such context is available during index-time. The only information available during indexing time (in absence of a very large query log that is suitable for the collections at hand) is – depending on the expansion method chosen – either a single vocabulary term or a document. This means that the term might have been added to documents that contain other, related, terms for the term-centric approach. For the document-centric approach more terms for a particular document may have been found that might be suitable for that document. In neither case, though, were these additional terms helpful in making this document more retrievable for relevant queries.

Our experiments involving phrases try to address this problem of a missing context; however, the generation method of phrases is most likely insufficient. Additionally, phrases are extracted from the collection at hand (rather than being sourced from a suitable query log for example) and therefore no new context from outside the collection is found.

## 7.6 Summary

A series of experiments cannot prove that a family of methods is not viable. Establishing a positive result is straightforward; establishing a negative result involves demonstrating that all reasonable avenues of progress have been investigated and found wanting. Nonetheless, we believe we have found that corpus-based document expansion is not promising. Other document expansion methods, based on extracting terms from external resources, have been found to give limited gains in some circumstances. However, while query-time costs are low, we were unable to use corpus-based document expansion to significantly improve effectiveness, and the index-time costs are considerable.

In contrast, our fresh investigation of query expansion showed that it was generally of benefit in the newswire collections used in our experiments, and that the evaluation costs can be much reduced while broadly maintaining the effectiveness gains. These results, we believe, should help focus future research in the area, by demonstrating that work on document expansion may not be warranted and by suggesting promising further directions for improving the efficiency and effectiveness of query expansion.

# Chapter 8

# Conclusions and Future Work

In this chapter we summarise our results and draw conclusions from our experiments. We also outline future research that the work in this thesis may lead to.

## 8.1 Robustness of local analysis query expansion

Although query expansion is a successful method for improving the average effectiveness of an information retrieval system, we found in Chapter 4 that blind relevance feedback can degrade results for some queries. In particular, we found that – depending on the collection and query set – roughly a quarter of queries are degraded, whereas for another quarter effectiveness is increased. For those queries, differences in average precision of up to 50% in absolute terms can be observed. The effectiveness of the bulk of queries is little changed. In effect, query expansion is less reliable than suggested previously in much of the relevant literature.

We examined the performance of a standard approach to local analysis query expansion by exploring the impact of two key parameters on the effectiveness of this technique. One of the two parameters we examined was the number of documents ranked against the initial query. This set of documents is assumed to be relevant to the query, and subsequently used as the local set from which expansion terms are drawn. The second parameter we examined was the number of terms added to the initial query. While varying these two parameters, we observed the following outcomes:

- Using the parameters that are suggested as default in the research literature (Robertson and Walker, 1999) on average improves the effectiveness of query expansion. In the case of the TREC 8 collection and query set, MAP increased from 21.6% to 25.4%.

- Tuning these parameters for a particular collection and query set can further increase these results, in the case of TREC 8, to 26.0%.

- The optimal parameters vary from query to query. Whereas one query is best improved with a low number of terms drawn from a small local set, another might be best expanded using a larger number of terms that have been retrieved from a substantial number of documents. Yet other queries are improved best with a single expansion term that is sourced from a large local set, or vice versa.
- Whereas some queries can be expanded successfully using the optimal set of parameters (for instance, average precision for TREC query 405 was improved to 31.7% from a baseline of 6.1%), for others even the best combination of parameters only leads to a marginal improvement (such as query 440, where average precision increased from 9.3% to 12.4%).
- The optimal choice of parameters varies between collections. For TREC 8, the best combination of parameters was to choose 15 terms from 13 documents, whereas for TREC 9 optimality was achieved by drawing 4 terms from 98 documents.
- An approach that uses a fixed set of parameters for all queries can significantly be improved upon, by using an optimal set of parameters for each individual query. For TREC 8, using parameters that are individually tuned for each query, improves MAP to 33.0%. This suggests an upper bound for performance of query expansion for the query set.
- What is not clear is how the parameters should be chosen. In our experiments, the optimal parameters were found using post-hoc tuning. Although we explored a range of options, we have not identified a metric that provides a method for guiding expansion.

Nonetheless, with appropriate parameter choices local analysis query expansion is a successful way of enhancing the effectiveness of queries, particularly on collections with typically consistent documents.

As discussed above, our results show that the performance of query expansion has significant scope for improvement: individually tuning parameters to queries can give much better performance than using fixed values. In future, we hope to be able to develop a method for predicting parameter values, and thus obtain greater effectiveness than is available with current methods.

In addition to this, we would like to investigate how many of the added terms have an impact on the effectiveness of a query. In preliminary experiments (not detailed in this thesis) we found that often the most dramatic change in effectiveness (whether this be positive or negative) is due to the addition of a single term. An interesting question is then whether these terms are likely to have some common feature, such as having a particularly high or low *tf.idf* value.

Furthermore, with the existing method, terms are weighted using the Robertson/Spark Jones relevance weight, where the number of terms added to the query and their assigned weight is independent of the weight of the initial query terms. It might be interesting to see whether query drift can be prevented by for instance distributing a combined weight over the expansion terms that is not greater than the combined weight of the initial query terms.

## 8.2 Use of past queries for query expansion

Although local analysis query expansion on average increases the effectiveness of queries, web queries – which are used in an environment where the language and vocabulary is less controlled than for instance a newswire collection – are often not well expanded and the use of expansion may lead to lower average effectiveness.

In Chapter 5 we explored a method that does allow web queries in particular to be expanded well. We make use of query association, where past user queries (which may be harvested from a query log) are associated with documents that are ranked highest in response to each query. For each web document a surrogate document is provided, which contains all associated queries.

These surrogates have the benefit that they consist of a more controlled vocabulary, as each query has been composed by users – at least indirectly – picking terms that best described the document at hand. Since we placed the restriction on associations that a query can only be associated with a document if the document contains all query terms, the surrogate can in effect be a concise and carefully worded summary of the document.

When expanding a query, instead of retrieving candidate terms from the web documents themselves, terms are sourced from the surrogates. We used almost one million past queries for associations in our experiments. We found that this method of expanding queries yields a relative improvement of 26%–29% over an unexpanded baseline, or an 18%–20% relative improvement over the standard approach to local analysis, where expansion parameters have been optimised for the baseline. The results achieved were found to be statistically significant. We conclude that query associations can be a powerful expansion method for web retrieval.

However, no improvements were found when we used a collection of individual past queries as expansion term sources, as those are too sparse to contain good expansion terms that are not in the initial query to begin with. Similarly, the use of surrogates consisting of anchor text instead of past queries was unsuccessful, in this case presumably because the test queries we used were topic-finding queries for which anchor text is not well suited.

We plan to pursue several directions in our future work. We will investigate the optimal parameters for query association in the context of query expansion; the work in Chapter 5 uses parameters that were determined through a surrogate retrieval task. We also plan to investigate whether fixed parameters for query association are appropriate, and whether all queries should be associated to documents. Furthermore, we are interested to evaluate the efficiency tradeoff between maintaining query associations in a live system, and being able to use associations for expansion (rather than having to rely on the inferior conventional approach).

## 8.3 Efficient query expansion using in-memory document summaries

We examined the costs involved in local analysis and made suggestions for reducing these costs in Chapter 6. In particular, there are five stages to local analysis: ranking an initial set of documents against the original query, retrieving these documents, extracting candidate terms from the local set of documents, selecting expansion terms, and ranking the final set of results.

The two ranking stages can be addressed with conventional approaches to more efficient query evaluation (see Section 2.4.4). The term selection phase is of low cost, and attempts to reduce the time to select candidate terms can only yield limited benefits. However, we identified the need for retrieving documents from disk as the main bottleneck of local analysis query expansion, in combination with the need to parse those documents.

In order to eliminate the time penalty for retrieving documents from disk, we store brief summaries for each document in main memory. The summaries consist of the most pertinent terms of each document, as identified by a *tf.idf* measure. We proposed three ways of determining the summary size: one is to take a constant number of terms from each document, the second is based on a threshold of the *tf.idf* value of summary terms, and the last uses a percentage of unique terms encountered in each document.

We found that summary sizes of between 20 and 70 terms deliver the most effective expansion results. While the amount of memory needed for the summaries is considerable (up to 150 Mb for a 2 Gb collection and 250 Mb for a 10 Gb collection), it is well within the capabilities of modern desktop machines.

Overall, the summaries are as effective as the conventional approach to local analysis, while reducing expansion time by a factor of five to nine. This increase in efficiency is due to several factors:

- The time needed to access in-memory summaries is negligible, when compared to retrieving documents. Using the conventional approach, due to the low probability of the need to access an already cached document, most documents need to be retrieved from disk.
- The time taken to extract terms is reduced as summaries consist of pre-selected terms only; no parsing or stopping is needed.
- Although term selection is only a minor contributor to the total cost of local analysis expansion, it is still reduced heavily in comparison through the use of in-memory summaries, as only a very small number of terms need to be considered for selection.
- Surprisingly, the stages that were impacted the most by the use of in-memory summaries are the ranking phases. While we expected that the large usage of memory for the summaries would have a negative effect on caching of inverted lists and thereby increasing the time needed to rank queries, the opposite was true. Because of the use of summaries, the erratic loading of documents, which caused inverted lists to be purged, is no longer necessary. This improved the caching of inverted lists, and ranking times are reduced by between 10% and 20%.

We also considered two other approaches to speed up expansion. One is to make a sub-collection available for the initial ranking and extraction of terms. The motivation for this approach is that the sub-collection should be representative of the whole collection and expansion terms should be similarly useful, since inverted lists are short, the initial ranking should be fast, while at the same time the likelihood of caching documents that are used in the expansion process is increased. The second approach is to use query associations (described earlier) in order to have pre-parsed, smaller document surrogates available, again with the aim of improving caching. In terms of efficiency, neither of these two approaches proved to be successful. For either strategy a second index needs to be used for ranking, which eliminates the expected benefit of improved caching.

As mentioned earlier, more efficient query evaluation schemes can have a marked impact on the overall efficiency of query expansion. As future work, it would be interesting to see whether our approach of using in-memory summaries is comparatively efficient when a more efficient evaluation scheme – such as impact ordering (Anh and Moffat, 2004) – is used.

Furthermore, we could use a hybrid approach of our in-memory summaries and using a sub-collection of documents by keeping only summaries of selected documents (for instance chosen by frequency of access (Garcia et al., 2004)) in memory. This would reduce the overall memory requirements and should increase the likelihood that needed inverted lists are cached. In addition to this, a second tier of summaries could be stored on disk, for retrieval in cases where not many of the highly-ranked documents are among the pre-selected set.

Finally, we would expect further improvements to any strategy by compressing the in-memory surrogates.

## 8.4 Use of document expansion instead of query expansion

We detailed approaches to document expansion for text retrieval in Chapter 7. Although document expansion has been used previously for other tasks, such as speech retrieval (Singhal and Pereira, 1999), it had not been used for automatic query expansion.

We suggested two techniques for document expansion where documents are increased by up to 10% of their original size. The *document centric* approach treats each document in turn as a query that is subsequently expanded. The identified expansion terms are added to the document. The *term centric* approach issues each vocabulary term as a query. This query is expanded and the term is appended to those documents that were ranked with the expanded query. We also investigated alternatives to the second approach whereby phrases, which were previously identified from the document collection at hand, are used as queries.

Using either technique, documents are expanded at considerable cost during indexing time. However, as no expansion of queries is performed during querying, querying time is only minimally increased, due to the slightly longer inverted lists. The document centric approach slowed query evaluation by between 10% and 15%, while the term centric approach had a much smaller impact (roughly a 2% increase in query time).

Although document expansion has been shown to be not as useful as query expansion, it has been shown to be more effective than no expansion in some cases. Therefore document expansion might be worthwhile to consider in contexts where query expansion is impractical. In this case, future research into document expansion could focus on more efficient approaches of expanding documents during indexing time. Document expansion could be considerably sped up by, for instance, using only a certain number of highly valued *tf.idf* terms instead of the whole document when constructing queries from documents, thus speeding up query evaluation time considerably (far fewer inverted lists have to be read). Another possibility is to use impact order indexes (see Section 2.4.4), where only those terms with the highest impact on the ranking will be evaluated, instead of all document terms.

Our experiments involving phrases were not as robust as other experiments in this thesis, since our approach to extracting phrases from the collection is somewhat naïve. Instead we could extract phrases on a more principled basis, for instance following the technique suggested by Joho and Sanderson (2000) or Liu et al. (2004).

A potential problem with any of the document expansion techniques we used is that – in the main – relatively rare terms are added to documents, since those typically have a high selection value. This has the effect that the number of documents containing these terms is increased considerably in some instances, which distorts the term distribution statistics in the collection, and can therefore have an effect on retrieval performance. This effect could be quantified in future work, and avenues to eliminate or at least reduce these effects could be explored.

A related problem is that the full weight of terms is used during query time, once these terms are appended to documents. That is, there is no distinction between expansion terms and original document terms. In future work, we could reduce the weight of document expansion terms.

However, the effectiveness results are not encouraging. We conducted experiments with multiple query sets and collections. Although more often than not mean average precision is improved through document expansion, the increases are only small and not statistically significant. On the other hand, the standard local analysis approach, and the technique involving in-memory summaries detailed earlier, show good improvements.

An avenue for future work is to use reduced weighting for document expansion terms. Currently terms are appended without their weight being diminished, unlike for the conventional approach, where expansion term weights are downgraded by two thirds.

It is not possible to prove that a family of methods or variants thereof are not viable using a series of experiments. While a positive result can be confirmed; in order to show a negative result, one needs to show that all reasonable avenues lead to bad results. Nonetheless, we believe that corpus-based document expansion is not promising while document expansion based on external resources has been shown to only give limited gains in some experiments.

However, as we compared the document expansion techniques to standard local analysis and the in-memory summary approach, we confirmed with other data sets that the latter two approaches can deliver good effectiveness results. This work also confirmed that expansion based on in-memory-based summaries is promising in terms of both effectiveness and efficiency and is a good avenue for future research into making query expansion more effective.

## 8.5 Summary

We have shown in this thesis that local analysis query expansion is a viable method for improving queries, although there is considerable room for improvement in the choice of parameter settings. We found a novel approach to expanded web queries that have in the past been particularly difficult to expand. Although document expansion as an alternative to query expansion is not promising, we have shown an efficient technique for expanding queries. Using in-memory summaries, the cost of expanding queries can be reduced by a factor of five to nine.

## References

Aalbersberg, I. J. (1992), Incremental relevance feedback, *in* N. J. Belkin, P. Ingwersen and A. M. Pejtersen, eds, “Proc. ACM-SIGIR Int. Conf. on Research and Development in Information Retrieval”, ACM Press, New York, Copenhagen, Denmark, pp. 11–22.

Ahmad, F., Yusoff, M. and Sembok, T. M. T. (1996), “Experiments with a stemming algorithm for Malay words”, *Jour. of the American Society for Information Science* **47**(12), 909–918.

Allen, J. (1995), *Natural Language Understanding*, 2nd edn, Benjamin-Cummings Publishing Co., Inc., Redwood City, CA.

Amati, G., Carpineto, C. and Romano, G. (2004), Query difficulty, robustness, and selective application of query expansion, *in* S. McDonald and J. Tait, eds, “European Conf. on IR Research”, Vol. 2997 of *Lecture Notes in Computer Science*, Springer, pp. 127–137.

Anh, V. N. and Moffat, A. (2002), Impact transformation: Effective and efficient web retrieval, *in* M. Beaulieu, R. Baeza-Yates, S. H. Myaeng and K. Järvelin, eds, “Proc. ACM-SIGIR Int. Conf. on Research and Development in Information Retrieval”, ACM Press, New York, Tampere, Finland, pp. 3–10.

Anh, V. N. and Moffat, A. (2004), Collection-independent document-centric impacts, *in* P. Bruza, A. Moffat and A. Turpin, eds, “Proc. Australian Document Computing Conf.”, University of Melbourne, Department of Computer Science, pp. 25–32.

Asian, J., Williams, H. E. and Tahaghoghi, S. M. M. (2005), Stemming Indonesian, *in* V. Estivill-Castro, ed., “Proc. Australasian Computer Science Conf.”, Australian Computer Society, pp. 307–314.

Baeza-Yates, R. and Ribeiro-Neto, B. (1999), *Modern Information Retrieval*, Addison-Wesley Longman, Boston, MA.

Bahle, D., Williams, H. E. and Zobel, J. (2002), Efficient phrase querying with an auxiliary index, *in* M. Beaulieu, R. Baeza-Yates, S. H. Myaeng and K. Järvelin, eds, “Proc. ACM-SIGIR Int. Conf. on Research and Development in Information Retrieval”, ACM Press, New York, Tampere, Finland, pp. 215–221.

Bailey, P., Craswell, N. and Hawking, D. (2003), “Engineering a multi-purpose test collection for web retrieval experiments”, *Information Processing & Management* **39**(6), 853–871.

Belkin, N. J., Cabezas, A., Cool, C., Kim, K., Ng, K. B., Park, S., Pressman, R. and Rieh, S. (1996), Rutgers interactive track at TREC-5, *in* E. M. Voorhees and D. K. Harman, eds, “Proc. Text Retrieval Conf. (TREC)", National Institute of Standards and Technology Special Publication 500-238, Gaithersburg, MD, pp. 257–266.

Belkin, N. J., Cool, C., Koenemann, J., Ng, K. B. and Park, S. (1995), Using relevance feedback and ranking in interactive searching, *in* D. K. Harman, ed., "Proc. Text Retrieval Conf. (TREC)", National Institute of Standards and Technology Special Publication 500-236, Gaithersburg, MD, pp. 181–210.

Belkin, N. J., Oddy, R. N. and Brooks, H. M. (1993), "Ask for information retrieval: Parts I and II", *Jour. of Documentation* **38**(2 & 3), 61–71,145–164.

Bell, T. C., Moffat, A., Witten, I. H. and Zobel, J. (1995), "The MG retrieval system: Compressing for space and speed", *Communications of the ACM* **38**(4), 41–42.

Berger, A. and Lafferty, J. D. (1999), Information retrieval as statistical translation, *in* F. Gey, M. Hearst and R. Tong, eds, "Proc. ACM-SIGIR Int. Conf. on Research and Development in Information Retrieval", ACM Press, New York, Berkeley, CA, pp. 222–229.

Bernstein, Y., Billerbeck, B., Garcia, S., Lester, N., Scholer, F., Zobel, J. and Webber, W. (2005), RMIT University at TREC 2005: Terabyte and robust track, *in* E. M. Voorhees and L. P. Buckland, eds, "Proc. Text Retrieval Conf. (TREC)", National Institute of Standards and Technology, Gaithersburg, MD.

Berry, M. W., Dumais, S. T. and O'Brien, G. W. (1995), "Using linear algebra for intelligent information retrieval", *SIAM Review* **37**(4), 573–595.

Bharat, K. and Henzinger, M. R. (1998), Improved algorithms for topic distillation in a hyperlinked environment, *in* W. B. Croft, A. Moffat, C. J. van Rijsbergen, R. Wilkinson and J. Zobel, eds, "Proc. ACM-SIGIR Int. Conf. on Research and Development in Information Retrieval", ACM Press, New York, Melbourne, Australia, pp. 104–111.

Billerbeck, B., Cannane, A., Chatteraj, A., Lester, N., Webber, W., Williams, H. E., Yiannis, J. and Zobel, J. (2004), RMIT University at TREC 2004, *in* E. M. Voorhees and L. P. Buckland, eds, "Proc. Text Retrieval Conf. (TREC)", National Institute of Standards and Technology Special Publication 500-261, Gaithersburg, MD.

Billerbeck, B., Scholer, F., Williams, H. E. and Zobel, J. (2003), Query expansion using associated queries, *in* D. Kraft, O. Frieder, J. Hammer, S. Qureshi and L. Seligman, eds, "Proc. Int. Conf. on Information and Knowledge Management", ACM Press, New York, New Orleans, LA, pp. 2–9.

Billerbeck, B. and Zobel, J. (2003), When query expansion fails, *in* J. Callan, G. Cormack, C. Clarke, D. Hawking and A. Smeaton, eds, "Proc. ACM-SIGIR Int. Conf. on Research and Development in Information Retrieval", ACM Press, New York, Toronto, Canada, pp. 387–388.

Billerbeck, B. and Zobel, J. (2004*a*), Questioning query expansion: An examination of behaviour and parameters, *in* K.-D. Schewe and H. E. Williams, eds, “Proc. Australasian Database Conf.”, Vol. 27, Australian Computer Society, Dunedin, New Zealand, pp. 69–76.

Billerbeck, B. and Zobel, J. (2004*b*), Techniques for efficient query expansion, *in* A. Apostolico and M. Melucci, eds, “Proc. String Processing and Information Retrieval Symp.”, Springer-Verlag, Padova, Italy, pp. 30–42.

Billerbeck, B. and Zobel, J. (2005), Document expansion versus query expansion for ad-hoc retrieval, *in* A. Turpin and R. Wilkinson, eds, “Proc. of the Tenth Australasian Document Computing Symposium”, Sydney, Australia, pp. 34–41.

Billerbeck, B. and Zobel, J. (to appear), “Efficient query expansion with auxiliary data structures”, *Information Systems* . Special issue.

Bilmes, J. A. (1998), A gentle tutorial of the EM algorithm and its application to parameter estimation for Gaussian mixture and hidden Markov models, Technical report, International Computer Science Institute and Computer Science Division University of California, Berkeley. tr-97-021.

Bookstein, A. (1982), Explanation and generalization of vector models in information retrieval, *in* G. Salton and H.-J. Schneider, eds, “Proc. ACM-SIGIR Int. Conf. on Research and Development in Information Retrieval”, Springer-Verlag, West Berlin, Germany, pp. 118–132.

Brajnik, G., Mizzaro, S. and Tasso, C. (1996), Evaluating user interfaces to information retrieval systems: a case study on user support, *in* H.-P. Frei, D. Harman, P. Schäubie and R. Wilkinson, eds, “Proc. ACM-SIGIR Int. Conf. on Research and Development in Information Retrieval”, ACM Press, New York, Zürich, Switzerland, pp. 128–136.

Brin, S. and Page, L. (1998), The anatomy of a large-scale hypertextual web search engine, *in* P. H. Enslow and A. Ellis, eds, “Proc. World-Wide Web Conference”, Elsevier, Brisbane, Australia, pp. 107–117.

Broder, A. (2002), “A taxonomy of web search”, *SIGIR Forum* **36**(2), 3–10.

Broglio, J., Callan, J. P., Croft, W. B. and Nachbar, D. W. (1994), Document retrieval and routing using the INQUERY system, *in* D. K. Harman, ed., “Proc. Text Retrieval Conf. (TREC)”, National Institute of Standards and Technology Special Publication 500-225, Gaithersburg, MD, pp. 22–29.

Buckley, C. (2004), Why current IR engines fail, *in* M. Sanderson, K. Järveln, J. Allan and P. Bruza, eds, “Proc. ACM-SIGIR Int. Conf. on Research and Development in Information Retrieval”, ACM Press, New York, Sheffield, United Kingdom, pp. 584–585.

Buckley, C., Allan, J. and Salton, G. (1995), “Automatic routing and retrieval using Smart: TREC-2”, *Information Processing & Management* **31**(3), 315–326.

Buckley, C. and Harman, D. K. (2004), Reliable information access final workshop report, Technical report, Advanced Research and Development Activity (ARDA) Northeast Regional Research Center.

Buckley, C. and Salton, G. (1995), Optimization of relevance feedback weights, *in* E. A. Fox, P. Ingwersen and R. Fidel, eds, “Proc. ACM-SIGIR Int. Conf. on Research and Development in Information Retrieval”, ACM Press, New York, Seattle, WA, pp. 351–357.

Buckley, C., Salton, G., Allan, J. and Singhal, A. (1994), Automatic query expansion using SMART: TREC 3, *in* D. K. Harman, ed., “Proc. Text Retrieval Conf. (TREC)”, National Institute of Standards and Technology Special Publication 500-225, Gaithersburg, MD, pp. 69–80.

Campbell, I. (1995), Supporting information needs by ostensive definition in an adaptive information space, *in* I. Ruthven, ed., “Proc. of the MIRO ’95, Electronic Workshops in Computing”, Springer-Verlag, Glassgow, United Kingdom, pp. 1–25.

Carpineto, C., de Mori, R., Romano, G. and Bigi, B. (2001), “An information-theoretic approach to automatic query expansion”, *ACM Transactions on Information Systems* **19**(1), 1–27.

Chang, Y. K., Cirillo, C. and Razon, J. (1971), Evaluation of feedback retrieval using modified freezing, residual collection, and test and control groups, *in* G. Salton, ed., “The SMART Retrieval System – Experiments in Automatic Document Processing”, Prentice-Hall, Englewood Cliffs, NJ, pp. 355–370.

Chen, S. F. and Goodman, J. (1996), An empirical study of smoothing techniques for language modeling, *in* “Proc. of the 34th Conf. on Association for Computational Linguistics”, Association for Computational Linguistics, Santa Cruz, CA, pp. 310–318.

Clarke, C. L. A., Cormack, G. V. and Lynam, T. R. (2001), Exploiting redundancy in question answering, *in* D. H. Kraft, W. B. Croft, D. J. Harper and J. Zobel, eds, “Proc. ACM-SIGIR Int. Conf. on Research and Development in Information Retrieval”, ACM Press, New York, New Orleans, LA, pp. 358–365.

Clough, P. and Sanderson, M. (2004), Relevance feedback for cross language image retrieval, *in* S. McDonald and J. Tait, eds, “European Conf. on IR Research”, Vol. 2997 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 238–252.

Connell, M., Feng, A., Kumaran, G., Raghavan, H., Shah, C. and Allan, J. (2004), UMass at TDT 2004, *in* “Topic Detection and Tracking Workshop (TDT)”, National Institute of Standards and Technology, Gaithersburg, MD.

Cooper, W. S. (1973), “On selecting a measure of retrieval effectiveness”, *Jour. of the American Society for Information Science* **24**, 87–100.

Craswell, N., Hawking, D. and Robertson, S. E. (2001), Effective site finding using link anchor information, *in* D. H. Kraft, W. B. Croft, D. J. Harper and J. Zobel, eds, “Proc. ACM-SIGIR Int. Conf. on Research and Development in Information Retrieval”, ACM Press, New York, New Orleans, LA, pp. 250–257.

Cristianini, N., Shawe-Taylor, J. and Lodhi, H. (2002), “Latent semantic kernels”, *J. Intell. Inf. Syst.* **18**(2-3), 127–152.

Croft, W. B., ed. (2000), *Advances in Information Retrieval*, Kluwer Academic Publishers, Norwell, MA.

Croft, W. B. and Harper, D. (1979), “Using probabilistic models of information retrieval without relevance information”, *Jour. of Documentation* **35**(4), 285–295.

Croft, W. B. and Lafferty, J. D. (2003), *Language Modeling for Information Retrieval*, Kluwer Academic Publishers, Dordrecht, The Netherlands.

Cronen-Townsend, S., Zhou, Y. and Croft, W. B. (2002), Predicting query performance, *in* M. Beaulieu, R. Baeza-Yates, S. H. Myaeng and K. Järvelin, eds, “Proc. ACM-SIGIR Int. Conf. on Research and Development in Information Retrieval”, ACM Press, New York, Tampere, Finland, pp. 299–306.

Cronen-Townsend, S., Zhou, Y. and Croft, W. B. (2004*a*), A framework for selective query expansion, *in* D. Grossman, L. Gravano, C. Zhai, O. Herzog and D. A. Evans, eds, “Proc. Int. Conf. on Information and Knowledge Management”, ACM Press, New York, Washington, D.C., pp. 236–237.

Cronen-Townsend, S., Zhou, Y. and Croft, W. B. (2004*b*), A language modeling framework for selective query expansion, Technical Report IR-338, Center for Intelligent Information Retrieval, University of Massachusetts.

Crouch, C. J. (1988), A cluster-based approach to thesaurus construction, *in* Y. Chiaramella, ed., “Proc. ACM-SIGIR Int. Conf. on Research and Development in Information Retrieval”, ACM Press, New York, Grenoble, France, pp. 309–320.

Crouch, C. J. and Yang, B. (1992), Experiments in automatic statistical thesaurus construction, *in* N. J. Belkin, P. Ingwersen and A. M. Pejtersen, eds, “Proc. ACM-SIGIR Int. Conf. on Research and Development in Information Retrieval”, ACM Press, New York, Copenhagen, Denmark, pp. 77–88.

Daniel, W. (1990), *Applied Nonparametric Statistics*, 2nd edn, PWS-KENT Publishing Company, Boston, MA.

De Mori, R. and Brugnara, F. (1996), HMM methods in speech recognition, *in* G. B. Varile and A. Zampolli, eds, "Survey of the State of the Art in Human Language Technology", US National Science Foundation and Directorate XIII-E of the European Community, pp. 21–30.

Deerwester, S., Dumais, S. T., Furnas, G. W. and Landauer, T. (1990), "Indexing by latent semantic analysis", *Jour. of the American Society for Information Science* **41**(6), 391–407.

Dempster, A. P., Laird, N. M. and Rubin, D. B. (1977), "Maximum likelihood from incomplete data via the EM algorithm", *Jour. of the Royal Statistical Society (B)* **39**(1), 1–38.

Dennis, S., McArthur, R. and Bruza, P. D. (1998), Searching the world wide web made easy? the cognitive load imposed by query refinement mechanisms, *in* J. Kay and M. Milosavljevic, eds, "Proc. Australian Document Computing Conf.", Sydney, Australia, pp. 65–71.

Doyle, L. B. (1965), "Is automatic classification a reasonable application of statistical analysis of text?", *Jour. of the ACM* **12**(4), 473–489.

Dumais, S. T. (1990), Enhancing performance in latent semantic indexing (LSI) retrieval, Technical report, Bellcore.

Dumais, S. T. (1991), "Improving the retrieval of information from external sources", *Behavior Research Methods, Instruments & Computers* **23**(2), 229–236.

Dumais, S. T. (1994), Latent semantic indexing (LSI): TREC-3 report, *in* D. K. Harman, ed., "Proc. Text Retrieval Conf. (TREC)", National Institute of Standards and Technology Special Publication 500-225, Gaithersburg, MD, pp. 219–230.

Dumais, S. T., Furnas, G. W., Landauer, T. and Deerwester, S. (1988), Using latent semantic analysis to improve information retrieval, *in* "Proc. of the SIGCHI Conf. on Human Factors in Computing Systems", ACM Press, New York, pp. 281–285.

Efthimiadis, E. N. (1993), A user-centred evaluation of ranking algorithms for interactive query expansion, *in* R. Korfhage, E. Rasmussen and P. Willett, eds, "Proc. ACM-SIGIR Int. Conf. on Research and Development in Information Retrieval", ACM Press, New York, Pittsburgh, PA, pp. 146–159.

Efthimiadis, E. N. (1995), "User choices: a new yardstick for the evaluation of ranking algorithms for interactive query expansion", *Information Processing & Management* **31**(4), 605–620.

Efthimiadis, E. N. (1996), "Query expansion", *Annual Review of Information Science and Technology* **31**, 121–187.

Eisenberg, M. and Barry, C. (1988), “Order effects: A study of the possible influence of presentation order on user judgments of document relevance”, *Jour. of the American Society for Information Science* **39**(5), 293–300.

Eisenberg, M. and Hu, X. (1987), Dichotomous relevance judgments and the evaluation of information systems, *in* C.-C. Chen, ed., “Proc. of the 50th Annual Meeting of the American Society for Information Science (ASIS)”, Boston, MA, pp. 66–70.

Elias, P. (1975), “Universal codeword sets and representations of the integers”, *IEEE Transactions on Information Theory* **IT-21**(2), 194–202.

Ellis, D. (1989), “A behavioural approach to information system design”, *Jour. of Documentation* **45**(3), 171–212.

Evans, D. A. and Lefferts, R. G. (1993), Design and evaluation of the CLARIT-TREC-2 system, *in* D. K. Harman, ed., “Proc. Text Retrieval Conf. (TREC)”, National Institute of Standards and Technology Special Publication 500-215, Gaithersburg, MD, pp. 137–150.

Faloutsos, C. (1985), “Access methods for text”, *Computing Surveys* **17**(1), 49–74.

Fitzpatrick, L. and Dent, M. (1997), Automatic feedback using past queries: Social searching?, *in* N. J. Belkin, A. D. Narasimhalu, P. Willett, W. Hersh, F. Can and E. Voorhees, eds, “Proc. ACM-SIGIR Int. Conf. on Research and Development in Information Retrieval”, ACM Press, New York, Philadelphia, PA, pp. 306–313.

Foskett, D. J. (1997), Thesaurus, *in* K. Sparck Jones and P. Willet, eds, “Readings in Information Retrieval”, Morgan Kaufman, San Francisco, CA, pp. 111–134.

Frakes, W. B. (1992), Stemming algorithms, *in* W. B. Frakes and R. Baeza-Yates, eds, “Information Retrieval: Data Structures and Algorithms”, Prentice-Hall, Englewood Cliffs, NJ, pp. 131–160.

Frakes, W. B. and Baeza-Yates, R. (1992), *Information Retrieval: Data Structures and Algorithms*, Prentice-Hall, Englewood Cliffs, NJ.

Fuhr, N., Malik, S. and Lalmas, M. (2004), Overview of the INitiative for the Evaluation of XML retrieval (INEX) 2003, *in* N. Fuhr, M. Lalmas and S. Malik, eds, “Proc. of the second INEX Workshop”, Dagstuhl, Germany, pp. 1–11.

Furnas, G. W. (1985), Experience with an adaptive indexing scheme, *in* L. Borman and R. Smith, eds, “Proc. of the SIGCHI Conf. on Human Factors in Computing Systems”, ACM Press, New York, San Francisco, CA, pp. 131–135.

Furnas, G. W., Landauer, T. K., Gomez, L. M. and Dumais, S. T. (1987), “The vocabulary problem in human-system communication”, *Communications of the ACM* **30**(11), 964–971.

Garcia, S., Williams, H. E. and Cannane, A. (2004), Access-ordered indexes, *in* V. Estivill-Castro, ed., “Proc. of the 27th Conf. on Australasian Computer Science”, Vol. 26, Australian Computer Society, Dunedin, New Zealand, pp. 7–14.

Gauch, S. and Wang, J. (1997), A corpus analysis approach for automatic query expansion, *in* F. Golshani and K. Makki, eds, “Proc. Int. Conf. on Information and Knowledge Management”, ACM Press, New York, Las Vegas, NV, pp. 278–284.

Gelman, A., Carlin, J. B., Stern, H. S. and Rubin, D. B. (2004), *Bayesian Data Analysis*, 2nd edn, Chapman and Hall/CRC, Boca Raton, FL.

Goldstein, J., Kantrowitz, M., Mittal, V. and Carbonell, J. (1999), Summarizing text documents: Sentence selection and evaluation metrics, *in* F. Gey, M. Hearst and R. Tong, eds, “Proc. ACM-SIGIR Int. Conf. on Research and Development in Information Retrieval”, ACM Press, New York, Berkeley, CA, pp. 121–128.

Golomb, S. W. (1966), “Run-length encodings”, *IEEE Transactions on Information Theory* **IT-12**(3), 399–401.

Gravetter, F. J. and Wallnau, L. B. (1991), *Statistics for the Behavioural Sciences*, 3rd edn, West Publishing Company, New York.

Grefenstette, G. (1994), *Explorations in Automatic Thesaurus Discovery*, Kluwer Academic Publishers, Norwell, MA.

Gu, Z. and Luo, M. (2004), Comparison of using passages and documents for blind relevance feedback in information retrieval, *in* M. Sanderson, K. Järveln, J. Allan and P. Bruza, eds, “Proc. ACM-SIGIR Int. Conf. on Research and Development in Information Retrieval”, ACM Press, New York, Sheffield, United Kingdom, pp. 482–483.

Guo, Y., Harkema, H. and Gaizauskas, R. (2004), Sheffield University and the TREC 2004 genomics track: Query expansion using synonymous terms, *in* E. M. Voorhees and L. P. Buckland, eds, “Proc. Text Retrieval Conf. (TREC)”, National Institute of Standards and Technology Special Publication 500-261, Gaithersburg, MD.

Haines, D. and Croft, W. B. (1993), Relevance feedback and inference networks, *in* R. Korfhage, E. Rasmussen and P. Willett, eds, “Proc. ACM-SIGIR Int. Conf. on Research and Development in Information Retrieval”, ACM Press, New York, Pittsburgh, PA, pp. 2–11.

Hall, P. A. V. and Dowling, G. R. (1980), “Approximate string matching”, *Computing Surveys* **12**(4), 381–402.

Hancock-Beaulieu, M., Fowkes, H., Alemayehu, N. and Sanderson, M. (1999), Interactive Okapi at Sheffield – TREC-8, *in* E. M. Voorhees and D. K. Harman, eds, “Proc. Text Retrieval Conf. (TREC)”, National Institute of Standards and Technology Special Publication 500-246, Gaithersburg, MD, pp. 689–698.

Hanks, P., ed. (1988), *The Collin’s Concise Dictionary of the English Language*, Collins, London, United Kingdom.

Harman, D. K. (1988), Towards interactive query expansion, *in* Y. Chiaramella, ed., “Proc. ACM-SIGIR Int. Conf. on Research and Development in Information Retrieval”, ACM Press, New York, Grenoble, France, pp. 321–331.

Harman, D. K. (1992*a*), Overview of the first Text REtrieval Conference (TREC-1), *in* D. K. Harman, ed., “Proc. Text Retrieval Conf. (TREC)”, National Institute of Standards and Technology Special Publication 500-207, Gaithersburg, MD, pp. 1–20.

Harman, D. K. (1992*b*), Relevance feedback revisited, *in* N. J. Belkin, P. Ingwersen and A. M. Pejtersen, eds, “Proc. ACM-SIGIR Int. Conf. on Research and Development in Information Retrieval”, ACM Press, New York, Copenhagen, Denmark, pp. 1–10.

Harman, D. K. and Buckley, C. (2004), The NRRC Reliable Information Access (RIA) workshop, *in* M. Sanderson, K. Järveln, J. Allan and P. Bruza, eds, “Proc. ACM-SIGIR Int. Conf. on Research and Development in Information Retrieval”, ACM Press, New York, Sheffield, United Kingdom, pp. 528–529.

Harper, D. J. and van Rijsbergen, C. J. (1978), “An evaluation of feedback in document retrieval using co-occurrence data”, *Jour. of Documentation* **34**(3), 189–216.

Harter, S. P. (1975), “A probabilistic approach to automatic keyword indexing”, *Jour. of the American Society for Information Science* **26**, 45–50.

Hawking, D. (2000), Overview of the TREC-9 web track, *in* E. M. Voorhees and D. K. Harman, eds, “Proc. Text Retrieval Conf. (TREC)”, National Institute of Standards and Technology Special Publication 500-249, Gaithersburg, MD, pp. 87–103.

Hawking, D. and Craswell, N. (2001), Overview of the TREC-2001 web track, *in* E. M. Voorhees and D. K. Harman, eds, “Proc. Text Retrieval Conf. (TREC)”, National Institute of Standards and Technology Special Publication 500-250, Gaithersburg, MD, pp. 61–67.

Hawking, D. and Craswell, N. (2004), Overview of the TREC-2004 web track, *in* E. M. Voorhees and L. P. Buckland, eds, "Proc. Text Retrieval Conf. (TREC)", National Institute of Standards and Technology Special Publication 500-261, Gaithersburg, MD.

Hawking, D., Craswell, N. and Thistlewaite, P. (1998), Overview of TREC-7 very large collection track, *in* E. M. Voorhees and D. K. Harman, eds, "Proc. Text Retrieval Conf. (TREC)", National Institute of Standards and Technology Special Publication 500-242, Gaithersburg, MD, pp. 91–104.

Hawking, D. and Robertson, S. E. (2003), "On collection size and retrieval effectiveness", *Kluwer International Journal of Information Retrieval* **6**(1), 99–150.

Heinz, S. and Zobel, J. (2003), "Efficient single-pass index construction for text databases", *Jour. of the American Society for Information Science and Technology* **54**(8), 713–729.

Hennessy, J. L. and Patterson, D. A. (2002), *Computer Architecture: A Quantitative Approach*, Morgan Kaufmann, San Mateo, CA.

Hiemstra, D., Robertson, S. and Zaragoza, H. (2004), Parsimonious language models for information retrieval, *in* M. Sanderson, K. Järveln, J. Allan and P. Bruza, eds, "Proc. ACM-SIGIR Int. Conf. on Research and Development in Information Retrieval", ACM Press, New York, Sheffield, United Kingdom, pp. 178–185.

Hoashi, K., Matsumoto, K., Inoue, N. and Hashimoto, K. (1999), Query expansion method based on word contribution, *in* F. Gey, M. Hearst and R. Tong, eds, "Proc. ACM-SIGIR Int. Conf. on Research and Development in Information Retrieval", ACM Press, New York, Berkeley, CA, pp. 303–304.

Hull, D. (1993), Using statistical testing in the evaluation of retrieval experiments, *in* R. Korfhage, E. Rasmussen and P. Willett, eds, "Proc. ACM-SIGIR Int. Conf. on Research and Development in Information Retrieval", ACM Press, New York, Pittsburgh, PA, pp. 329–338.

Hull, D. (1994), Improving text retrieval for the routing problem using latent semantic indexing, *in* W. B. Croft and C. J. van Rijsbergen, eds, "Proc. ACM-SIGIR Int. Conf. on Research and Development in Information Retrieval", Springer-Verlag, Dublin, Ireland, pp. 282–291.

Ide, E. (1971), New experiments in relevance feedback, *in* G. Salton, ed., "The SMART Retrieval System – Experiments in Automatic Document Processing", Prentice-Hall, Englewood Cliffs, NJ, pp. 337–354.

Ide, E. and Salton, G. (1971), Interactive search strategies and dynamic file organization in information retrieval, *in* G. Salton, ed., "The SMART Retrieval System – Experiments in Automatic Document Processing", Prentice-Hall, Englewood Cliffs, NJ, pp. 373–393.

Janes, J. W. (1991), “The binary nature of continuous relevance judgments: A study of users’ perceptions”, *Jour. of the American Society for Information Science* **42**(10), 754–756.

Jansen, B. J., Spink, A. and Pedersen, J. (2005), “A temporal comparison of AltaVista web searching”, *Jour. of the American Society for Information Science and Technology* **56**(6), 559–570.

Jelinek, F. (1990), Self-organized language modeling for speech recognition, *in* “Readings in Speech Recognition”, Morgan Kaufmann Publishers Inc., San Francisco, CA, pp. 450–506.

Jing, Y. and Croft, B. (1994), An association thesaurus for information retrieval, *in* “Proc. of RIAO-94, 4th Int. Conf. “Recherche d’Information Assistee par Ordinateur” (RIAO)”, pp. 146–160.

Joachims, T. (1997), A probabilistic analysis of the Rocchio algorithm with tfidf for text categorization, *in* D. H. Fisher, ed., “Proc. Int. Conf. on Machine Learning”, Morgan Kaufmann Publishers Inc., Nashville, TN, pp. 143–151.

Joachims, T. (2002), Evaluating retrieval performance using clickthrough data, *in* M. Beaulieu, R. Baeza-Yates, S. H. Myaeng and K. Järvelin, eds, “Workshop on Mathematical/Formal Methods in Information Retrieval”, ACM Press, New York, Tampere, Finland.

Joho, H., Coverson, C., Sanderson, M. and Beaulieu, M. (2002), Hierarchical presentation of expansion terms, *in* “Proc. of the 2002 ACM Symposium on Applied Computing”, ACM Press, New York, Madrid, Spain, pp. 645–649.

Joho, H. and Sanderson, M. (2000), Retrieving descriptive phrases from large amounts of free text, *in* A. Agah, J. Callan, E. Rundensteiner and S. Gauch, eds, “Proc. Int. Conf. on Information and Knowledge Management”, ACM Press, New York, McLean, VA, pp. 180–186.

Joho, H., Sanderson, M. and Beaulieu, M. (2002), Hierarchical approach to term suggestion device, *in* M. Beaulieu, R. Baeza-Yates, S. H. Myaeng and K. Järvelin, eds, “Proc. ACM-SIGIR Int. Conf. on Research and Development in Information Retrieval”, ACM Press, New York, Tampere, Finland, pp. 454–454.

Joho, H., Sanderson, M. and Beaulieu, M. (2004), A study of user interaction with a concept-based interactive query expansion support tool, *in* S. McDonald and J. Tait, eds, “European Conf. on IR Research”, Vol. 2997 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 42–56.

Kang, I.-H. and Kim, G. (2003), Query type classification for web document retrieval, *in* J. Callan, G. Cormack, C. Clarke, D. Hawking and A. Smeaton, eds, “Proc. ACM-SIGIR Int. Conf. on Research and Development in Information Retrieval”, ACM Press, New York, Toronto, Canada, pp. 64–71.

Kaszkiel, M. and Zobel, J. (2001), “Effective ranking with arbitrary passages”, *Jour. of the American Society for Information Science* **52**(4), 344–364.

Kazai, G., Lalmas, M. and Piwowarski, B. (2004), INEX relevance assessment guide, *in* N. Fuhr, M. Lalmas and S. Malik, eds, “Proc. of the second INEX Workshop”, Dagstuhl, Germany, pp. 204–209.

Kekäläinen, J. and Järvelin, K. (2002), “Using graded relevance assessments in IR evaluation”, *Jour. of the American Society for Information Science and Technology* **53**(13), 1120–1129.

Kim, W. and Wilbur, W. J. (2001), “Corpus-based statistical screening for content-bearing terms”, *Jour. of the American Society for Information Science and Technology* **52**(3), 247–259.

Knuth, D. E. (1978), *The Art of Computer Programming*, 2nd edn, Addison-Wesley Longman, Boston, MA.

Komarjaya, J., Poo, D. C. C. and Kan, M.-Y. (2004), Corpus-based query expansion in online public access catalogs, *in* “European Conf. on Research and Advanced Technology for Digital Libraries”, Bath, United Kingdom, pp. 61–69.

Kuhlthau, C. C. (1993), “Principle for uncertainty for information seeking”, *Jour. of Documentation* **49**(4), 339–355.

Kullback, S. (1959), *Information Theory and Statistics*, John Wiley and Sons, New York.

Kwok, K. L. (2002), Higher precision for two-word queries, *in* M. Beaulieu, R. Baeza-Yates, S. H. Myaeng and K. Järvelin, eds, “Proc. ACM-SIGIR Int. Conf. on Research and Development in Information Retrieval”, ACM Press, New York, Tampere, Finland, pp. 395–396.

Kwok, K. L. and Chan, M. (1998), Improving two-stage ad-hoc retrieval for short queries, *in* W. B. Croft, A. Moffat, C. J. van Rijsbergen, R. Wilkinson and J. Zobel, eds, “Proc. ACM-SIGIR Int. Conf. on Research and Development in Information Retrieval”, ACM Press, New York, Melbourne, Australia, pp. 250–256.

Lafferty, J. D. and Zhai, C. (2001), Document language models, query models, and risk minimization for information retrieval, *in* D. H. Kraft, W. B. Croft, D. J. Harper and J. Zobel, eds, “Proc. ACM-SIGIR Int. Conf. on Research and Development in Information Retrieval”, ACM Press, New York, New Orleans, LA, pp. 111–119.

Lam-Adesina, A. M. and Jones, G. J. F. (2001), Applying summarization techniques for term selection in relevance feedback, *in* D. H. Kraft, W. B. Croft, D. J. Harper and J. Zobel, eds, “Proc. ACM-SIGIR Int. Conf. on Research and Development in Information Retrieval”, ACM Press, New York, New Orleans, LA, pp. 1–9.

Landauer, T. K., Foltz, P. and Laham, D. (1998), “An introduction to latent semantic analysis”, *Discourse Processes* **25**(2 & 3), 259–284.

Larkey, L. S., Allan, J., Connell, M. E., Bolivar, A. and Wade, C. (2002), UMass at TREC 2002: Cross language and novelty tracks, *in* E. M. Voorhees and L. P. Buckland, eds, “Proc. Text Retrieval Conf. (TREC)”, National Institute of Standards and Technology Special Publication 500-251, Gaithersburg, MD, pp. 721–732.

Lavrenko, V. (2004), A Generative Theory of Relevance, PhD thesis, University of Massachusetts.

Lavrenko, V. and Croft, W. B. (2001), Relevance based language models, *in* D. H. Kraft, W. B. Croft, D. J. Harper and J. Zobel, eds, “Proc. ACM-SIGIR Int. Conf. on Research and Development in Information Retrieval”, ACM Press, New York, New Orleans, LA, pp. 120–127.

Lee, J. H. (1998), “Combining the evidence of different relevance feedback methods for information retrieval”, *Information Processing & Management* **34**(6), 681–691.

Lesk, M. E. (1969), “Word-word associations in document retrieval systems”, *American Documentation* **20**(1), 27–38.

Lester, N. and Williams, H. E. (2002), Topic tracking at RMIT University, *in* “Topic Detection and Tracking Workshop (TDT)”, National Institute of Standards and Technology, Gaithersburg, MD.

Leuski, A. (2000), Relevance and reinforcement in interactive browsing, *in* A. Agah, J. Callan, E. Rundensteiner and S. Gauch, eds, “Proc. Int. Conf. on Information and Knowledge Management”, ACM Press, New York, McLean, VA, pp. 119–126.

Levow, G.-A. and Oard, D. W. (2002), Signal boosting for translingual topic tracking: Document expansion and n-best translation, *in* “Topic Detection and Tracking: Event-Based Information Organization”, Kluwer Academic Publishers, Norwell, MA, pp. 175–195.

Li, Y.-C. and Meng, H. M. (2003), Document expansion using a side collection for monolingual and cross-language spoken document retrieval, *in* “ISCA Workshop on Multilingual Spoken Document Retrieval”, Hong Kong, pp. 85–90.

Liu, S., Liu, F., Yu, C. and Meng, W. (2004), An effective approach to document retrieval via utilizing WordNet and recognizing phrases, *in* M. Sanderson, K. Järveln, J. Allan and P. Bruza, eds, “Proc. ACM-SIGIR Int. Conf. on Research and Development in Information Retrieval”, ACM Press, New York, Sheffield, United Kingdom, pp. 266–272.

Liu, X. and Croft, W. B. (2004), Cluster-based retrieval using language models, *in* M. Sanderson, K. Järveln, J. Allan and P. Bruza, eds, “Proc. ACM-SIGIR Int. Conf. on Research and Development in

Information Retrieval”, ACM Press, New York, Sheffield, United Kingdom, pp. 186–193. relevant to doc expansion paper.

Lovins, J. B. (1968), “Development of a stemming algorithm”, *Mechanical Translation and Computation* **11**(1-2), 22–31.

Luhn, H. P. (1958), “The automatic creation of literature abstracts”, *IBM Jour. of Research and Development* **2**(2), 159–165.

Lundquist, C., Grossman, D. A. and Frieder, O. (1997), Improving relevance feedback in the vector space model, *in* F. Golshani and K. Makki, eds, “Proc. Int. Conf. on Information and Knowledge Management”, ACM Press, New York, Las Vegas, NV, pp. 16–23.

Lynam, T. R., Buckley, C., Clarke, C. L. A. and Cormack, G. V. (2004), A multi-system analysis of document and term selection for blind feedback, *in* D. Grossman, L. Gravano, C. Zhai, O. Herzog and D. A. Evans, eds, “Proc. Int. Conf. on Information and Knowledge Management”, ACM Press, New York, Washington, D.C., pp. 261–269.

Magennis, M. and van Rijsbergen, C. J. (1997), The potential and actual effectiveness of interactive query expansion, *in* N. J. Belkin, A. D. Narasimhalu, P. Willett, W. Hersh, F. Can and E. Voorhees, eds, “Proc. ACM-SIGIR Int. Conf. on Research and Development in Information Retrieval”, ACM Press, New York, Philadelphia, PA, pp. 324–332.

Mandala, R., Tokunaga, T. and Tanaka, H. (1999), Combining multiple evidence from different types of thesaurus for query expansion, *in* F. Gey, M. Hearst and R. Tong, eds, “Proc. ACM-SIGIR Int. Conf. on Research and Development in Information Retrieval”, ACM Press, New York, Berkeley, CA, pp. 191–197.

Mandala, R., Tokunaga, T., Tanaka, H., Okumura, A. and Satoh, K. (1998), Ad hoc retrieval experiments using WordNet and automatically constructed thesauri, *in* E. M. Voorhees and D. K. Harman, eds, “Proc. Text Retrieval Conf. (TREC)”, National Institute of Standards and Technology Special Publication 500-242, Gaithersburg, MD, pp. 475–480.

Manning, C. D. and Schütze, H. (2003), *Foundations of Statistical Natural Language Processing*, The MIT Press, Cambridge Massachusetts.

Mano, H. and Ogawa, Y. (2001), Selecting expansion terms in automatic query expansion, *in* D. H. Kraft, W. B. Croft, D. J. Harper and J. Zobel, eds, “Proc. ACM-SIGIR Int. Conf. on Research and Development in Information Retrieval”, ACM Press, New York, New Orleans, LA, pp. 390–391.

Margulis, E. L. (1993), “Modelling documents with multiple Poisson distributions”, *Information Processing & Management* **29**(2), 215–227.

Maron, M. E. (1961), “Automatic indexing: An experimental inquiry”, *Jour. of the ACM* **8**(3), 404–417.

Metzler, D., Lavrenko, V. and Croft, W. B. (2004), Formal multiple-Bernoulli models for language modeling, *in* M. Sanderson, K. Järveln, J. Allan and P. Bruza, eds, “Proc. ACM-SIGIR Int. Conf. on Research and Development in Information Retrieval”, ACM Press, New York, Sheffield, United Kingdom, pp. 540–541.

Miller, G. A. (1995), “WordNet: a lexical database for English”, *Communications of the ACM* **38**(11), 39–41.

Minker, J., Wilson, G. A. and Zimmerman, B. H. (1972), “An evaluation of query expansion by the addition of clustered terms for a document retrieval system”, *Information Storage and Retrieval* **8**(6), 329–348.

Mitra, M., Singhal, A. and Buckley, C. (1998), Improving automatic query expansion, *in* W. B. Croft, A. Moffat, C. J. van Rijsbergen, R. Wilkinson and J. Zobel, eds, “Proc. ACM-SIGIR Int. Conf. on Research and Development in Information Retrieval”, ACM Press, New York, Melbourne, Australia, pp. 206–214.

Mizzaro, S. (1997), “Relevance: The whole history”, *Jour. of the American Society for Information Science* **49**(9), 810–832.

Mizzaro, S. (1998), “How many relevances in information retrieval?”, *Interacting with Computers* **10**(3), 303–320.

Moffat, A. and Bell, T. A. H. (1995), “In situ generation of compressed inverted files”, *Jour. of the American Society for Information Science* **46**(7), 537–550.

Moffat, A. and Zobel, J. (1994), Fast ranking in limited space, *in* “Proc. IEEE Int. Conf. on Data Engineering”, IEEE Computer Society Press, Los Alamitos, California, Houston, TX, pp. 428–437.

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

Moffat, A., Zobel, J. and Sacks-Davis, R. (1994), “Memory efficient ranking”, *Information Processing & Management* **30**(6), 733–744.

Montgomery, J., Si, L., Callan, J. and Evans, D. A. (2004), Effect of varying number of documents in blind feedback: Analysis of the 2003 NRRC RIA workshop bf_numdocs experiment suite, *in* M. Sanderson, K. Järveln, J. Allan and P. Bruza, eds, “Proc. ACM-SIGIR Int. Conf. on Research and Development in Information Retrieval”, ACM Press, New York, Sheffield, United Kingdom, pp. 476–477.

Nallapati, R., Croft, B. and Allan, J. (2003), Relevant query feedback in statistical language modeling, *in* D. Kraft, O. Frieder, J. Hammer, S. Qureshi and L. Seligman, eds, “Proc. Int. Conf. on Information and Knowledge Management”, ACM Press, New York, New Orleans, LA, pp. 560–563.

Needham, R. M. and Sparck Jones, K. (1964), “Keywords and clumps”, *Jour. of Documentation* **20**, 5–15.

Nethercote, N. and Seward, J. (2003), Valgrind: A program supervision framework, *in* O. Sokolsky and M. Viswanathan, eds, “Proc. of the third Workshop on Runtime Verification (RV’03)”, Boulder, CO.

Ozcan, R. and Aslandogan, Y. A. (2004), Concept based information access using ontologies and latent semantic analysis information retrieval, Technical report, Department of Computer Science and Engineering University of Texas at Arlington.

Page, L., Brin, S., Motwani, R. and Winograd, T. (1998), The PageRank citation ranking: Bringing order to the web, Technical report, Stanford Digital Library Technologies Project.

Park, L. A. F. and Ramamohanarao, K. (2004), Hybrid pre-query term expansion using latent semantic analysis, *in* “Proc. of the fourth IEEE Int. Conf. on Data Mining (ICDM’04)”, IEEE Computer Society, Washington, DC, USA, pp. 178–185.

Patterson, D. A. and Hennessy, J. L. (2005), *Computer Organization and Design*, 3rd edn, Morgan Kaufman Publishers, San Francisco, CA.

Peat, H. J. and Willett, P. (1991), “The limitations of term co-occurrence data for query expansion in document retrieval systems”, *Jour. of the American Society for Information Science* **42**(5), 378–383.

Persin, M., Zobel, J. and Sacks-Davis, R. (1996), “Filtered document retrieval with frequency-sorted indexes”, *Jour. of the American Society for Information Science* **47**(10), 749–764.

Ponte, J. M. and Croft, W. B. (1998), A language modeling approach to information retrieval, *in* W. B. Croft, A. Moffat, C. J. van Rijsbergen, R. Wilkinson and J. Zobel, eds, “Proc. ACM-SIGIR Int. Conf. on Research and Development in Information Retrieval”, ACM Press, New York, Melbourne, Australia, pp. 275–281.

Porter, M. F. (1980), “An algorithm for suffix stripping”, *Program* **13**(3), 130–137.

Qiu, Y. and Frei, H.-P. (1993), Concept based query expansion, *in* R. Korfhage, E. Rasmussen and P. Willett, eds, “Proc. ACM-SIGIR Int. Conf. on Research and Development in Information Retrieval”, ACM Press, New York, Pittsburgh, PA, pp. 160–169.

R. Mandala, T. Tokunaga, H. T. (2000), “The exploration and analysis of using multiple thesaurus types for query expansion in information retrieval”, *Jour. of Natural Language Processing* **7**(2), 117–140.

Raghavan, V. V. and Sever, H. (1995), On the reuse of past optimal queries, *in* E. A. Fox, P. Ingwersen and R. Fidel, eds, "Proc. ACM-SIGIR Int. Conf. on Research and Development in Information Retrieval", ACM Press, New York, Seattle, WA, pp. 344–350.

Ramakrishnan, R. and Gehrke, J. (2003), *Database Management Systems*, 3rd edn, McGraw-Hill.

Robertson, S. E. (1977), "The probability ranking principle in IR", *Jour. of Documentation* **33**(4), 294–304.

Robertson, S. E. (1986), "On relevance weight estimation and query expansion", *Jour. of Documentation* **42**(3), 182–188.

Robertson, S. E. (1990), "On term selection for query expansion", *Jour. of Documentation* **46**(4), 359–364.

Robertson, S. E. and Sparck Jones, K. (1976), "Relevance weighting of search terms", *Jour. of the American Society for Information Science* **27**(3), 129–146.

Robertson, S. E., van Rijsbergen, C. J. and Porter, M. F. (1981), Probabilistic models of indexing and searching, *in* C. J. Crouch, ed., "Proc. ACM-SIGIR Int. Conf. on Research and Development in Information Retrieval", Butterworth & Co., Cambridge, England, pp. 35–56.

Robertson, S. E. and Walker, S. (1994), Some simple effective approximations to the 2-Poisson model for probabilistic weighted retrieval, *in* W. B. Croft and C. J. van Rijsbergen, eds, "Proc. ACM-SIGIR Int. Conf. on Research and Development in Information Retrieval", Springer-Verlag, Dublin, Ireland, pp. 232–241.

Robertson, S. E. and Walker, S. (1999), Okapi/Keenbow at TREC-8, *in* E. M. Voorhees and D. K. Harman, eds, "Proc. Text Retrieval Conf. (TREC)", National Institute of Standards and Technology Special Publication 500-246, Gaithersburg, MD, pp. 151–161.

Robertson, S. E. and Walker, S. (2000), Microsoft Cambridge at TREC-9: Filtering track, *in* E. M. Voorhees and D. K. Harman, eds, "Proc. Text Retrieval Conf. (TREC)", National Institute of Standards and Technology Special Publication 500-249, Gaithersburg, MD, pp. 361–368.

Robertson, S. E., Walker, S. and Beaulieu, M. (1998), Okapi at TREC-7: Automatic ad hoc, filtering, VLC and interactive, *in* E. M. Voorhees and D. K. Harman, eds, "Proc. Text Retrieval Conf. (TREC)", National Institute of Standards and Technology Special Publication 500-242, Gaithersburg, MD, pp. 253–264.

Robertson, S. E., Walker, S., Hancock-Beaulieu, M., Gull, A. and Lau, M. (1992), Okapi at TREC, *in* D. K. Harman, ed., "Proc. Text Retrieval Conf. (TREC)", National Institute of Standards and Technology Special Publication 500-207, Gaithersburg, MD, pp. 21–30.

Robertson, S. E., Walker, S., Jones, S., Hancock-Beaulieu, M. M. and Gatford, M. (1994), Okapi at TREC-3, *in* D. K. Harman, ed., “Proc. Text Retrieval Conf. (TREC)”, National Institute of Standards and Technology Special Publication 500-225, Gaithersburg, MD, pp. 109–126.

Robertson, S. and Soboroff, I. (2001), The TREC 2001 filtering track report, *in* E. M. Voorhees and D. K. Harman, eds, “Proc. Text Retrieval Conf. (TREC)”, National Institute of Standards and Technology Special Publication 500-250, Gaithersburg, MD, pp. 26–38.

Rocchio, J. J. (1971), Relevance feedback in information retrieval, *in* G. Salton, ed., “The SMART Retrieval System – Experiments in Automatic Document Processing”, Prentice-Hall, Englewood Cliffs, NJ, pp. 313–323.

Ruthven, I. (2003), Re-examining the potential effectiveness of interactive query expansion, *in* J. Callan, G. Cormack, C. Clarke, D. Hawking and A. Smeaton, eds, “Proc. ACM-SIGIR Int. Conf. on Research and Development in Information Retrieval”, ACM Press, New York, Toronto, Canada, pp. 213–220.

Ruthven, I. and Lalmas, M. (2003), “A survey on the use of relevance feedback for information access systems”, *Knowledge Engineering Review* **18**(2), 95–145.

Sacks-Davis, R., Kent, A. J. and Ramamohanarao, K. (1987), “Multikey access methods based on superimposed coding techniques”, *ACM Transactions on Database Systems* **12**(4), 655–696.

Sakai, T. and Robertson, S. E. (2001), Flexible pseudo-relevance feedback using optimization tables, *in* D. H. Kraft, W. B. Croft, D. J. Harper and J. Zobel, eds, “Proc. ACM-SIGIR Int. Conf. on Research and Development in Information Retrieval”, ACM Press, New York, New Orleans, LA, pp. 396–397.

Salton, G. (1980), “Automatic term class construction using relevance – a summary of work in automatic pseudoclassification”, *Information Processing & Management* **16**(1), 1–15.

Salton, G. and Buckley, C. (1988), “Term-weighting approaches in automatic text retrieval”, *Information Processing & Management* **24**(5), 513–523.

Salton, G. and Buckley, C. (1990), “Improving retrieval performance by relevance feedback”, *Jour. of the American Society for Information Science* **41**(4), 288–297.

Salton, G., ed. (1971), *The SMART Retrieval System – Experiments in Automatic Document Processing*, Prentice-Hall, Englewood Cliffs, NJ.

Salton, G. and McGill, M. J. (1983), *Introduction to Modern Information Retrieval*, McGraw-Hill, New York.

Sanderson, M. (1994), Word sense disambiguation and information retrieval, *in* W. B. Croft and C. J. van Rijsbergen, eds, “Proc. ACM-SIGIR Int. Conf. on Research and Development in Information Retrieval”, Springer-Verlag, Dublin, Ireland, pp. 142–151.

Sanderson, M. and Croft, B. (1999), Deriving concept hierarchies from text, *in* F. Gey, M. Hearst and R. Tong, eds, “Proc. ACM-SIGIR Int. Conf. on Research and Development in Information Retrieval”, ACM Press, New York, Berkeley, CA, pp. 206–213.

Saracevic, T. (1975), “Relevance: a review of and a framework for the thinking on the notion in information science”, *Jour. of the American Society for Information Science* **26**(6), 321–343.

Saracevic, T. (1999), “Information science”, *Jour. of the American Society for Information Science* **50**(12), 1051–1063.

Schamber, L. (1994), “Relevance and information behavior”, *Annual Review of Information Science and Technology* **29**, 3–48.

Scholer, F. (2004), Query Association for Effective Retrieval, PhD thesis, RMIT University, Melbourne, Australia.

Scholer, F. and Williams, H. E. (2002), Query association for effective retrieval, *in* C. Nicholas, D. Grossman, K. Kalpakis, S. Qureshi, H. van Dissel and L. Seligman, eds, “Proc. Int. Conf. on Information and Knowledge Management”, ACM Press, New York, McLean, VA, pp. 324–331.

Scholer, F., Williams, H. E. and Turpin, A. (2004), “Query association surrogates for web search”, *Jour. of the American Society for Information Science and Technology* **55**(7), 637–650.

Scholer, F., Williams, H. E., Yiannis, J. and Zobel, J. (2002), Compression of inverted indexes for fast query evaluation, *in* M. Beaulieu, R. Baeza-Yates, S. H. Myaeng and K. Järvelin, eds, “Proc. ACM-SIGIR Int. Conf. on Research and Development in Information Retrieval”, ACM Press, New York, Tampere, Finland, pp. 222–229.

Schwartz, C. (1998), “Web search engines”, *Jour. of the American Society for Information Science* **49**(11), 973–982.

Shannon, C. E. (2001), “A mathematical theory of communication”, *SIGMOBILE Mob. Comput. Commun. Rev.* **5**(1), 3–55.

Sheskin, D. J. (1997), *Handbook of Parametric and Nonparametric Statistical Procedures*, CRC Press LLC, Boca Raton, FA.

Singhal, A., Buckley, C. and Mitra, M. (1996), Pivoted document length normalization, *in* H.-P. Frei, D. Harman, P. Schäubie and R. Wilkinson, eds, “Proc. ACM-SIGIR Int. Conf. on Research and Development in Information Retrieval”, ACM Press, New York, Zürich, Switzerland, pp. 21–29.

Singhal, A. and Pereira, F. (1999), Document expansion for speech retrieval, *in* F. Gey, M. Hearst and R. Tong, eds, “Proc. ACM-SIGIR Int. Conf. on Research and Development in Information Retrieval”, ACM Press, New York, Berkeley, CA, pp. 34–41.

Smeaton, A. F. and van Rijsbergen, C. J. (1983), “The retrieval effects of query expansion on a feedback document retrieval system”, *Computer Jour.* **26**(3), 239–246.

Soboroff, I. and Harman, D. K. (2003), Overview of the TREC 2003 novelty track, *in* E. M. Voorhees and L. P. Buckland, eds, “Proc. Text Retrieval Conf. (TREC)”, National Institute of Standards and Technology Special Publication 500-255, Gaithersburg, MD, pp. 38–53.

Soboroff, I. and Nicholas, C. (2000), Collaborative filtering and the generalized vector space model, *in* E. Yannakoudakis, N. J. Belkin, M.-K. Leong and P. Ingwersen, eds, “Proc. ACM-SIGIR Int. Conf. on Research and Development in Information Retrieval”, ACM Press, New York, Athens, Greece, pp. 351–353.

Song, F. and Croft, W. B. (1999), A general language model for information retrieval, *in* “Proc. Int. Conf. on Information and Knowledge Management”, ACM Press, New York, Kansas City, MO, pp. 316–321.

Sparck Jones, K. and Jackson, D. M. (1970), “The use of automatically-obtained keyword classifications for information retrieval”, *Information Storage and Retrieval* **5**, 175–201.

Sparck Jones, K., Walker, S. and Robertson, S. E. (2000), “A probabilistic model of information retrieval: Development and comparative experiments. Parts 1&2”, *Information Processing & Management* **36**(6), 779–840.

Spink, A. and Losee, R. M. (1996), “Feedback in information retrieval”, *Annual Review of Information Science and Technology* **34**, 37–78.

Spink, A. and Wilson, T. D. (1999), Toward a theoretical framework for information retrieval (IR) evaluation in an information seeking context, *in* S. Draper, M. Dunlop, I. Ruthven and C. J. van Rijsbergen, eds, “Proc. of MIRA 99: Evaluation Information Retrieval”, Electronic Workshops in Computing, Glasgow, Scotland, United Kingdom, pp. 75–92.

Spink, A., Wolfram, D., Jansen, M. B. J. and Saracevic, T. (2001), “Searching the web: the public and their queries”, *Jour. of the American Society for Information Science and Technology* **52**(3), 226–234.

Spink, A., Wolfram, D., Jansen, M. B. J. and Saracevic, T. (2002), “From e-sex to e-commerce: Web search changes”, *IEEE Computer* **35**(3), 107–109.

Taghva, K., Borsack, J., Nartker, T. and Condit, A. (2004), “The role of manually-assigned keywords in query expansion”, *Information Processing & Management* **40**(3), 441–458.

Tombros, A. and Sanderson, M. (1998), Advantages of query biased summaries in information retrieval, *in* W. B. Croft, A. Moffat, C. J. van Rijsbergen, R. Wilkinson and J. Zobel, eds, “Proc. ACM-SIGIR Int. Conf. on Research and Development in Information Retrieval”, ACM Press, New York, Melbourne, Australia, pp. 2–10.

Tseng, Y.-H. and Juang, D.-W. (2003), Document-self expansion for text categorization, *in* J. Callan, G. Cormack, C. Clarke, D. Hawking and A. Smeaton, eds, “Proc. ACM-SIGIR Int. Conf. on Research and Development in Information Retrieval”, ACM Press, New York, Toronto, Canada, pp. 399–400.

Upstill, T., Craswell, N. and Hawking, D. (2003), “Query-independent evidence in home page finding”, *ACM Transactions on Information Systems* **21**(3), 286–313.

van Rijsbergen, C. J. (1977), “A theoretical basis for the use of co-occurrence data in information retrieval”, *Jour. of Documentation* **33**(2), 106–119.

van Rijsbergen, C. J. (1979), *Information Retrieval*, 2nd edn, Dept. of Computer Science, University of Glasgow.

van Rijsbergen, C. J. (2000), “Another look at the logical uncertainty principle”, *Kluwer International Journal of Information Retrieval* **2**(1), 17–26.

van Rijsbergen, C. J., Harper, D. J. and Porter, M. F. (1981), “The selection of good search terms”, *Information Processing & Management* **17**(2), 77–91.

Vinay, V., Cox, I. J., Milic-Frayling, N. and Wood, K. (2004), Evaluating relevance feedback and display strategies for searching on small displays, *in* A. Apostolico and M. Melucci, eds, “Proc. String Processing and Information Retrieval Symp.”, Springer-Verlag, Padova, Italy, pp. 131–133.

Voorhees, E. M. (1994), Query expansion using lexical-semantic relations, *in* W. B. Croft and C. J. van Rijsbergen, eds, “Proc. ACM-SIGIR Int. Conf. on Research and Development in Information Retrieval”, Springer-Verlag, Dublin, Ireland, pp. 61–69.

Voorhees, E. M. (1998), Variations in relevance judgments and the measurement of retrieval effectiveness, *in* W. B. Croft, A. Moffat, C. J. van Rijsbergen, R. Wilkinson and J. Zobel, eds, “Proc. ACM-SIGIR Int. Conf. on Research and Development in Information Retrieval”, ACM Press, New York, Melbourne, Australia, pp. 315–323.

Voorhees, E. M. (2000), “Variations in relevance judgments and the measurement of retrieval effectiveness”, *Information Processing & Management* **36**(5), 697–716.

Voorhees, E. M. (2001), Evaluation by highly relevant documents, *in* D. H. Kraft, W. B. Croft, D. J. Harper and J. Zobel, eds, “Proc. ACM-SIGIR Int. Conf. on Research and Development in Information Retrieval”, ACM Press, New York, New Orleans, LA, pp. 74–82.

Voorhees, E. M. (2003), Overview of the TREC 2003 question answering track, *in* E. M. Voorhees and L. P. Buckland, eds, “Proc. Text Retrieval Conf. (TREC)”, National Institute of Standards and Technology Special Publication 500-255, Gaithersburg, MD, pp. 54–68.

Voorhees, E. M. (2004), Overview of the TREC 2004 robust track, *in* E. M. Voorhees and L. P. Buckland, eds, “Proc. Text Retrieval Conf. (TREC)”, National Institute of Standards and Technology Special Publication 500-261, Gaithersburg, MD.

Voorhees, E. M. and Buckley, C. (2002), The effect of topic set size on retrieval experiment error, *in* M. Beaulieu, R. Baeza-Yates, S. H. Myaeng and K. Järvelin, eds, “Proc. ACM-SIGIR Int. Conf. on Research and Development in Information Retrieval”, ACM Press, New York, Tampere, Finland, pp. 316–323.

Voorhees, E. M. and Harman, D. K. (1998), Overview of the seventh Text REtrieval Conference (TREC-7), *in* E. M. Voorhees and D. K. Harman, eds, “Proc. Text Retrieval Conf. (TREC)”, National Institute of Standards and Technology Special Publication 500-242, Gaithersburg, MD, pp. 1–24.

Voorhees, E. M. and Harman, D. K. (1999), Overview of the eighth Text REtrieval Conference (TREC-8), *in* E. M. Voorhees and D. K. Harman, eds, “Proc. Text Retrieval Conf. (TREC)”, National Institute of Standards and Technology Special Publication 500-246, Gaithersburg, MD, pp. 1–23.

Voorhees, E. M. and Harman, D. K. (2000), Overview of the ninth Text REtrieval Conference (TREC-9), *in* E. M. Voorhees and D. K. Harman, eds, “Proc. Text Retrieval Conf. (TREC)”, National Institute of Standards and Technology Special Publication 500-249, Gaithersburg, MD, pp. 1–14.

Voorhees, E. M. and Harman, D. K. (2001), Overview of TREC 2001, *in* E. M. Voorhees and D. K. Harman, eds, “Proc. Text Retrieval Conf. (TREC)”, National Institute of Standards and Technology Special Publication 500-250, Gaithersburg, MD, pp. 1–15.

Warren, R. H. and Liu, T. (2004), A review of relevance feedback experiments at the 2003 Reliable Information Access (RIA) workshop, *in* M. Sanderson, K. Järveln, J. Allan and P. Bruza, eds, “Proc. ACM-SIGIR Int. Conf. on Research and Development in Information Retrieval”, ACM Press, New York, Sheffield, United Kingdom, pp. 570–571.

Williams, H. E. and Zobel, J. (2005), “Searchable words on the web”, *Int. Jour. of Digital Libraries* **5**(2), 99–105.

Witten, I. H., Moffat, A. and Bell, T. C. (1999), *Managing Gigabytes: Compressing and Indexing Documents and Images*, 2nd edn, Morgan Kaufman, San Francisco, CA.

Wu, C. F. (1983), “On the convergence properties of the EM algorithm”, *Annals of Statistics* **11**(1), 95–103.

Xi, W., Fox, E. A., Tan, R. P. and Shu, J. (2002), Machine learning approach for homepage finding task, *in* A. H. F. Laender and A. L. Oliveira, eds, “Proc. String Processing and Information Retrieval Symp.”, Springer-Verlag, Lisbon, Portugal, pp. 145–159.

Xu, J. and Croft, W. B. (1996), Query expansion using local and global document analysis, *in* H.-P. Frei, D. Harman, P. Schäubie and R. Wilkinson, eds, “Proc. ACM-SIGIR Int. Conf. on Research and Development in Information Retrieval”, ACM Press, New York, Zürich, Switzerland, pp. 4–11.

Xu, J. and Croft, W. B. (2000), “Improving the effectiveness of information retrieval with local context analysis”, *ACM Transactions on Information Systems* **18**(1), 79–112.

Zhai, C. and Lafferty, J. D. (2001*a*), Model-based feedback in the language modeling approach to information retrieval, *in* H. Paques, L. Liu and D. Grossman, eds, “Proc. Int. Conf. on Information and Knowledge Management”, ACM Press, New York, Atlanta, GA, pp. 403–410.

Zhai, C. and Lafferty, J. D. (2001*b*), A study of smoothing methods for language models applied to ad hoc information retrieval, *in* D. H. Kraft, W. B. Croft, D. J. Harper and J. Zobel, eds, “Proc. ACM-SIGIR Int. Conf. on Research and Development in Information Retrieval”, ACM Press, New York, New Orleans, LA, pp. 334–342.

Zhai, C. and Lafferty, J. D. (2002), Two-stage language models for information retrieval, *in* M. Beaulieu, R. Baeza-Yates, S. H. Myaeng and K. Järvelin, eds, “Proc. ACM-SIGIR Int. Conf. on Research and Development in Information Retrieval”, ACM Press, New York, Tampere, Finland, pp. 49–56.

Zhai, C. and Lafferty, J. D. (2004), “A study of smoothing methods for language models applied to information retrieval”, *ACM Transactions on Information Systems* **22**(2), 179–214.

Zobel, J. (1998), How reliable are the results of large-scale information retrieval experiments?, *in* W. B. Croft, A. Moffat, C. J. van Rijsbergen, R. Wilkinson and J. Zobel, eds, “Proc. ACM-SIGIR Int. Conf. on Research and Development in Information Retrieval”, ACM Press, New York, Melbourne, Australia, pp. 307–314.

Zobel, J., Moffat, A. and Ramamohanarao, K. (1998), “Inverted files versus signature files for text indexing”, *ACM Transactions on Database Systems* **23**(4), 453–490.