# Solving The Word Mismatch Problem Through Automatic Text Analysis

A Dissertation Presented

by

Jinxi Xu

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

Doctor of Philosophy

May 1997

Department of Computer Science

# Solving The Word Mismatch Problem Through Automatic Text Analysis

A Dissertation Presented

by

Jinxi Xu

Approved as to style and content by:

W. Bruce Croft, Chair

James Callan, Member

Jack Wileden, Member

Thomas Roeper, Member

David Stemple, Department Chair
Computer Science

To my wife Hongmin and my parents

## Acknowledgements

Thanks to Bruce Croft for his guidance throughout this research. Thanks to members of my dissertation committee, Jamie Callan, Jack Wileden and Thomas Roeper whose valuable comments improved this disserataion.

Thanks to all CIIR faculty, students and staff members for their help during my study. Special thanks to Jay Ponte, who carefully read earlier drafts of this dissertation, corrected many errors in grammer and provided with many valuable suggestions. Thanks to James Allan, Lisa Ballesteros, John Broglio, Yufeng Jing, Warren Greiff, Bob Krovetz, Steve Harding and Dan Nachbar for making software available, help in experiments and helpful discussions.

Abstract

Solving The Word Mismatch Problem
Through Automatic Text Analysis

May 1997

Jinxi Xu, B.S., Hunan University

M.S., Institute of Computing Technology, the Chinese Academy of Sciences

Ph.D., University of Massachusetts Amherst

Directed by: Professor W. Bruce Croft

Information Retrieval (IR) is concerned with locating documents that are relevant for a user's information need or query from a large collection of documents. A fundamental problem for information retrieval is word mismatch. A query is usually a short and incomplete description of the underlying information need. The users of IR systems and the authors of the documents often use different words to refer to the same concepts.

This thesis addresses the word mismatch problem through automatic text analysis. We investigate two text analysis techniques, corpus analysis and local context analysis, and apply them in two domains of word mismatch, stemming and general query expansion. Experimental results show that these techniques can result in more effective retrieval.

Table of Contents

## List of Tables

## List of Figures

# Chapter 1

# Introduction

## 1.1 Information Retrieval and INQUERY

Information Retrieval (IR) is concerned with locating documents relevant to a user's information need from a collection of documents. The user describes his/her information need using a query which consists of a number of words. Information retrieval systems compare the query with the documents in the collection and return the documents that are likely to satisfy the information need. In this thesis we will use the terms "information need" and "query" interchangeably.

There are two central problems in IR. One is how to represent the queries and the documents. The other is how to compare the representation of the queries and the representation of the documents. In the field of representation, most research attempts have been directed at solving the word mismatch problem, which refers to the phenomenon that the users of IR systems often use different words to describe the concepts in their queries than the authors use to describe the same concepts in their documents. The word mismatch problem is a serious one, as observed by Furnas et al in a more general context[20]. In their experiments, two people use the same term to describe an object less than 20% of the time. The problem is more severe for short casual queries than for long elaborate queries because as queries get longer, there is more chance of some important words co-occurring in the query and the relevant documents. With the advent of the Internet, short causal queries have become increasingly more common. Consequently, techniques to address the word mismatch problem are more urgently needed than ever. Solving the word mismatch problem through automatic text analysis is the focus of this thesis. We will revisit this issue later in this chapter.

The problem of how to compare query and document representations is addressed by retrieval models. Three widely used retrieval models are Boolean, vector space and probabilistic. In the Boolean retrieval model, a query is represented as a Boolean expression, in which a query word is regarded as a proposition. If the query word occurs in the document being compared with the query, the proposition is true. Otherwise, it is false. The document matches the query if it satisfies the whole Boolean expression. As the response to a query, a Boolean retrieval system returns a set of documents which match the query.

In the vector space model [56], we have a set of representation features. A feature is usually a word or term. Queries and documents are represented as vectors in a high dimensional space in which a dimension corresponds to a feature. The $i$th element in a query vector or document vector represents the importance or weight of the $i$th feature in the query or document. The weights in a query vector are specified by the user. The weights in a document vector are usually estimated based on the occurrence statistics of the features in the document and in the collection. The cosine of the angle between a query vector and a document vector is normally used to measure how well the document matches the query. Unlike a Boolean retrieval system, a vector space retrieval system outputs a ranked list of documents based on their cosine similarity with the query.

In the probabilistic model, the retrieval is based on the Probability Ranking Principle [50] which asserts that best retrieval effectiveness will be achieved when documents are ranked in decreasing order according to their probability of relevance. Similar to a vector space retrieval system, a probabilistic retrieval system outputs a ranked list of documents in response to a query. Traditionally, the probability that a document is relevant to a query $Q$ is estimated using Bayes' theorem based on a training set of documents whose relevance to $Q$ is known. Recently, a new method to estimate the probability was proposed by Turtle [64] based on the Bayesian inference network [41]. To distinguish it from the probability estimation method used

in the traditional probabilistic model, this new method is named the *inference network model.*

The INQUERY retrieval system [9], which is the retrieval system used in this thesis, is based on the inference network model. To facilitate the future discussion in the thesis, we describe the inference network model and INQUERY in slightly more detail. The inference network model views information retrieval as an inferential or evidential reasoning process. An inference network is a directed acyclic dependency graph in which nodes represent propositional variables and edges represent dependence relations between propositions. An edge from proposition $Y$ to proposition $X$ represents that $Y$ implies or causes $X$. $Y$ is called a parent node of $X$. In the network, with the exception of the nodes corresponding to the documents, which have no parent nodes, every node has a link matrix which specifies how to calculate the probability for the node given the probabilities of its parents. Alternatively, canonical link matrices can be written as formulas, such as the following INQUERY operators:

$$\begin{array}{lcl} p_{not}(X) & = & 1 - p_1 \\ p_{or}(X) & = & 1 - (1 - p_1)...(1 - p_n) \\ p_{and}(X) & = & p_1 p_2 ... p_n \\ p_{max}(X) & = & max(p_1, ..., p_n) \\ p_{wsum}(X) & = & \frac{w_1 p_1 + w_2 p_2 + ... + w_n p_n}{w_1 + w_2 + ... + wn} \\ p_{sum}(X) & = & \frac{p_1 + p_2 + ... + p_n}{n} \end{array}$$

In the formulas, the $p_i$'s are probabilities of $X$'s parents. The names of the above operators are NOT, OR, AND, MAX, WSUM (weighted sum) and SUM. The last two operators are the most commonly used.

In INQUERY, the probability $P(Q|t_j)$ that a document $t_j$ satisfies a query $Q$ is calculated based on a network like the one shown in Figure 1.1. In the figure, $r_i$'s are representation concepts and $c_i$'s are query concepts of $Q$. The concepts can be single terms or phrases. For convenience, we sometimes also refer query concepts as query words. Though the inference network model allows a query concept $c_i$ to have multiple parent nodes, $c_i$ usually has exactly one parent node $r_i$ and the two are identical. The probability $P(r_i|t_j)$ is estimated based on $r_i$'s frequency in $t_j$ (term

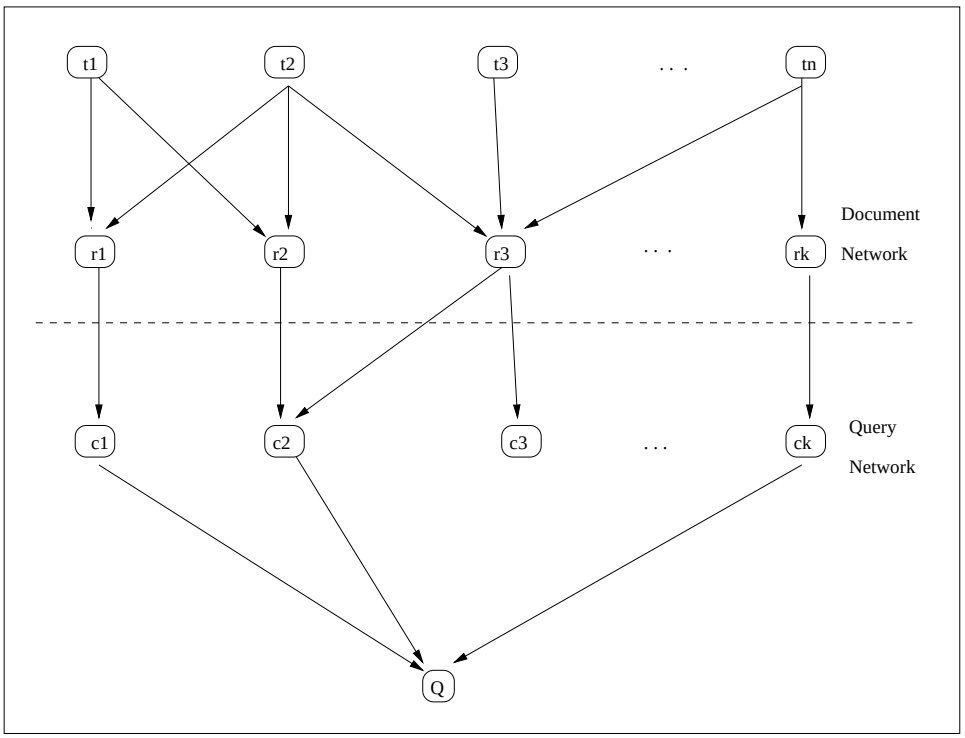


Figure 1.1. An example inference network

frequency–tf) and its frequency in the collection (inverse document frequency–idf). The more frequent $r_i$ is in $t_j$ and the less frequent it is in the collection, the larger is the estimation. The most common operator for node $Q$ is WSUM. The calculation of $P(Q|t_j)$ starts with document $t_j$ by making $P(t_j) = 1$ (for all remaining documents $t_i$'s, $i \neq j$, $P(t_i)$ is set to 0). The calculation propagates downwards until node $Q$ is reached.

## 1.2 Existing Approaches to Solving Word Mismatch

There are a number of existing approaches to solving the word mismatch problem:

**Stemming**. Morphological variation is the most obvious and the narrowest type of word mismatch. For example, if a query contains the word "computer", the user is usually also interested in documents containing "computers" though "computers" is not in the query. This type of word mismatch is often dealt with by stemming. A stemming algorithm, or stemmer, reduces words in queries and documents to common roots. As the result of the reduction, documents containing morphological

variants of the query words are retrieved as well as the documents containing the exact query words. Example stemmers for English are the Porter stemmer [46], the Lovins stemmer [35] and KSTEM [31].

**Manual Thesauri**. Synonymous relationships between words are another type of word mismatch. For example, if a query contains the word "automobile", documents that contain the words "car", "minivan", "caravan" and so on are likely to be relevant.

Manual thesauri are compiled by linguists and domain experts. A well known thesaurus is WordNet [39]. An entry in these thesauri lists not only a word but also the synonyms of the word identified by the human experts. These thesauri are consulted in query processing so that synonyms of the query words are added into the original query. A major disadvantage of manual thesauri is that they are costly to build.

**Automatic Thesauri**. Automatic thesauri provide an alternative to manual thesauri. The underlying hypothesis for automatic thesauri construction is the so called *association hypothesis*: related words in a corpus (collection) of documents tend to co-occur in the documents of the corpus [67]. Words that co-occur with the query words are used for query expansion. This usually can capture word relationships that are specific to a corpus. For example, the word "growth" has different meanings in a medical corpus and a corpus of Wall Street Journal articles. It will co-occur with words such as "tumor" and "cancer" in one corpus and with words such as "profit" and "economy" in the other. The simplest technique expands each query word independently by adding its co-occurring words. More sophisticated techniques expand a query as a whole by adding words co-occurring with all of the query words.

**Dimensionality Reduction**. Dimensionality reduction techniques decompose words and documents into vectors in a low dimensional space. The dimensions in the space can be viewed as "semantic primitives" which can be linearly combined to form the "meaning" of any word, document or query. The word mismatch problem is addressed because any word can be matched with another word to some degree in

the low dimensional space. Latent Semantic Indexing (LSI) is an example of these techniques [16].

**Relevance Feedback**. Relevance feedback is a technique that modifies a query based on the relevance judgment on the retrieved documents [56]. Common words from the relevant documents are added to the query and the weights of the query words are adjusted according to their frequencies in the relevant and non-relevant documents. Relevance feedback techniques can find valuable words which have no obvious semantic relationship with the original query words. For example, "apartheid", "ANC" and "Nelson Mandela" may be added to the query "sanctions against South Africa". It is hard for thesauri-based techniques to find such words for query expansion. The major disadvantage of relevance feedback is that it increases the burden on the users.

**Local Feedback**. Local feedback is similar to relevance feedback [2, 13]. The only difference is that it assumes that the top retrieved documents for a query are relevant without human judgment. It saves the cost of relevance judgment, but it can result in poor retrieval if the top retrieved documents used for feedback are non-relevant.

## 1.3 Research Summary

The general research goal of this thesis is to address the word mismatch problem through automatic text analysis. We will employ two techniques for text analysis in this thesis:

- Corpus analysis, which attempts to establish word relationships based on co-occurrence information in a corpus. The underlying hypothesis is the above mentioned association hypothesis.

- Local context analysis, which attempts to identify words that help to characterize the underlying information need of a query based on word-word co-occurrence in the top ranked documents retrieved for the query. The underlying hypothesis

is that words which co-occur with all query words in the top ranked documents are useful for characterizing the information need.

The words that are identified by these techniques are explicitly added into the original queries. One advantage of explicit query expansion is that it provides the users with a chance to alter the expanded queries if the added words are not desirable. The other advantage is easy "debugging": by simply looking at the expanded queries we can often tell what is wrong with a certain technique. This is very useful for research.

We will apply the techniques for text analysis in two domains of query expansion:

- Stemming, which involves adding to the original queries the morphological variants of the query words.

- General query expansion, which goes beyond word variants for query expansion. Words used for query expansion are not required to be morphologically related to the query words.

The application of these text analysis techniques attempts to address the following problems with the existing approaches to the word mismatch problem:

- Human labor.

  Traditional stemmers require a considerable amount of human labor to develop. Linguistic experts need to synthesize the necessary morphology rules. Computer programmers need to code them. Many IR systems incorporate exception lists to handle words that are exceptions to the morphology rules. Additional human labor is required to compile these exception lists. We will use corpus analysis to save human labor in developing stemmers in two ways. One is to automatically create new stemmers. The other is to automatically identify exception words.

  Manual thesauri require expensive human labor to compile. Relevance feedback requires human relevance judgment. In comparison, our text analysis techniques to identify expansion words are completely automatic.

- Lack of effectiveness.

  Traditional stemmers have a number of problems which can result in incorrect conflations. They are not adaptable to corpus specific meanings of word variants. For example, the meaning of the word "stocks" in Wall Street Journal articles is different from its meaning in a history book. Traditional stemmers may not use all the morphology rules. Even if they use all the rules, there are exceptions to the rules. In this thesis we will use corpus analysis to improve stemming by identifying related and unrelated word variants.

  Automatic thesauri based on corpus analysis have not been consistently successful in improving retrieval effectiveness. In this thesis we will investigate why corpus analysis is not effective for general query expansion.

  Local feedback has enjoyed more success than automatic thesauri. This technique, however, is not robust and can cause serious retrieval failure when its assumption about relevance of the top ranked documents is violated. In this thesis, we will use local context analysis to overcome the problem with local feedback and achieve more robust and effective retrieval.

- Inefficiency.

  Automatic thesauri construction and dimensionality reduction are computationally costly. Automatic thesauri construction needs to collect the co-occurrence information for every pair of words in a corpus. Dimensionality reduction techniques typically need to decompose a huge matrix which is comparable in size to the corpus. As text collections are getting increasingly larger, these techniques become more impractical. In comparison, local context analysis is applied only to the top ranked documents and is efficient and practical even on large collections.

  Collection size is mainly to blame for the inefficiency of automatic thesauri construction and dimensionality reduction techniques since they rely on the information in the whole corpus. One research objective of this thesis is to

determine whether such global information is necessary or even desirable for effective query expansion.

We have the following hypotheses regarding the application of the text analysis techniques in stemming and general query expansion.

1. Corpus analysis can help identify related and unrelated word variants specific to a corpus. A traditional stemmer enhanced by corpus analysis is more suited for the corpus being searched, makes fewer mistakes, adds fewer word variants to the original queries and improves retrieval effectiveness and efficiency.

2. It is possible to automatically create a new stemmer solely based on corpus analysis and simple initial character matching. We use initial character matching as a simple stemmer and enhance it by corpus analysis. The resulting stemmer can compete with traditional stemmers in terms of retrieval effectiveness.

3. Corpus analysis has limited value for general query expansion. In general, the words added by corpus analysis should be related to the query words. But since individual query words are often ambiguous or too general, the added words may not be related to the query words in the context of the queries.

4. Global information is not necessary for effective query expansion. The information in the top ranked documents is most important.

5. Local context analysis can achieve more effective retrieval than local feedback and corpus analysis techniques.

6. Local context analysis is computationally practical. Both the one time cost for a collection and the cost of query expansion for a query should be relatively low.

The main results that are demonstrated are:

- Corpus analysis significantly reduces the number of incorrect word conflations made by traditional stemmers. When corpus analysis is used to enhance a traditional stemmer, the enhanced stemmer avoids most of the incorrect conflations made by the original stemmer, reduces the average number of words in the expanded queries by 50% or more and slightly improves retrieval. This is desirable in interactive applications where response time is important and stemming decisions are visible to end users.

- Experiments on both English and Spanish collections show that a simple initial character matching algorithm enhanced by corpus analysis is comparable with traditional stemmers in terms of retrieval effectiveness.

- Query expansion based on corpus analysis achieves a small improvement in retrieval effectiveness on a large collection.

- By requiring expansion words to co-occur with all query words, corpus analysis achieves large improvements in retrieval effectiveness on two collections. Further experiments show that the performance largely depends on the top ranked documents. When applied on low ranked documents, it essentially does not work.

- Local context analysis achieves significant improvements in retrieval effectiveness on a wide variety of collections of different characteristics and different languages. Experimental results show that local context analysis is more effective than corpus analysis and local feedback.

- Building a local context analysis database on a 2.0 GB collection takes 4 hours, 1.5 GB of disk space and 200 MB of memory. Query expansion takes 2 seconds per query and 240 MB of memory. The time and disk usage are small enough to be practical. The current implementation is built for flexibility but it can be easily made more efficient with regard to memory usage.

Our techniques assume that query words and their alternatives have some chance to co-occur in a collection. Although the experiments demonstrated that the assumption holds on the test collections used in this thesis, it is not hard to come up with cases where the assumption is violated. On some collections, the authors of the documents may never use the words used by some users in their queries due to difference in things such as personal backgrounds and professions. For example, a user who describes his information need as "legislations about the state of black America" to search a collection of congressional bills will meet with retrieval failure because politicians never use words such as "black America". Our techniques will not work in such cases if query expansion and retrieval are carried out on the same collection. One possible approach awaiting investigation in future work is to expand a query on a different collection than the one being searched. We expect that some collections, e.g. newspapers, will be more suitable for query expansion because documents in them are more likely to use alternative words to describe the same concepts. Potential benefits and problems of such a cross corpora approach remain to be determined in future work.

## 1.4 Research Contributions

- We establish that automatic corpus analysis based on co-occurrence information can help to identify related and unrelated word variants specific to a corpus. A stemmer enhanced by corpus analysis is adaptable to the corpus being searched, significantly reduces the amount of expansion, makes fewer mistakes and improves retrieval effectiveness.

- We propose a retargetable stemming algorithm based on simple initial character matching and corpus analysis. The algorithm can be easily ported to a new language and offers retrieval performance competitive with the traditional stemmers.

- We establish that global corpus analysis has limited value for general query expansion. We also establish that contrary to widely held belief, query expansion based on global co-occurrence information and query context has been successful almost entirely due to the information in the top retrieved documents.

- We establish that analysis of the top ranked documents is more effective and more efficient for query expansion than global corpus analysis. Specifically, we proposed a query expansion technique, called local context analysis, which outperforms global query expansion techniques in terms of effectiveness and efficiency.

- We demonstrate that local context analysis is more effective and more robust than local feedback because local context analysis does not make strong assumptions about the relevance of the top ranked documents.

## 1.5 Outline of Thesis

The remainder of this thesis is organized as follows: Chapter 2 reviews related work. Chapter 3 describes the test collections used in this thesis and the evaluation metrics. Chapter 4 presents the techniques of text analysis in detail. Chapter 5 and 6 present experimental results of applying corpus analysis to stemming and general query expansion. Chapter 7 presents experimental results of local context analysis. Chapter 8 discusses the conclusions and future work.

# Chapter 2

## Related Research

In this chapter we review the research related to this thesis. We discuss stemming in Section 2.1, manual thesauri in Section 2.2, automatic thesaurus construction in Section 2.3, document clustering in Section 2.4, dimensionality reduction in Section 2.5, local clustering and local feedback in Section 2.6.

### 2.1 Stemming

In information retrieval systems, stemming is routinely used to reduce different word forms to common roots. The purpose of stemming is to retrieve documents which contain morphological variants of the query words. Stemming is a form of query expansion which adds morphological variants of the query words to the original query.

English stemming is traditionally carried out by stripping off the common suffixes. Two widely used stemmers, the Porter stemmer [46] and the Lovins stemmer [35] are examples. The Lovins stemmer simply removes the longest suffix of a word. The Porter stemmer is somewhat more complex. It iteratively removes endings from a word according to a set of rules until no more can be removed. The major problem with this type of morphologic stemmers is that they make mistakes because they do not pay attention to the meanings of the words.

To address the problem with the morphologic stemmers, Krovetz devised a new stemmer, now called KSTEM, which not only makes use of morphologic rules, but also semantic relationships between words obtained from a machine readable dictionary [31]. Although KSTEM prevents many mistakes made by the morphologic stemmers, retrieval effectiveness using it is not consistently better than using the Porter stemmer.

KSTEM is too conservative in conflation. For example, it does not stem "stocks" to "stock" because "stocks" may mean something different from "stock". While the separation is good in some cases, it definitely hurts retrieval on a collection of Wall Street Journal articles.

Though some researchers have questioned the value of stemming for information retrieval in the past [23], more recent studies have demonstrated that stemming can produce consistent though small improvements in retrieval effectiveness over a large range of collections [31, 28]. Krovetz also showed that stemming is more useful for short queries and short documents.

Some studies show that stemming is also useful for retrieval in languages other than English. Popovic and Willet [45] found that stemming improved retrieval on a Slovene collection. Kraaij [30] found that stemming helps retrieval in Dutch.

## 2.2 Manual Thesauri

Generally speaking, manual thesauri describe the synonymous relationship between words, though many thesauri use finer grained relationships such as Broader Terms, Narrow Terms and so on. The synonyms in the manual thesauri can be used for query expansion. Some of the thesauri such as UMLS [65] and MeSH (Medical Subject Headings) [37] are for specific domains while others such as WordNet [39] are for general purpose.

Building manual thesauri requires a lot of human labor from linguists or domain experts. Though they are expensive to build, query expansion using manual thesauri has not been successful in improving retrieval effectiveness. Salton and Lesk obtained some improvement in retrieval effectiveness by adding synonyms selected from a manual thesaurus but the improvement was too small to have any practical value [57]. A recent study by Voorhees on the large TREC collections using WordNet even produced negative results [69]. Generally speaking, manual thesauri do not consistently improve retrieval performance.

Salton proposed a method to construct a thesaurus based on relevance judgments for a large number of training queries [53]. The underlying idea is simple: given a matrix $A$, which represents the relevance and non-relevance information for each document-query pair, a similarity function $F$ and a cut off value $T$, there must exist an optimal non-overlapping clustering of the terms in the sense that after replacing each term by its cluster, the retrieval matrix $B$ is closest to $A$. To find the optimal clustering, the straightforward approach is to try all possible clustering choices. But it is obviously impractical given the explosive number of choices. Salton suggested some heuristic methods to find good suboptimal solutions. We should point out this method of thesaurus construction is designed to optimize retrieval effectiveness rather than capture the semantic relationship between words. The major disadvantage with this method is that it requires a large and comprehensive set of training queries. The cost of relevance judgments for so many queries is prohibitive.

## 2.3 Automatic Thesaurus Construction

Automatic thesaurus construction is an extensively studied area in information retrieval. The original motivation behind automatic thesaurus construction is to find an economic alternative to manual thesauri. Almost all automatic thesauri are based on the so called association hypothesis, which states that words related in a corpus tend to co-occur in the documents of that corpus [67].

Term clustering represents the earliest work on automatic thesauri construction. It groups related terms into clusters based on their co-occurrence in a corpus. The most representative work on term clustering was conducted by Sparck Jones in the late 60's and early 70's [62, 60]. We briefly summarize Sparck Jones' experiments and her conclusions. In her experiments, she investigated different ways to form clusters and different ways to use the clusters. The variations in her experiments were:

- Clustering all terms vs. clustering only infrequent terms. In term of retrieval effectiveness, the latter is better than the former.

- Different similarity metrics: cosine metric, weighted and unweighted Tanimoto. There are no significant differences in retrieval effectiveness using different similarity metrics.

- Different class formation methods: strings, stars, cliques and clumps. Different methods achieved comparable retrieval effectiveness. The important factor is the size and tightness of the clusters: small strongly connected clusters are better than large loosely connected ones.

- Query formulation: simple substitution of query terms with clusters is better than the combination of substitution and the initial query, though the latter is supposed to be able to combine the merits of the two representations.

Her major conclusion was that well constructed term clusters can improve retrieval performance. We should point out that her experiments were based on a small test collection and used a rather primitive retrieval system. The test corpus had only 200 documents. The retrieval system used coordination level match (number of terms a query and a document have in common) and did not weight query and document terms. Her conclusions may not apply to other collections using modern retrieval systems. To further test the value of term clustering, Minker, Wilson and Zimmerman [40] applied similar term clustering techniques on two other collections. They used unweighted Tanimoto similarity, connected component and clique cluster formation methods, different threshold values to control the class size and a weighting scheme to weight the terms added to the initial query. They used SMART, a more sophisticated retrieval system which weights terms in queries and documents by frequencies. After a thorough investigation, they concluded that term clustering does not improve overall retrieval effectiveness. A major difference between Sparck Jones's experiments and Minker's is that the former optionally clustered only infrequent terms while the latter clustered all. It is fair to say that early work on term clustering was not very successful.

Peat and Willet [42] attributed the failure of term clustering to the bad statistical characteristics of the clusters generated based on co-occurrence data: terms of comparable collection frequencies tend to be in the same cluster because of the mathematical properties of similarity metrics used. Because query terms tend to be frequent ones, the expansion terms also tend to be so. Since frequent terms tends to be bad discriminators in separating relevant and non-relevant documents, using them for query expansion is likely to degrade the original query. This also explained why clustering only infrequent terms was better than clustering all terms in Sparck Jones' experiments. We think their argument is too simplistic to reveal the true causes of the failure of term clustering. Firstly, a bad expansion term hurts retrieval because it is unrelated to the topic of the query, not because it is frequent. Secondly, most query terms are mid frequency terms, not high frequency ones. For information retrieval, they are usually the most useful terms [54]. The similarity metrics were indeed a problem in Sparck Jones and Minker's work: the metrics used were not properly normalized for frequent terms. For example, two independent terms each of which occurs in 50% of the documents will have a Tanimoto similarity value 0.5. A more serious problem is that term clustering regards whether two terms are related to be a binary property: two terms are either related or not related. But the reality is that two terms are usually related to some degree. The most serious problem with term clustering is that it does not take into account the ambiguity of the query terms. If a query term has several meanings, term clustering may add terms related to the incorrect meanings of the term. As the result, the expanded queries may contain more ambiguity than the original queries and produce poor retrieval.

Lesk [32] proposed a revised version of the association hypothesis. The revised version is based on the observation that synonyms usually do not co-occur directly, but co-occur with the same terms. The revised hypothesis, called second order association hypothesis, states that if two terms are related, they tend to co-occur with the same terms. To make a distinction, we call the original hypothesis the first order association

hypothesis. Some recent studies on query expansion also exploited the second order association hypothesis [58, 52, 21].

Lewis investigated the use of phrase clustering in both traditional IR and text categorization tasks [34]. His experiments did not produce positive results. Other attempts to make use of syntactic information in clustering were made by Ruge [52] and Grefenstette [21].

To address the ambiguity problem of query terms, some recent techniques for thesaurus construction require the expansion terms to co-occur with all query terms. The representatives of these techniques are Qiu and Frei's Concept Based Query Expansion [47] and Jing and Croft's Phrasefinder [29]. The basic idea was also used in Sparck Jones' work on term clustering in which she optionally used terms co-occurring with more than one query terms for query expansion [60]. While the early application of the idea did not work well, the more recent techniques extended it and achieved better results. In Qiu and Frei's work, an association thesaurus for a corpus is built in two steps:

1. Build a term-document matrix $T$. $T_{ij}$ denotes the association significance between document $j$ and term $i$. It is computed by normalizing the number of occurrences of term $i$ in document $j$ over the length of document $j$.

2. The association thesaurus is defined as the matrix $S = TT^t$, where $T^t$ is the transpose of $T$. $S_{ij}$ denotes the similarity between term $i$ and term $j$.

In Qiu and Frei's work, a query is represented as a vector of weighted terms. The similarity between a term and a query is defined as the weighted sum of the similarity values between the term and individual terms in the query. To expand a query, terms with the highest similarities with the query are added to the query. The weight of each added term in the expanded query takes its similarity value with the original query. Qiu and Frei tested their approach on three small test collections and obtained

significant improvements in retrieval effectiveness. A major disadvantage with Qiu and Frei's technique is that its computational cost is very high.

Qiu and Frei's technique is very similar to a number of other techniques , e.g., Wong's GVSM [72] and Schutze's context vectors [58], in that they all assign weights to expansion terms based term-term similarities. The difference is that Qiu and Frei's technique only uses the terms with the highest similarities to the query. It is not just for efficiency. Terms with low similarity values to the query tend to be related to only one query term but not to the topic of the whole query. If such terms are used for query expansion, they will hurt retrieval even though they have relatively small weights in the expanded query.

In Jing and Croft's work [29], noun phrases, which are called *concepts*, are used for query expansion. Similar to Qiu and Frei's techniques, co-occurrences with all query terms are considered in choosing expansion concepts. Specifically, each concept $C$ is represented as a set of tuples $< t_1, a_1 >, < t_2, a_2 >, ...$, where $t_i$ is a term co-occurring with $C$ and $a_i$ is the number of co-occurrences between the concept and the term in a corpus. The set is called the pseudo-document of the concept. The same metric to calculate the similarity between a true document and a query is used to calculate the similarity between a pseudo-document and a query. The concepts whose pseudo-documents are most similar to the query are used for query expansion. Jing and Croft's technique is more efficient than Qiu and Frei's. Pseudo-documents are easier to compute than term-term similarity matrix. Document indexing techniques such as inverted list representation and data compression can be used to index pseudo-documents to save disk space. The major problem with Jing and Croft's work is that it treats pseudo-documents just like true documents although they have very different characteristics. The length of pseudo-documents conforms to Zipf's law [74]: a few are very long, but most are very short. The length distribution of true documents does not conform to Zipf's law. The *idf* of a term in a true document collection measures the probability that the term occurs in a randomly chosen document. It does not

depend on the size of the collection. The *idf* of a term in a pseudo-document collection measures the probability that a term co-occurs with a randomly chosen concept. It depends on the size of the original document collection: the more documents are in the document collection, the greater the chance that a term and a concept ever co-occur.

## 2.4 Document Clustering

Like term clustering, documents can also be clustered. Documents can be grouped into clusters of documents based on document-document similarity. The purpose of document clustering is two fold: search efficiency and search effectiveness. Some IR applications such as browsing, require similar documents to be physically close to each other. Document clustering is an efficient organization for these applications. We are more interested in the retrieval effectiveness of document clustering. Document clustering can enhance retrieval effectiveness when the so called cluster hypothesis holds: closely associated documents tend to be relevant to the same information request [67]. Document clustering can also be viewed as some sort of query expansion: clusters of documents contain more terms and have a greater chance to match the query terms than individual documents. There are many works on document clustering in the literature. Instead of describing all of them, we only summarize the major points based on Chapter 10 of [54].

First we talk about document cluster organization and generation. The organization of document clusters usually conforms to a tree-like structure: smaller and tighter clusters are at lower levels and larger and coarser ones at higher levels of the tree. The clusters are composite in that each cluster consists of several smaller ones except for the leaf clusters that are single documents. Each cluster has a representation, called its centroid. The centroid is a summary of the contents of the documents which are its offspring in the tree. To generate the tree-like structure, the straightforward methods start with the similarity information between all pairs of documents in the

collection. Usually the pairs must be sorted by the similarity values. This means that such methods run in $O(n^2 \log n)$, where $n$ is the number of documents in the corpus. With the information, clustering can proceed either top down or bottom up. The bottom up approach is also referred to as the agglomerative algorithm, because in each step, it agglomerates two closest related clusters into a larger one. There are three definitions of the "closeness" between two clusters: single link, complete link and average link. The single link similarity between two clusters is the similarity between the two most similar documents, one of which appears in each cluster. The complete link similarity is the similarity between the two most dissimilar documents, one from each cluster. The average link similarity is a compromise between the two. With the same similarity value, a single link cluster tends to be larger and looser than an average link one, which in turn tends to be larger and looser than a complete link one. There are also clustering methods which do not require prior knowledge of the similarities between all pairs of documents. Though such algorithms run in less than $O(n^2 \log n)$, they are not robust and the clusters generated using them do not have some of desirable properties possessed by the ones using the $O(n^2 \log n)$ methods.

Document retrieval based on cluster organization can be either top down or bottom up. The top down method uses a breadth first search to find clusters whose similarity to the query is greater than a specified threshold. The bottom up method only searches the lowest, non-singleton clusters to find those whose similarity to the query is greater then a specified threshold.

Crouch and Yang proposed a novel approach to term clustering using document clustering[15]. It is known that clustering of low frequency terms is most useful for retrieval purpose [60], but co-occurrence based techniques can not properly handle low frequency terms because there is little co-occurrence information about them. To address the problem, whether two terms are related is determined based on whether they co-occur in the same cluster of closed related documents. The complete link similarity is used to ensure that documents in the same cluster are closely

related. Though this approach improved retrieval effectiveness on the test collections, performance is largely dependent on the settings of the parameters in the experiments, e.g., the similarity threshold between documents in the same cluster, number of documents per cluster and so on. Because this approach requires document clustering, it is very expensive and can not easily be scaled up to large collections.

## 2.5 Dimensionality Reduction Techniques

If documents and queries are viewed as vectors in a $n$ dimensional space, term clustering can be viewed as a dimensionality reduction technique: before clustering, $n$ is the number of distinct terms in the corpus, and after clustering, $n$ is the number of clusters. Since different terms may be represented by the same axis in the reduced space, documents and queries have a greater chance to be matched. In this section we discuss several techniques which explicitly make use of dimensionality reduction. The difference of these techniques from term clustering is that in the reduced space, a term is decomposed into a number of components. Each component takes a weight to indicate its importance in the representation of the term. Different dimensionality reduction techniques have different motivations and are different in the ways that a term is decomposed. We will discuss Deerwester *et al*'s Latent Semantic Indexing (LSI) [16, 19], HNC's MatchPlus system [6, 7] and Schutze and Pedersen's Coocurrence Thesaurus [58].

Furnas in [19] gave a rationale for LSI and explained how it works. The author of a document usually chooses only a sample of plausible terms in describing the topic of the document. Many terms, though not in the document, could also be used to describe the topic of the document. The purpose of LSI is to "rediscover" these "missing" terms of the document. More specifically, we can represent a corpus or collection by a term-document matrix. The true term-document matrix describes the complete content of each document intended by its author, including the "missing" terms. The observed term-document matrix only describes what terms are actually

present in each document. The two matrices are different, but we could rediscover (or predict) the true based on the observed. The predicted matrix may be somewhat different from the true matrix but should be closer to the true matrix than the observed matrix. The mathematical tool to make the prediction is the factor analysis model, based on the singular value decomposition (SVD) of the observed matrix. Ignoring some of the technical details, we briefly summarize how LSI works. Let $S$ be the term-document matrix (observed), $S$ can be decomposed into the product of three matrices $X = TSD$, such that $T$ and $D$ have orthogonal columns and $S$ is diagonal. The SVD can be geometrically interpreted as a scheme to represent both documents and terms as vectors in a high dimensional space. Each dimension of the space is called a LSI factor, which has no intuitive meaning. Most of the diagonal elements of $S$ are very small and can be discarded as "noise". By keeping only the $k$ largest factors and truncating the rows or columns of $T$ and $D$ which correspond to the "noisy" LSI factors, we get $\overline{X} = \overline{TSD}$. With the "noise" purged, $\overline{X}$ should be a better representation of the corpus than $X$. Therefore, retrieval using $\overline{X}$ should be better than using $X$.

Despite the potential of LSI claimed by its advocates, the retrieval effectiveness on a number of test collections is not conclusively better than standard vector space retrieval systems. Dumais [17] pointed out some retrieval failures made by LSI. A major source of the failures is the lack of specificity of the retrieved documents. In the reduced space, everything can match anything to some degree. A document containing no query terms may be ranked high because it contains many terms which are somewhat related to the query terms. This may also be the major problem with all dimensionality reduction techniques. The other problem with LSI is its computational complexity. It is very costly to decompose a very large matrix. It is true that computer power is growing rapidly, but so is the size of the data. To make it practical on large collections, SVD has to be applied to subsets of documents at the price of losing some of the power of the method.

HNC's MatchPlus [6] retrieval system used a technique similar to LSI. Words (terms), queries and documents are represented as vectors in a $n$ dimensional space. The typical value of $n$ is several hundred. The vectors in the space are called *context vectors*. Each dimension of the space is called a *feature* . Unlike LSI factors which have no intuitive meanings, these features are intended to be the basic units of semantics. In fact, some manual selected features even have names like "agriculture", "DNA", "human" and so on. The context vectors of the words are automatically learned from the collection using machine learning techniques. The context vector of a query or a document is the centroid of the context vectors of the words in the query or in the document. Retrieval is carried out by computing and ranking the dot products between context vectors of the documents and the context vector of the query.

A problem with HNC's context vector approach is that it can not properly handle words with multiple meanings. For example, the context vector of "windows" likely will consist of "computer", "electronics", "building" and so on. This will hurt retrieval precision because of possible spurious query-document matches.

Schutze and Pederen in [58] proposed a slightly different approach from MatchPlus. They also used context vectors to represent words, documents and queries. Retrieval is also based on similarities between query and document context vectors. But there are major differences between their work and HNC's MatchPlus. The first difference is how the context vectors are derived. Conceptually, a context vector of a word in Schutze's work is a row in a $n \times n$ matrix $T$, where $n$ is the number of words in the collection and $T_{ij}$ is the number of co-occurrences of word $i$ and word $j$ in the corpus. Since $n$ is very large, $T$ is very costly to compute and store. To save space, $T$ is single value decomposed and each context vector is reduced to a vector in a low dimensional space. To make the SVD faster, words are initially clustered. The other difference is that in Schutze's work, retrieval results using both standard $tf \times idf$ and context vectors are merged to produce a final retrieval result. The purpose is

to complement the precision power of $tf \times idf$ with the recall power of the context vectors.

## 2.6 Local Clustering and Local Feedback

Query expansion techniques based on information in a whole corpus is very costly if the corpus is very large. In this section we discuss query expansion techniques based on the documents retrieved for the original queries. Because the problem size is reduced, these techniques are far more efficient.

One of the techniques, which we call local clustering, was proposed by Attar and Fraenkel [2]. Query expansion using this technique consists of three steps: (1) Perform an initial search for a query and retrieve a set of documents. (2) Cluster the terms in the retrieved documents. (3) Expand the query using the term clusters and perform a second search. Attar and Fraenkel produced positive improvement in retrieval effectiveness, but the test collection they used is too small to draw any definite conclusion.

The other technique is local feedback. Local feedback is just like relevance feedback, except that the top retrieved documents are assumed to be relevant without relevance judgment. Common terms from the top ranked documents are added to the original query. If a significant percentage of the top ranked documents are relevant, local feedback will improve results much like relevance feedback. The major problem with local feedback is that if few top retrieved documents are relevant, local feedback can significantly hurt retrieval effectiveness. A number of groups participating in the TREC conferences used local feedback and obtained significant improvements in retrieval effectiveness [5]. A related technique using information from the top ranked documents for query modification was proposed by Croft and Harper [12]. That technique changes weights of the original query terms but does not add new terms.

# Chapter 3

## Evaluation Issues

In this chapter we discuss issues related to the evaluation of the techniques in this thesis. We will describe the test collections in Section 3.1 and effectiveness measures in Section 3.2.

### 3.1 Test Collections

We use standard test collections to evaluate the retrieval effectiveness. Each test collection consists of a set of documents, a set of queries and a relevance judgment file. The relevance judgment file contains information to determine whether a document is relevant to a query. The test collections used in this thesis are WSJ, WSJ91, WEST, TREC3, TREC4, TREC5, TREC4-SPANISH, TREC5-SPANISH, TREC5-CHINESE and TIME. Table 3.1 lists statistics about these test collections.

- WSJ–This collection consists of 5 years (1987-1991) of Wall Street Journal news articles. The query set has 66 queries, which are chosen from Tipster (TREC) topics 1–100 such that each query has more than 50 relevant documents in the collection. Documents are in English. Past research in stemming showed that queries with few relevant documents can cause unstable retrieval results and make the evaluation difficult [28]. We choose queries with enough relevant documents to avoid the problem.

- WSJ91–This collection consists of 1 year (1991) of Wall Street Journal. The query set has 60 queries, which are chosen from the 66 WSJ queries such that each query has more than 10 relevant documents in WSJ91. Documents are in English.

- WEST–This collection consists of legal documents and is provided by the WEST Publishing Company. Documents are in English. The WEST query set has two versions. One, called the natural language query set, treats a query as a collection of individual words. The other, called the structured query set, uses phrase and proximity operators to structure the combinations of words.

- TREC3–This is the test collection for the English ad hoc queries in the third Text Retrieval Conference [24]. It is 2.2 Gigabytes large and consists of Wall Street Journal 1987–1992, Associated Press Newswire 1988-1989, Department of Energy abstracts, Federal Register 1988-1989, and Ziff Davis Computer-Select articles. The query set has 50 queries. Documents are in English.

- TREC4–This is the test collection for the English ad hoc queries in the fourth Text Retrieval Conference. It is 2.0 Gigabytes large and consists of Wall Street Journal 1990–1992, Associated Press Newswire 1988 and 1990, Federal Register 1988, Ziff Davis Computer Select articles, San Jose Mercury News articles 1991, and U.S. Patents articles 1993. The query set has 49 queries. Documents are in English.

- TREC5–This is the test collection for the English ad hoc queries in the fifth Text Retrieval Conference. It is 2.2 Gigabytes large. Content is similar to that of TREC3 and TREC4. Documents are in English.

- TREC4-SPANISH–This is the test collection for the Spanish ad hoc queries in the fourth Text Retrieval Conference. It has 25 queries. Documents are Spanish newspaper articles.

- TREC5-SPANISH–This is the test collection for the Spanish ad hoc queries in the fifth Text Retrieval Conference. It has 25 queries. Documents are Spanish newspaper articles.

- TREC5-CHINESE–This is the test collection for the Chinese ad hoc queries in the fifth Text Retrieval Conference. It has 19 queries. Documents are Chinese newspaper articles.

- TIME–This collection consists of 423 articles from TIME magazine. It has 83 queries.

Table 3.1. Statistics on test collections. Stop words are not included. Each Chinese character is counted as a word.

| collection | query count | size (GB) | document count | words per query | words per doc | rel docs per query | word count in collection |
|---|---|---|---|---|---|---|---|
| WEST | 34 | 0.26 | 11,953 | 9.6 | 1967 | 28.9 | 23,516,042 |
| WSJ | 66 | 0.5 | 163,092 | 37.5 | 273 | 144 | 44,495,324 |
| WSJ91 | 60 | 0.15 | 42,652 | 35 | 291 | 55.6 | 12,418,568 |
| TREC3 | 50 | 2.2 | 741,856 | 34.5 | 260 | 196 | 192,584,738 |
| TREC4 | 49 | 2.0 | 567,529 | 7.5 | 299 | 133 | 169,682,351 |
| TREC5 | 50 | 2.2 | 524,929 | 7.1 | 333 | 110 | 174,731,151 |
| TREC4-SPANISH | 25 | 0.2 | 57,868 | 5.3 | 284 | 88 | 16,427,826 |
| TREC5-SPANISH | 25 | 0.34 | 172,952 | 8.2 | 156 | 100 | 26,907,847 |
| TREC5-CHINESE | 19 | 0.17 | 164,779 | 21 | 411 | 73.6 | 67,720,598 |
| TIME | 83 | 0.0015 | 423 | 7.4 | 323 | 3.9 | 136,975 |

Until recently, IR experiments were carried out on very small test collections. The use of small collections suffers from a serious problem: The retrieval effectiveness observed in the experiments may not reflect the true effectiveness of the system applied to real world text collections. Most of our test collections are reasonably large. Some of them, e.g, TREC3 and TREC4, are close to realistic text collections in size. Therefore we largely avoid the above problem. Though most real world collections are large, some can be quite small. Therefore we also use the small TIME collection in evaluation.

It is well known that many IR techniques are sensitive to factors such as query length [71], document length [59], query structure [31] and so forth. For example, one technique which works very well for long queries may not work well for short queries. To ensure that our techniques and conclusions are general, we use a large number of test collections of different characteristics. The TREC3 queries are very long, averaging 34.5 words per query. The TREC4-SPANISH queries are very short, averaging only 5.3 words per query. The WEST documents are very long, about 7 times as long as the TREC4 documents on average. The TREC3 queries have far more relevant documents than the WEST queries. Some of the collections, e.g, WEST and WSJ, are homogeneous in that they contain documents of similar types and similar content. Some of them, e.g., TREC3 and TREC4, are non-homogeneous and contain documents of different types, different content, different lengths and different sources. The collections are in different languages: English, Spanish and Chinese. We use both natural language and structured queries on WEST to evaluate stemming algorithms.

## 3.2 Effectiveness Measures

### 3.2.1 Recall and Precision

Recall and precision are two widely used metrics to measure the retrieval effectiveness of an IR system. Suppose we have a collection C, a query Q and a retrieval system S. C contains $R$ relevant documents to Q. S retrieves a set of $n$ documents for Q and $r$ of the $n$ retrieved documents are relevant to Q. The recall and precision for Q on C are defined as

$$recall(S) = r/R$$

$$precision(S) = r/n$$

In other words, recall is the proportion of the relevant documents that the retrieval system found and precision is the proportion of documents that the system found that are relevant. But most modern IR systems do not output a set of documents

for a query. Instead, they output a list of documents ranked in descending order of probability of relevance to the query. To get around this problem, we calculate a recall/precision pair for each relevant document in the ranked output and then interpolate to find the precision at standard recall points 0.0, 0.1, 0.2, ... 1.0. Precision for recall point 0.0 is the precision for the first relevant document in the ranked output. Often the average of the precision values at the standard recall points is used as a single metric to measure the quality of the ranked output. It is called the 11 point average precision. When a set of query is considered, the precision at a recall point for the query set takes the average of the precision values of all queries at that recall point. The average precision for the query set takes the average of the average precision values of all queries.

When we report the retrieval effectiveness of a method, we use a table showing the precision values at the standard recall points and the average precision. When several methods are compared, the table also shows the percentage of improvement of a method over the base line method. For example, Table 3.2 compares the retrieval effectiveness of two techniques X and Y for 50 queries on a collection C.

Table 3.2. Retrieval performance of X and Y on collection C

| | Precision (% change) – 50 queries | | |
|---|---|---|---|
| Recall | X | Y | |
| 0 | 82.2 | 85.3 | (+3.8) |
| 10 | 57.3 | 65.1 | (+13.5) |
| 20 | 46.2 | 54.7 | (+18.5) |
| 30 | 39.1 | 46.8 | (+19.9) |
| 40 | 32.7 | 40.0 | (+22.1) |
| 50 | 27.5 | 34.6 | (+25.9) |
| 60 | 22.6 | 28.4 | (+25.2) |
| 70 | 18.0 | 23.0 | (+27.3) |
| 80 | 13.3 | 17.4 | (+30.7) |
| 90 | 7.9 | 10.7 | (+34.4) |
| 100 | 0.5 | 0.7 | (+36.9) |
| average | 31.6 | 37.0 | (+17.0) |

Besides precision figures averaged over a query set, we want to know whether a technique is robust. A robust technique improves many queries and rarely hurts a query substantially. A non-robust technique, however, may significantly improve a few queries but cause large degradations in the others. Obviously, robust techniques are preferred over non-robust techniques. Since we usually can not determine whether a technique is robust based on averaged precision, we sometimes perform query by query evaluation for that purpose.

### 3.2.2 Significance Tests

Like in any scientific experiment, the outcome of an IR experiment is affected by random errors. As the result, we can not conclude that one technique is better than the other based on a small performance difference between two techniques. Significance tests are needed to decide whether the performance difference between two techniques is statistically significant. The paired *t-test* and *sign-test* are the most widely used in IR.

The general idea behind the tests is: we assume that two techniques being compared are equally good. Under the assumption, we calculate a probability (p_value) that the observed performance difference could occur by chance. The smaller is the p_value, the less likely that the two techniques are equally good and the more significant is the difference. If the p_value is very small, i.e. lower than a threshold $\alpha$, we reject the original assumption and conclude the two techniques are indeed different. In this thesis, we use $\alpha = 0.05$. In other words, if the p_value $< 0.05$, we conclude that one technique is better than the other.

Suppose we want to compare two techniques X and Y on $n$ queries $Q_1$, $Q_2$, ..., $Q_n$. We define the random variables

$D_i$ = average precision of query $Q_i$ using Y - average precision of $Q_i$ using X

The t-test assumes that $D_1$, $D_2$, ... $D_n$ are $n$ independent random variables following the same normal distribution with 0 mean and unknown variance. It can

be shown that under that assumption the random variable $T$ defined below follows the t-distribution with $n-1$ degrees of freedom:

$$T = \frac{\overline{D}}{\sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(D_i - \overline{D})^2}/\sqrt{n}}$$

$$\overline{D} = \sum_{i=1}^{n} D_i$$

The p_value is therefore

$$p\_value = 1.0 - F(n-1, T)$$

where $F$ is the cumulative distribution function of the t-distribution with $n-1$ degrees of freedom.

The sign test assumes that $P(D_i > 0) = P(D_i < 0) = 1/2$. In other words it assumes that the number of queries for which Y is better than X is a random variable which follows the binomial distribution with $n$ trials and success rate 0.5. Let $m$ be the number of queries for which Y is better than X. When $n$ is large enough (>20), the random variable $(2m-n)/\sqrt{n}$ closely follows the standard normal distribution. Therefore we can calculate

$$p\_value = 1.0 - G((2m-n)/\sqrt{n})$$

where $G$ is the cumulative distribution function of the standard normal distribution.

Definitions of the distributions and related proofs can be found in statistics textbooks (e.g. [48]).

Both tests have pros and cons. The t-test is more powerful in that it can detect smaller differences between two techniques. But it makes a strong assumption about the form of the distribution of the data involved. The sign test makes a weaker assumption about the data but is less powerful. In this thesis we use the t-test. Since

the data in the experiments may not strictly follow the normal distribution assumed by the t-test, caution should be taken in interpreting the results of the significance tests in this thesis. But as Hull pointed out in [27], the problem is not very serious and the t-test results generally hold even when the data do not strictly follow the normal distribution.

We should not confuse statistical significance with significance from the users' perspective. A positive statistical test only means the observed performance difference between two techniques are unlikely to occur by chance. A powerful test like the t-test can detect a very small (e.g 1%) improvement in average precision if a query set has many queries. Though statistically significant, such a small improvement in average precision is unlikely to have any impact on a user. As an informal rule among IR researchers, a 5% improvement in average precision is considered to be significant and a 10% improvement very significant from the users' perspective [61].

# Chapter 4

## Techniques for Automatic Text Analysis

### 4.1 Introduction

Past studies on the word mismatch problem focused on establishing and utilizing relationships among words. Word relationships are either lexical-semantic or statistically derived. Both types of word relationships have been extensively studied.

Lexical-semantic word relationships are usually represented in the form of thesauri. Some thesauri such as WordNet [39] are general purpose, while others such as MeSH [37] and UMLS [65] are for specific domains. Typical relationships used in the thesauri are Narrower Term (e.g. "workstation" is a narrower term of "computer"), Broader Term (e.g. "computer" is a broader term of "workstation"), Synonym ("aircraft" is synonymous with "airplane") and so on. The intention of using thesauri is to retrieve more relevant documents by adding words related to the words in a query.

In recent years, some knowledge based and logic based IR systems have used more sophisticated representations such as frames [70, 33], inference rules [63] and logic assertions [68, 18, 38] to describe domain knowledge. Since the central part of domain knowledge is about word relationships, these representations achieve basically the same function as thesauri.

The acquisition of lexical-semantic word relationships requires expensive human labor. To overcome this problem, some techniques attempt to establish word relationships based on statistics about word co-occurrences. Work in this area has been alternatively known as term clustering [67], keyword classification [62], association thesauri [29] and so on. Most of these techniques are based on the association hypothesis discussed in Chapter 1. Also related to term clustering are dimensionality

reduction techniques such as Latent Semantic Indexing (LSI) [16]. Using LSI, word relationships are expressed implicitly: Words with similar occurrence patterns have similar representations in the reduced space.

Generally speaking, past research failed to demonstrate consistent improvements in retrieval effectiveness using either lexical-semantic or statistical word relationships [55, 69, 62, 40, 16]. We think the following problems are mainly to blame for this failure:

- Ambiguous words. If a query word has multiple meanings, query expansion using either a thesaurus or term clustering will add spurious words. For example, "tree" and "crop" may be added to the query "power plant", because "plant" is ambiguous.

- Query context. Past research incorrectly assumed that the relationship between two words holds independent of the queries and hence query words were expanded independently. We use examples to show the problem. The words "tax evasion", "computer fraud" and "battery" are obviously narrower terms for the word "crime". But for the query "gun control and crime", we do not want to add the above words to the query, because the presence of "gun control" in the query implies that good expansion words must be related to violent crimes, e.g., "murder" and "bank robbery". Similarly, if the query is "computer and crime", we do not want to add "murder" and "bank robbery" to the query. These examples show that word relationships are not query invariant. How to expand a word depends on other words in the query.

- Relevance. To improve retrieval effectiveness, words added to a query must have to do with relevance to the query. In other words, these words must be used in the relevant documents. We say that such words have predictive power of relevance for the query.

Past research assumed that if two words are related and one word has predictive power of relevance for a query, so does the other. Van Rijsbergen, for example, made such an assumption concerning statistical word relationships [66]. We use the query "DNA testing in trial of criminal cases" to show why the assumption is wrong. Two phrases "DNA profile" and "DNA sequence" are nearly synonymous (both are related to the query word "DNA"). But interestingly, "DNA profile" is used frequently in the relevant documents and "DNA sequence" is not. The reason is that journalists reporting crime events such as murder cases tend to use the word "DNA profile" while the authors of scientific work in biology and medicine tend to use the word "DNA sequence".

- Granularity of expansion. There is no agreement in past research on what a word is. Some researchers treated single terms as words. This can be a problem because phrases which should be treated as a whole are broken into several words. For example, it does not make sense to treat "South" and "Africa" in the query "sanction against South Africa" as two words and expand them separately. Other researchers treated phrases such as "South Africa" as words. This introduces another problem: Should we treat the whole query as a phrase and expand it as a whole or should we treat "sanction" and "South Africa" as two words and expand them separately? According to the above discussion of query context, treating the whole query as a phrase may be better. But the number of words increases exponentially as the word syntax becomes more complex. It is impossible for thesaurus based techniques to handle such complex phrases.

This thesis deals with two levels of query expansion: stemming and general query expansion. The two have a number of differences. Stemming involves adding morphological variants of the query words while general query expansion can add any words. Granularity is not a problem for stemming, because morphology is usually applied to one word at a time. Stemming is usually independent of query context:

Whether two words should be conflated to the same root usually does not depend on the query. Due to these differences, we will use different techniques for stemming and general query expansion.

For stemming, we will use *corpus analysis*. In this thesis, corpus analysis refers to the technique of identifying related words based on their co-occurrence information in a corpus. Since query context is a minor problem for stemming, we will ignore it for sake of simplicity.

For general query expansion, we adopt a query specific approach in order to overcome the above mentioned problems with past research. Our technique is called *local context analysis*. Local context analysis expands a query as a whole, and thus avoids the problems of query context and granularity. It selects expansion words from the top ranked documents retrieved for a query. Since the top ranked documents usually contain several words from the query, it largely avoids the problem of word ambiguity. Since the top ranked documents are more likely to be relevant to the query than ordinary documents, words chosen from them are more likely to be predictive of relevance than words chosen from ordinary documents. Local context analysis assumes that some of the top ranked documents are relevant. It regards the expansion words as additional sources of evidence in inferring the relevance of a document, and combines them with words in the original query under the inference network retrieval model [64] using the INQUERY retrieval system [9]. David Haines had a thorough discussion of the issue of incorporating relevance information in the inference network model [22]. We should point out that though local context analysis can be viewed as a technique to identify new sources of evidence for a query, it is not specific to the inference network model or the INQUERY retrieval system. It is a general technique which can potentially be used with retrieval systems based on other retrieval models (e.g. the vector space model).

In the rest of this chapter, we will discuss the above techniques in more detail. We will discuss corpus analysis in Section 4.2. The focus of that section is on the

application of corpus analysis to stemming. We will also discuss the application of corpus analysis to general query expansion, but the primary purpose of the discussion is to expose the problems. We will discuss local context analysis in Section 4.3.

## 4.2 Corpus Analysis

### 4.2.1 Why Corpus Analysis?

As we mentioned in Section 4.1, corpus analysis refers to the technique of identifying related words based on their co-occurrence information in a corpus. Query expansion using corpus analysis has been extensively studied for at least 30 years. In general, it has failed to produce consistent improvement in retrieval effectiveness. A number of factors provide the motivation for pursuing this technique further. These include:

- In the past, corpus analysis was applied on arbitrary pairs of words. The problem of determining the relationship between two arbitrary words may be too large to be satisfactorily solved by corpus analysis.

  One research objective of this thesis is to investigate the application of corpus analysis in stemming. Since we limit the use of corpus analysis to morphological variants, it may be more successful.

- Most studies in the past were conducted on small text collections. One widely suspected cause for the failure of corpus analysis in these studies is that the corpora used are too small to have sufficient statistics about the words [34, 10]. With the advent of large corpora such as the TREC collections, there is hope among IR researchers that corpus analysis may prove to be a viable technique.

- In recent years, there has been a revitalized interest in techniques that attempt to find word relationship based on co-occurrence information in a corpus. Two lines of research are particularly influential. One is the use of co-occurrence

information with all query words in selecting expansion words [47, 29]. This line of research has produced promising retrieval results. The other line of research is dimensionality reduction techniques such as LSI [16]. LSI makes use of sophisticated mathematical tools but is less successful in improving retrieval effectiveness than query expansion methods. One research objective of this thesis is to explain why these techniques work or do not work and what the problems are.

### 4.2.2 Corpus Based Stemming

#### 4.2.2.1 What is Corpus Based Stemming?

Stemming refers to a procedure to reduce different word variants to common roots. In IR systems, stemming is used in document indexing and query formulation so that documents containing the related variants of the query words are retrieved in addition to those containing the exact query words. Stemming is a special case of query expansion that expands a query word to all its variants.

English language stemming is traditionally carried out by stripping off the common suffixes based on morphologic rules. Two well known stemmers, the Porter stemmer [46] and Lovins stemmer [35] adopted such an approach. Another stemmer, devised by Krovetz [31], now known as KSTEM, is more complex. In addition to morphologic rules, it makes use of a machine readable dictionary to infer the semantic relationship between word variants.

One problem with traditional stemmers is that they can not handle corpus specific meanings of words such as "stocks" and "stock". "Stocks" is the plural form of "stock" in the Wall Street Journal, but its primary meaning in a corpus about medieval history may be a device to punish prisoners. The other problem is that traditional stemmers make mistakes when there are exceptions to the morphologic rules. For example, the Porter stemmer incorrectly stems "policy"/"police" and "addition"/"additive" to the same roots.

Stemming can be viewed as constructing equivalence classes among the words. Two words are in the same equivalence class if they are stemmed to the same root. Depending on the stemmer used, one such class may be, for example, "stock", "stocks", "stocked" and "stocking". In this thesis, we overcome the problems with traditional stemmers by modifying the equivalence classes based on corpus specific co-occurrence statistics about word variants. For example, in the Wall Street Journal, "stocks" and "stock" will co-occur frequently. As a result, we will put them in the same class. The Porter stemmer incorrectly puts "police" and "policy" in one equivalence class. Since "police" and "policy" rarely co-occur, we will put them in different classes. We will show that this produces more effective and efficient retrieval and makes fewer obvious errors from the end user's point of view. We label this approach of modifying equivalence classes based on corpus specific co-occurrence information *corpus based stemming*. The equivalence classes obtained from corpus based stemming can be used at document indexing time or at query time.

Hypothesis 1: Corpus analysis can help to identify related and unrelated word variants specific to a corpus. A traditional stemmer enhanced by corpus analysis is adaptable to the corpus being searched, makes fewer mistakes, adds fewer word variants to the original queries and improves retrieval effectiveness and efficiency.

Traditional stemmers require a considerable amount of human labor to develop and maintain. Since different languages have different morphologies, we have to write different stemmers for them. Our approach is to produce equivalence classes based on co-occurrence information and a simple heuristic about likely word variants. The heuristic is initial character matching: two words are likely to be word variants of the same root if they share initial characters. This heuristic should work for languages such as English and Spanish. For other languages such Arabic, we can use different heuristics.

Hypothesis 2: It is possible to automatically create a new stemmer solely based on corpus analysis and simple initial character matching. We use initial character

matching as a simple stemmer and enhance it by corpus analysis. The resulting stemmer should compete with traditional stemmers in terms of retrieval effectiveness.

#### 4.2.2.2 Co-occurrence Metrics

Since equivalence classes are modified based on co-occurrence, we need to define a metric to measure the co-occurrence strength between two word variants. To avoid problems with long documents, co-occurrences are determined using text windows of fixed size. Specifically, for two word variants $a$ and $b$, we define their number of co-occurrences $n_{ab}$ as the number of elements in the set $\{< a_i, b_j > | dist(a_i, b_j) < win\}$, where $a_i$'s and $b_j$'s are distinct occurrences of $a$ and $b$ in the corpus, $dist(a_i, b_j)$ is the distance between $a_i$ and $b_j$ measured using a word count within each document, and $win$ is the window size.

The metric used in this thesis is a variation of EMIM (expected mutual information measure) [67, 10]. EMIM is widely used to measure strength of associations. For word form co-occurrences, EMIM can be defined as

$$EMIM(a,b) = P(a,b) log_{10}(\frac{P(a,b)}{P(a)P(b)})$$

where $P(a,b) = n_{ab}/N$, $P(a) = n_a/N$, $P(b) = n_b/N$, $N$ is the number of text windows in the corpus, $n_a$ and $n_b$ are the number of occurrences of $a$ and $b$ in the corpus. The reason we choose not to use EMIM is that it is not normalized over the number of occurrences of $a$ and $b$ and unfairly favors high frequency word variants.

The metric we use, $em$, is defined as

$$em(a,b) = max(\frac{n_{ab} - En(a,b)}{n_a + n_b}, 0)$$

where $En(a,b)$ is the expected number of co-occurrences assuming $a$ and $b$ are statistically independent. The $em$ metric measures the percentage of the occurrences of $a$ and $b$ which are net co-occurrences (co-occurrences minus expected co-occurrences) and intuitively is more suitable for our purpose. Since a negative number does not make sense here, we make the $em$ score non-negative.

The role of $En(a,b)$ in the *em* formula is important, because two words may co-occur by chance. For example, the words "the" and "of" almost always co-occur in a reasonably large text window, but it is erroneous to conclude they are related. Our formula to calculate $En(a,b)$ is:

$$En(a,b) = kn_a n_b$$

where $k$ is a constant factor given the corpus and the window size. If we assume that a word never occurs in a text window more than once, we can calculate the expected co-occurrence using the formula

$$En(a,b) = P(a,b)N = P(a)P(b)N = \frac{n_a}{N}\frac{n_b}{N}N = \frac{n_a n_b}{N}$$

based on the independence assumption about $a$ and $b$'s occurrences and we can derive $k = 1/N$. Because in reality a word can occur in a text window several times and our method of counting co-occurrence counts multiple co-occurrences in a text window, we can not use the above formula to calculate $k$. However, $1/N$ provides a lower bound for $k$. In practice, we estimate $k$ based on the co-occurrence data for a large sample of randomly chosen word pairs by using the value $\sum n_{ab} / \sum n_a n_b$. Though any single pair of words may be biased, when many random word pairs are considered together, they should give a good estimation of $k$.

#### 4.2.2.3 Algorithms for Equivalence Class Refinement

Our approach to corpus based stemming is to break up equivalence classes generated using an aggressive stemmer or initial character matching. The purpose of using an aggressive stemmer or initial character matching is to make sure we will not miss any good conflations. Based on co-occurrence statistics, we detect the equivalence classes that contain unrelated word variants and break them up properly. We will use two algorithms for class refinement. One is the connected component algorithm. The other is the optimal partition algorithm.

**Connected component algorithm.** To refine an equivalence class, the class is mapped to a graph, each vertex representing a word and an edge between two vertices representing an *em* score greater than a threshold $t$. The vertices in each connected component of the graph form a new equivalence class. The threshold $t$ is a parameter in our experiments.

The connected component algorithm is efficient. The algorithm runs in $O(n^2)$ to refine a class of size n. But the connected component algorithm occasionally produces large sparsely connected equivalence classes, called "strings" in the clustering literature [54], which hurt the retrieval effectiveness.

**Optimal Partition algorithm**. Stemming may improve recall by retrieving more relevant documents. It may also decrease precision by retrieving non-relevant ones. The motivation behind the optimal partition algorithm is to achieve the best tradeoff between the two. More precisely, for any pair of words $a$ and $b$, we let the recall benefit of keeping $a$ and $b$ together be $em(a, b)$ because the higher the *em* score, the more likely that $a$ and $b$ are related and the conflation may improve recall. We choose a constant $\delta$ such that the conflation of $a$ and $b$ will not harm precision more than $\delta$. Thus the net benefit of keeping $a$ and $b$ together is $em(a, b) - \delta$ and the net benefit of separating them is 0. The net benefit of a partition of an equivalence class is the sum of the net benefits for all pairs of words in the class. An optimal partition of an equivalence class is one that maximizes the net benefit. If there are several optimal partitions, one of them is chosen arbitrarily to refine the initial equivalence class.

So far we have not found an efficient implementation of the optimal partition algorithm. Its similarity to some NP-complete optimization problems such as the Traveling Salesman Problem prompts us to guess that it may also be NP-complete. Our current implementation of the algorithm is a brute force search of all possible partitions. To make it feasible, we first apply the connected component algorithm to make the input classes smaller before we apply the optimal partition algorithm.

For a few large ones (containing more than a dozen words), we use the following approximation algorithm. The solution may be suboptimal, but should be practical for our purpose.

**Approximation of the optimal partition algorithm**. To refine an equivalence class, we start by assigning each word in it to a singleton class. At each step, we merge the pair of classes which have the largest cohesion. The process continues until either all words are in the same class or no two classes have a positive cohesion. We define the cohesion of two classes $C_1$ and $C_2$ as

$$cohesion(C_1, C_2) = \sum_{a \in C_1, b \in C_2} (em(a, b) - \delta)$$

where $\delta$ is the same as in the original optimal partition algorithm. The approximation algorithm is a slightly modified version of the average link clustering algorithm [54]. While the latter algorithm is interested in a hierarchical structure and will continue until all words are put in one cluster, our algorithm is interested in a proper partition of the words and therefore stops when further merging would result in undesirable clusters.

The approximation algorithm fails to make the optimal decision in cases where two strongly connected words $a$ and $b$ are associated with two otherwise unconnected classes $A$ and $B$. We can view that $a$ and $b$ form a bridge between $A$ and $B$. In such cases, the optimal decision will have $a$ and $b$ in two classes. The approximation algorithm will put $a$ and $b$ in one class.

### 4.2.3 Corpus Based Query Expansion

Now we discuss how to apply corpus analysis for general query expansion. The task is to use corpus analysis to determine the relationship between two arbitrary words.

The naive approach would be the same as we used for corpus based stemming. We could put all words in an initial equivalence class and refine it based on the

co-occurrence information between all pairs of words using either the connected component or optimal partition algorithm. We could then expand a query by replacing each query word with its equivalence class. This naive approach suffers one problem: by simply replacing query words with equivalence classes, we give all the expansion words and the original query words the same weight in the expanded query. This is not desirable for two reasons. Firstly, we want to weight the original query words more heavily because we have more confidence in them. Secondly, we want to weight the expansion words based on their co-occurrence strength with the original query words because how strongly they are related to the query words depends on the co-occurrence strength.

With the above considerations, we propose a nearest neighbors approach. We define the "nearness" between two words to be their *em* score. For a query word, we add its nearest neighbors to the original query. The central issue is how to use the nearest neighbors properly in query expansion to achieve effective retrieval. To cast the question in the context of the inference network retrieval model, what is the proper way to combine sources of evidence from a query word and its nearest neighbors? To simplify the discussion, we treat a word as a special nearest neighbor whose co-occurrence strength with itself is 1.0. Assume we have a document $t$, a query word $W_q$ and its nearest neighbors $N_i$'s, $0 \leq i \leq n$, $N_0$ being $W_q$ itself. In the inference network in Figure 4.1, node $t$ represents the event that $t$ is observed, node $N_i$ represents the event that $N_i$ is a correct indexing word for document $t$ and node $W_q$ represents the event that $W_q$ is satisfied. In the basic inference model used in INQUERY, $W_q$ would have only one parent, which is $N_0$. This means that for a query word to be satisfied by a document under the basic model, the query word itself must be a correct indexing word for the document. Figure 4.1 expands the basic model to allow $W_q$ to have multiple parents so that a query word can also be satisfied if its nearest neighbors are correct indexing words for the document. In the network, we assume that $N_i$'s are independent. Of course, a query word and its nearest neighbors

are unlikely to be truly independent. But as in many IR theories, independence is assumed to make the problem more tractable.

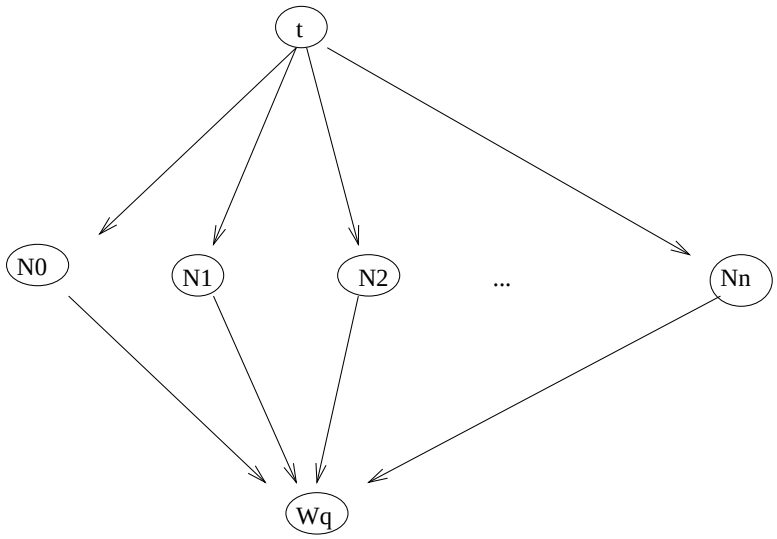


Figure 4.1. Represent nearest neighbors in inference network

Let $I = (x_0, x_1, ...x_n)$ be a truth assignment for the nodes $(N_0, N_1, ...N_n)$, where $x_i$ is either 1 or 0. $x_i = 1$ means $N_i$ is a correct indexing word for $t$ and $x_i = 0$ means $N_i$ is not a correct indexing word for $t$. Let $P(N_i|t)$ be the probability that $N_i$ is a correct indexing word for $t$. Since $N_i$'s are independent,

$$P(I) = \prod_{i=0}^{n} P(N_i|t)^{x_i}(1 - P(N_i|t))^{1-x_i}$$

Let $E_i$ be the event that $N_i$ causes $W_q$ to be satisfied. If we assume that $E_i$'s are independent, then the probability $W_q$ is satisfied given $I$ is

$$P(W_q|I) = 1 - \prod_{i=0}^{n}(1 - P(E_i))^{x_i}$$

By law of total probability, the probability that $W_q$ is satisfied by $t$ is therefore

$$P(W_q|t) = \sum_{\text{all possible } I\text{'s}} P(W_q|I)P(I)$$

It is straightforward to show that

$$P(W_q|t) = 1 - \prod_{i=0}^{n}(1 - P(E_i)P(N_i|t))$$

We will use the above formula to combine evidence from a word and its nearest neighbors. We call the formula a weighted OR operator because it is similar to INQUERY's OR operator except that each participating probability $P(N_i|t)$ is multiplied by a "weight" $P(E_i)$. INQUERY will estimate $P(N_i|t)$ using the standard $tf \times idf$ method. We will use $em(W_q, N_i)$ as an estimation for $P(E_i)$. With this arrangement, the contribution of a nearest neighbor to the score of a document depends on its co-occurrence strength with the original query words. In theory, we can use all words in a corpus for query expansion. Using the nearest neighbors is only an efficiency consideration. We label this approach of query expansion *corpus based* because the expansion of a query word depends on the corpus specific co-occurrence statistics.

The advantage of the corpus based approach is that it employs the statistical word relationship specific to the corpus being searched. One problem with this approach is that it is computationally costly. We need to collect the co-occurrence data for every pair of words in a corpus. Assuming the size of the text window used to determine co-occurrence is $k$ words and the size of the corpus is $l$ words, the time cost of collecting co-occurrence data is $O(kl)$. Assuming the number of word pairs that ever co-occur in the corpus is $r$, the space cost is $O(r)$. For large text collections, the space and time costs are enormous. Other problems, which were discussed in Section 4.1, are concerned with the value of statistical word relationships for retrieval. These problems severely limit the value of the corpus based approach.

Hypothesis 3: Corpus analysis has limited value for general query expansion. The words added by corpus analysis should be related to the query words in general. But since individual query words are usually ambiguous or too general, the added words may not be related to the query words in the context of the queries.

### 4.2.4 Corpus-query Based Query Expansion

In recent years, researchers have proposed the idea of combining corpus analysis and query context for query expansion [47, 29]. Instead of expanding each query word independently, the decision whether to use a word for query expansion is made based on its co-occurrence with all query words. The underlying hypothesis is that good expansion words should co-occur with all query words. We label this approach *corpus-query based* because the decision which words to be used for query expansion depends on both the corpus being searched and the query in question.

Like the corpus based approach, the corpus-query based approach needs the co-occurrence data between every pair of words in a corpus. Therefore the efficiency problem with the corpus based approach is also a problem with the corpus-query based approach. Unlike the corpus based approach, the corpus-query based approach has obtained marked improvements in retrieval effectiveness on a number of test collections [29].

The popular explanation for the success of the corpus-query based approach is that it employs the query context to disambiguate individual query words. A problem with the corpus based approach is that it may add words related to the incorrect meanings of the query words. The corpus-query based approach overcomes this problem by forcing expansion words to co-occur with all query words.

We have a different explanation for the success of the corpus-query based approach. Let us notice the fact that if we use a retrieval system to perform a search for a query $Q$, the common words from the top ranked documents will co-occur with many or all query words of $Q$ a number of times. Since the corpus-query based approach selects expansion words based on co-occurrence with all query words, this means that the common words from the top ranked documents have a very good chance of being used for query expansion. We know that techniques which add common words from the top ranked documents have achieved significant improvements in retrieval effectiveness [5]. Our conjecture is that the corpus-query based approach works largely

by indirectly finding words from the top ranked documents. The information in the top ranked documents is mainly responsible for the performance of the corpus-query based approach. If we only use the co-occurrence information outside the top ranked documents, query expansion using the words co-occurring with all query words would not be very successful.

Hypothesis 4: Global information is not necessary for effective query expansion. The information in the top ranked documents is most important.

## 4.3 Local Context Analysis

The bottom line of query expansion is to find representative words for the relevant documents. Research in relevance feedback suggests that such words are usually the common words (excluding stop words) in the relevant documents. Therefore we can reformulate the task of query expansion as finding common words from the relevant documents. Since the top retrieved documents are more likely to be relevant to the query than average documents, techniques that focus on the information in the top ranked documents should have a better chance of finding good expansion words than techniques based on corpus wide information such as corpus analysis. In this section we describe our technique, *local context analysis*, which expands a query by analyzing co-occurrence information in the top ranked documents.

Before we describe local context analysis, a brief discussion about a related technique, *local feedback*, is necessary. Local feedback assumes the top ranked documents are relevant and uses standard relevance feedback techniques to add new words to the original query and modify the weights of the query words. Usually the expansion words are the common words from the top ranked documents. Although local feedback can improve average retrieval effectiveness [5], its performance depends on the percentage of relevant documents in the set of top ranked documents used for feedback. If the percentage is very low, local feedback usually finds common words from non-relevant documents and seriously hurts retrieval effectiveness.

Local context analysis operates on a set of top ranked documents retrieved for a query. Rather than assuming all the documents in the top ranked document set are relevant, local context analysis makes a weaker assumption: a reasonable number of documents in the set are relevant. Unless a query is totally ill-formed or the collection has very few relevant documents for the query, this assumption holds if we make the set of top ranked documents relatively large.

The task of local context analysis is to find common words from top ranked relevant documents without relevance information. Before we discuss how local context analysis achieves this task, we make a number of observations about the top ranked documents.

- Due to the redundant words in typical queries, it is unrealistic to require that top ranked documents contain all query words.

  The purpose of using redundant words in a query is two fold. This first is to improve recall. To retrieve as many relevant documents as possible, alternative words are used to describe the same concept or subtopic in a query. For example, in the query "as a result of DNA testing, are more defendants being absolved or convicted of crimes?", the words "defendants" "absolved", "convicted" and "crimes" are alternative words to describe one subtopic of the query. The second is to improve precision. For example, if we remove the word "military" from the query "deaths caused by friendly fire or training accidents in the military" , a retrieval system will match the query with documents about a train accident in which a friend of somebody was caught in the fire ("friendly" and "training" become "friend" and "train" after stemming). The word "military" in the query helps to prevent such spurious matches.

  If a document contains all of the query words, it is very likely to be relevant. If we have enough such documents, query expansion is simple: We can simply use the common words from them for query expansion. But due to the redundant query

words, few documents, if any, will contain all of the query words. Therefore, we can not use this simple approach.

- Top ranked relevant documents tend to form a cluster.

  Because all relevant documents are about the same information need, they tend to use the same words. More precisely, for a given query, there is a set of words which will occur frequently in the relevant documents. Since the top ranked relevant documents are a sample of all relevant documents, such words will occur frequently in the top ranked relevant documents.

- Top ranked non-relevant documents often form clusters too.

  This phenomenon can be explained by the existence of *overlapping queries*. Two queries are overlapping if they share most of the query words but are about different information needs. Consider the example query "as a result of DNA testing, are more defendants being absolved or convicted of crimes?". One overlapping query might be "As the result of polygraph test, are more defendants being convicted or absolved of crimes?". The other one might be "Blood tests and defendants being convicted or absolved of crimes for drunk driving". If a query has one or more overlapping queries, we expect that the top ranked documents will consist of several clusters, each of them about a different query.

- Local feedback fails when top ranked non-relevant documents form clusters. More precisely, it fails when one non-relevant document cluster is larger than the relevant document cluster. In this case, the frequent words in the top ranked documents are, in fact, from the non-relevant document cluster.

- Common words from the top ranked relevant documents tend to co-occur with all query words in the top ranked documents.

  Because of the redundant query words, a relevant document may not contain all of the query words. We can model the occurrence of a query word in the

relevant documents as a stochastic process. In any given relevant document, a query word $w$ has a certain probability $p$ of being used. If we assume that $w$'s occurrence/absence in one relevant document is independent of its occurrence/absence in another, the probability that it occurs in at least one relevant document is $1 - (1 - p)^n$ when $n$ relevant documents are observed. If $n$ is reasonably large, the probability is close to 1. This means if a word occurs in a reasonably large number of top ranked relevant documents, it likely will co-occur at least once with any of the query words in the top ranked relevant documents.

- If a word in the top ranked documents is not a common word in the top ranked relevant documents, it is likely to co-occur with fewer of the query words in the top ranked documents.

  If a word does not occur, or occurs rarely in the top ranked relevant documents, its co-occurrences with the query words in the top ranked documents will mostly happen in the top ranked non-relevant documents. We consider two scenarios.

  Scenario 1: The top ranked non-relevant documents are random in content. In this case, most words in the non-relevant documents (except stop words) will occur only a small number of times, let alone co-occur with all query words.

  Scenario 2: There are clusters of top ranked non-relevant documents. As we discussed before, each cluster of non-relevant documents can be viewed as the documents retrieved for an overlapping query with the original query. Since an overlapping query misses some words in the original query, the documents retrieved for it will also miss some of them. In this case, even if a word occurs frequently in a cluster of non-relevant documents, it is unlikely to co-occur with all of the query words.

From the above discussion, it becomes clear that we can employ co-occurrence information in the top ranked documents for query expansion. Since only the common words in the top ranked relevant documents have the property of co-occurring with

all of the query words, we can conversely use the property to decide whether a word is a common word in the top ranked relevant documents.

Hypothesis: Common words in the top ranked relevant documents tend to co-occur with all of the query words in the top ranked documents. Query expansion using words co-occurring with all of the query words in the top ranked documents will produce more effective retrieval than local feedback.

Following the IR tradition, we would like to have a function that calculates a number telling us how good a word $c$ is for expanding a query $Q$ based on $c$'s co-occurrence information with the words of $Q$ in the top ranked documents. Assuming that the words in $Q$ are $w_1, w_2, ...w_n$, we will consider the following issues:

- What do we mean when we say that $c$ co-occurs with $w_i$ in the top ranked documents?

  We must take random co-occurrences into account: $c$ could just co-occur with $w_i$ in the top ranked documents by chance. The higher its frequency in the whole corpus, the more likely it is that $c$ co-occurs with $w_i$ by chance. The larger the number of co-occurrences, the less likely that $c$ co-occurs with $w_i$ by chance. Let $N$ be the number of documents in the corpus, $D$ the number of top ranked documents used, $n_c$ the number of documents in the corpus that contain $c$, $co(c, w_i)$ the number of co-occurrences between $c$ and $w_i$ in the top ranked documents. We come up with the following metric measuring the degree of co-occurrence of $c$ with $w_i$ :

$$co\_degree(c, w_i) = log_{10}(co(c, w_i) + 1)idf(c)/log_{10}(D)$$

$$idf(c) = min(1.0, log_{10}(N/N_c)/5.0)$$

  The metric takes into account the frequency of $c$ in the corpus ($idf(c)$) and the number of co-occurrences between $c$ and $w$ in the top ranked documents

$(co(c, w_i))$. The logarithm function is used to dampen the raw numbers of occurrences and co-occurrences. The metric is also normalized over the number of top ranked documents used. We should point out that our definition of *idf* is somewhat different from the standard definition $idf(c) = log_{10}(N/N_c)$ used by other researchers. The problem with the latter definition is that mathematically it has no upper limit when $N$ approaches infinity. Our formula sets an upper limit on $idf(c)$. Any word which occurs in 1/100000 of the documents or fewer in the collection will have *idf* 1.0.

- How do we combine the degrees of co-occurrence with all query words?

  To obtain a value measuring how good $c$ is for query $Q$ overall, we need to combine its degrees of co-occurrence with $w_1$, $w_2$, ... $w_n$. We consider two plausible functions for the purpose.

  One is to simply sum the degrees of co-occurrences together. The sum function, however, has a drawback. If some of the $n$ numbers are very small and the rest are very large, the sum is still large, though in this case $c$ is not particularly good according to our hypothesis.

  The other is to multiply the $n$ numbers together. In order for the product to be large, none of them can be very small. This property suits us well. A problem with the product function is if one of the $n$ numbers is 0, the product is 0 no matter what the other $n - 1$ numbers are. If one query word does not occur in the top ranked documents, the output value will be 0 for any $c$. This is not desirable. A more desirable function should produce a non-zero value based on the other $n - 1$ numbers. To overcome this problem, we add a small value $\delta$ to each degree of co-occurrence. The function for combining the $n$ numbers is therefore

$$g(c, Q) = \prod_{w_i \text{ in Q}} (\delta + co\_degree(c, w_i))$$

If $co\_degree(c, c_j) = 0$ for some $j$,

$$g(c, Q) = \delta \prod_{w_i \text{ in } Q \text{ and } i \neq j} (\delta + co\_degree(c, w_i))$$

Obviously, not all query words are equally important. While deciding the importance of a query word is a hard problem in general, it is well known that infrequent query words are usually more important than frequent ones. Based on this fact, most IR systems incorporate the inverse document frequency ($idf$) of the query words in their function of calculating the score of a document to a query. Taking into account the $idf$ of the query words, we get the following function,

$$f(c, Q) = \prod_{w_i \text{ in } Q} (\delta + co\_degree(c, w_i))^{idf(w_i)}$$

The $idf(w_i)$'s can be viewed as weights in the formula. It is easy to see this by taking $log$ on both sides of the formula. This formula is used in this thesis for selecting expansion words for a query. We call $f(c, Q)$ $c$'s *suitability* for $Q$.

To expand a query $Q$, we rank the words in the top ranked documents according to their suitability for $Q$ and choose the top ranked words for query expansion.

With local context analysis, we do not need to build a global data structure. This cost is the computational bottleneck of the query expansion techniques based corpus analysis. Local context analysis incurs a run time cost for each query, though the cost is modest. The run time cost of local context analysis can be broken into two parts. The first part is the cost to carry out an initial retrieval for a query and return the top ranked documents. This is the dominant part. Since we are only interested in the top ranked documents, techniques for efficient retrieval of the top ranked documents can be used to minimize this cost [4]. The second part is the cost to process the top ranked documents, build the co-occurrence information, compute the suitability for

each word in the top ranked documents and return the top ranked words. The time cost of this part is bounded by $O(nl)$, where $n$ is the number of query words and $l$ the total length (in words) of the top ranked documents used. The space cost is $O(nm)$, where $m$ is the number of unique words in the top ranked documents.

Hypothesis 5: Local context analysis can achieve more effective retrieval than local feedback and corpus analysis.

Hypothesis 6: Local context analysis is computationally practical.

## 4.4 Summary

In this chapter we have discussed query expansion techniques based on corpus analysis and analysis of top retrieved documents. We hypothesized that corpus analysis based on the association hypothesis is useful for stemming, but its value is limited for general query expansion. We also hypothesized that applying statistical co-occurrence analysis on the top ranked documents could find better expansion words and produce more effective retrieval. In the remaining chapters we will present experimental results which verify the hypotheses.

# Chapter 5

## Corpus Analysis for Stemming: Experiments

In this chapter we will carry out experiments to investigate the usefulness of corpus analysis for stemming, a limited type of query expansion, which expands a query word to its related morphological variants. We discuss how to use the equivalence classes for query expansion in Section 5.1. We describe the procedure of generating equivalence classes based on co-occurrence information in Section 5.2. We report experimental results on English test collections in Section 5.3 and results on a Spanish collection in 5.4. We discuss portability of equivalence classes across corpora in Section 5.5. We discuss efficiency issues in Section 5.6. We summarize the experimental results in Section 5.7. The test collections used are WSJ, WSJ91, WEST and TREC4-SPANISH. In previous work, stemming phrases produced different results than stemming words [31]. To ensure the generality of the results, we use two versions of queries on WEST, the natural language and the structured query sets. Descriptions and statistics about these collections can be found in Chapter 3.

### 5.1 Query Based Stemming vs Indexing Time Stemming

As we discussed in Chapter 4, stemming forms equivalence classes among word variants. There are two approaches of using equivalence classes generated by a stemming algorithm. One approach is to assign a unique identifier to each equivalence class and replace occurrences of a word variant with its class identifier at document indexing time and at query processing time. We label this approach *indexing time stemming*. The other approach does not touch the word variants at document indexing time but instead, at query processing time, expands a word variant to all the word variants in its equivalence class. We call this approach *query based stemming*. For example, given the following query using INQUERY's query language,

```
#SUM(stock prices IBM)
```

where #SUM is an INQUERY operator that combines the evidence from the individual words, the expanded query using a particular stemmer could be

```
#SUM( #SYN(stock stocks) #SYN(price prices) IBM)
```

where the #SYN operator is used to group synonyms. In INQUERY, this means that the inverted lists corresponding to each word in the argument of #SYN are combined to create a list for a new "virtual" word.

Since both approaches of using equivalence classes match a query word with all the words in its equivalence class, they are equivalent from the retrieval point of view. Given a query, they should produce identical output. However, this is not always true depending on the underlying retrieval system. Most retrieval systems use some form of the $tf \times idf$ formula to estimate the likelihood that a document is relevant to a query. In some variations of the formula, the $tf$ component is normalized by the maximum frequency of any word in the document. Since the two approaches can produce different maximum frequencies for the same document, the outputs can be slightly different. Table 5.1 compares the retrieval effectiveness of indexing time stemming and query based stemming on the WSJ collection using the 2.1 version of INQUERY. As we can see, the difference between query based stemming and indexing time stemming is very small. This means any technique that improves the retrieval performance using one approach will likely improve retrieval performance using the other.

Both approaches have pros and cons. Indexing time stemming saves storage space and offers faster retrieval. But it is inflexible in that the user has no way to alter the expansion even when it is necessary. Query based stemming requires slightly more storage space and offers slower retrieval. But it is more flexible in that the user can alter the expanded query if necessary by adding a word variant missed by the stemmer or deleting an inappropriate expansion word. The capability of

Table 5.1. Comparing the retrieval performance of indexing time and query based stemming

| | Precision (% change) – 66 queries | | |
|---|---|---|---|
| Recall | INDEX | QUERY BASED | |
| 0 | 80.6 | 82.2 | (+2.0) |
| 10 | 52.8 | 53.0 | (+0.4) |
| 20 | 45.5 | 45.2 | (−0.6) |
| 30 | 40.8 | 40.5 | (−0.7) |
| 40 | 36.1 | 35.8 | (−0.9) |
| 50 | 31.3 | 31.2 | (−0.1) |
| 60 | 26.5 | 26.2 | (−1.0) |
| 70 | 22.3 | 22.1 | (−0.9) |
| 80 | 17.2 | 17.1 | (−0.5) |
| 90 | 11.9 | 11.8 | (−0.7) |
| 100 | 2.7 | 2.7 | (−0.1) |
| average | 33.4 | 33.4 | (+0.1) |

allowing the user to control the expanded query is very important in cases where small differences in word forms result in large differences in the relevance of the retrieved documents. For example, in looking for articles about specific terrorist incidents, the word "assassination" is very good at discriminating relevant from non-relevant documents, but the word "assassinations" is much less useful because it tends to be used in more general stories [49].

In addition to its flexibility, query based stemming is more suitable for investigating the retrieval effectiveness of various methods of generating equivalence classes. With indexing time stemming, we need to re-index a collection every time we modify the equivalence classes, even when the modification is limited to only a few equivalence classes. This is impractical because of amount of the storage and time required. With query based stemming, we only need to index a collection once. When we modify the equivalence classes, we only need to modify the expanded queries.

With query based stemming, the retrieval efficiency is significantly affected by the number of words in the expanded queries, as we will show in Section 5.6. We will use two metrics to measure how many words on average are added for each query

word using certain set of equivalence classes. One is the average number of words in an equivalence class, *average class length*. The average class length is somewhat misleading because it does not take into account word frequencies. The other metric, *expansion factor*, is based on the actual queries provided for each test collection. The expansion factor of a set of equivalence classes X for set of queries Y is defined as

$$\text{expansion factor} = \frac{\text{number of words in Y after expansion using X}}{\text{number of words in Y before expansion}}$$

## 5.2 Equivalence Class Generation

As we discussed in Chapter 4, the way we employ corpus analysis for stemming is to refine a set of initial equivalence classes based on co-occurrence information. In this section we will describe equivalence class generation in detail. We will first describe how to construct the initial equivalence classes. Then we will describe the steps of applying corpus analysis to generate the final equivalence classes.

### 5.2.1 Initial Equivalence Class Generation

We will use three methods to generate the initial equivalence classes. The first method is to use an existing stemmer. In this case, each equivalence class corresponds with a root. The equivalence class corresponding with root $r$ consists of the words which are stemmed to $r$ by the stemmer.

The second method is to use two different stemmers. In this case, we need to merge two sets of equivalence classes, each generated by one stemmer. The algorithm to merge two sets of equivalence classes $S1$ and $S2$ to generate a single set of equivalence classes $S$ is:

1. Make each word a singleton equivalence class in $S$.
2. For each equivalence class $X \in S_1 \bigcup S_2$, let $X = x_1, x_2, ...x_m$,
   (a) For $2 \leq i \leq m$, if $x_1$ and $x_i$ are in two different equivalence classes $C_1$ and $C_2$ in $S$, merge $C_1$ and $C_2$ in $S$.

The running time of the algorithm is $O(nlog^*n)$, "almost" linear time because $log^*n$ is a small number even if $n$ is very large [11].

The third method is to use initial character matching. In this case, two words are in the same equivalence class if their first 3 characters are the same.

### 5.2.2 Steps to Apply Corpus Analysis to Generate Equivalence Classes

Four steps are taken to generate the equivalence classes using corpus analysis on a collection $C$:

1. Collect all unique word variants in $C$. Numbers, stop words and possible proper nouns are discarded. Words that only occur in the capitalized form (except at the beginning of a sentence) are considered to be proper nouns.

2. Generate the initial equivalence classes using one of the three methods described in Section 5.2.1.

3. Collect the co-occurrence data and calculate the *em* scores for the word pairs from the same initial equivalence class. Co-occurrence is determined using a fixed size text window. To estimate $k$ for calculating the expected co-occurrences we also need to collect the co-occurrence data for a number of randomly chosen word pairs. The details of calculating the *em* scores can be found in Chapter 4.

4. Refine the initial equivalence classes based on the *em* scores using the connected component and the optimal partition algorithms. The algorithms were discussed in Chapter 4. The optimal partition algorithm is applied on the equivalence classes generated by the connected component algorithm. For equivalence classes having more than 12 members, the approximation algorithm for the optimal partition algorithm is applied. Two parameters are involved in this step: the *em* threshold of the connected component algorithm and $\delta$ of the optimal partition algorithm.

The equivalence classes generated from the above steps are used for query based stemming as we discussed in Section 5.1. There are a number of variations and parameters involved in the above process. In the experiments, we will investigate different ways of generating equivalence classes and observe their impact on retrieval performance. Appropriate parameter values will be empirically determined.

## 5.3 English Experiments

### 5.3.1 Baseline Stemmers

The Porter stemmer [46] and KSTEM [31] are the two baseline stemmers against which we compare retrieval results on the English corpora. We will use the terms *collection* and *corpus* interchangeably. Tables 5.2 and 5.3 show some statistics about the sets of equivalence classes generated by Porter and KSTEM. As we can see from the tables, the average class lengths of KSTEM and Porter are very close (1.84 vs 1.9 on WEST, 1.7 vs 1.9 on WSJ) but the expansion factor of KSTEM is much smaller than that of Porter (3.1 vs 5.5 on WEST, 2.8 vs 4.5 on WSJ). The expansion factors suggest that Porter is more aggressive than KSTEM. Figures 5.1 and 5.2 list some equivalence classes generated by KSTEM and Porter on the WSJ collection. In the figures, each line shows an equivalence class if it fits. If an equivalence class is too long to fit in one line, multiple lines are used but all lines except the first are indented and start with '–'. These equivalence classes also support that Porter is more aggressive than KSTEM.

```
abandon abandoned abandoning abandonment abandonments abandons
abate abated abatement abatements abates abating
abrasion abrasions abrasive abrasively abrasiveness abrasives
absorb absorbable absorbables absorbed absorbencies absorbency
 -absorbent absorbents absorber absorbers absorbing absorbs
abusable abuse abused abuser abusers abuses abusing abusive abusively
access accessed accessibility accessible accessing accession
```

Figure 5.1. Example Porter equivalence classes on WSJ

```
abandon abandoning abandonment abandonments abandons
abandoned
abate abated abatement abatements abates abating
abraded abrading
abrasion abrasions
abrasive abrasively abrasiveness abrasives
absorb absorbable absorbables absorbed absorber absorbers absorbs
absorbencies absorbency absorbent absorbents
absorbing absorbingly
abusable abuse abused abuser abusers abuses abusing
abusive abusively
access accessed accessing
accessibility accessible
accession
```

Figure 5.2. Example KSTEM equivalence classes on WSJ

Table 5.2. Equivalence class statistics of different stemmers on WEST, 49,964 unique word variants

| stemmer | KSTEM | Porter | Porter-CC | Porter-Optimal | ngram-CC | ngram-Optimal |
|---|---|---|---|---|---|---|
| number of classes | 27117 | 26211 | 40215 | 46632 | 38343 | 45042 |
| average class length | 1.84 | 1.9 | 1.24 | 1.07 | 1.32 | 1.11 |
| expansion factor | 3.1 | 5.5 | 2.9 | 2.6 | 3.99 | 2.97 |

Despite the large difference in the amounts of query expansion, Porter and KSTEM are comparable in terms of retrieval effectiveness, as shown by Tables 5.5, 5.6 and 5.7. KSTEM outperforms Porter on WEST using the natural language queries while Porter outperforms KSTEM on WSJ. The t-test indicates that only the difference between Porter and KSTEM on WSJ is statistically significant with p_value=0.03 (Porter is better).

Table 5.3. Equivalence class statistics of different stemmers on WSJ, 76181 unique word variants

| stemmer | KSTEM | Porter | Porter-CC | Porter-Optimal | ngram-CC | ngram-Optimal |
|---|---|---|---|---|---|---|
| number of classes | 44543 | 39949 | 64821 | 73015 | 62174 | 71275 |
| average class length | 1.7 | 1.9 | 1.17 | 1.04 | 1.23 | 1.07 |
| expansion factor | 2.8 | 4.5 | 2.2 | 2.06 | 2.85 | 2.28 |

### 5.3.2 Enhancing Porter by Corpus Analysis

As we discussed before, Porter is a relatively aggressive stemmer. In this section we refine the equivalence classes of Porter based on co-occurrence information. Our hope is that relying on the co-occurrence metric *em* (defined in Chapter 4), we will be able to tell whether two word variants are related. The parameter values used are window size=100, *em* threshold=0.01 and $\delta = 0.0075$. Later in this chapter we will discuss how sensitive the retrieval results are to the parameter values.

Table 5.4 shows a number of word pairs conflated to the same root by Porter and their statistics in WSJ. The table clearly shows the correlation between *em* score and appropriateness for conflations. Related word pairs such as "bonds"/"bond" and "stocks"/"stock" have high *em* scores. In comparison, the inappropriate conflations made by Porter such as "policy"/"police", "new"/"news", "arm"/"army" and "desirable"/"desires" have a zero *em* score.

Table 5.4. *em* scores of example word pairs on WSJ with window size 100 words

| word pairs | frequencies | co-occurrences | em |
|---|---|---|---|
| bond/bonds | 42255/49331 | 37706 | 0.35 |
| stock/stocks | 144076/35898 | 46030 | 0.18 |
| cruise/cruises | 1253/191 | 239 | 0.17 |
| animation/animators | 172/29 | 28 | 0.14 |
| votes/voting | 3349/4577 | 625 | 0.074 |
| policy/police | 26122/7290 | 294 | 0.0 |
| new/news | 225064/81711 | 27307 | 0.0 |
| arm/army | 3004/7684 | 37 | 0.0 |
| desirable/desires | 681/211 | 0 | 0.0 |

Applying the connected component and optimal partition algorithms on the Porter equivalence classes, we obtain much smaller equivalence classes and significantly reduce the amount of expansion. This is shown by the data in Tables 5.2 and 5.3. In the tables, we denote Porter equivalence classes refined by the connected component algorithm *PorterCC* and Porter equivalence classes refined by the optimal partition algorithm *PorterOptimal*. On WEST, the expansion factor of Porter is reduced from 5.5 to 2.9 by the connected component algorithm and to 2.6 by the optimal partition algorithm. On WSJ, the expansion factor is reduced from 4.5 to 2.2 to 2.06. In short, corpus analysis reduced the amount of expansion by 50% or more. The enhanced stemmers, PorterCC and PorterOptimal, make even less expansion to the original queries than the conservative KSTEM.

Figure 5.3 shows the result of applying the connected component algorithm on the equivalence classes in Figure 5.1.

```
abandonment abandonments
abated abatement abatements
abrasive abrasives
absorbable absorbables
absorbencies absorbency absorbent
absorber absorbers
abuse abused abuser abusers abuses abusing abusive
accessibility accessible
```

Figure 5.3. The result of applying the connected component algorithm on the classes in Figure 5.1. Singleton classes are not shown.

We mentioned in Chapter 4 that the connected component algorithm can generate largely loosely connected classes called "strings" which the optimal partition algorithm avoids. We now examine a concrete example to see why this can happen. One of the PorterCC equivalence class on WSJ is "organization organizations organize organized organizer organizers organizing". Though "organization" and "organize" are not directly connected (the *em* score is 0), they are indirectly linked through the

word pairs "organization"/"organizations", "organizations"/"organized" and "organized"/"organize". The optimal partition algorithm, however, breaks the class into two classes "organization organizations" and "organize organized organizer organizers organizing" because the two classes have no connection besides a marginal connection between "organizations"/"organized".

Table 5.5. Enhancing Porter by corpus analysis on WEST (natural language queries)

| | Precision (% change) – 34 queries | | | |
|---|---|---|---|---|
| Recall | KSTEM | Porter | PorterCC | PorterOptimal |
| 0 | 84.1 | 84.3 (+0.3) | 85.6 (+1.8) | 86.3 (+2.6) |
| 10 | 79.9 | 78.2 (−2.1) | 79.1 (−1.0) | 80.4 (+0.6) |
| 20 | 75.3 | 72.8 (−3.3) | 74.0 (−1.7) | 75.8 (+0.6) |
| 30 | 71.4 | 70.1 (−1.8) | 71.3 (−0.1) | 72.6 (+1.7) |
| 40 | 61.2 | 60.2 (−1.7) | 60.4 (−1.3) | 61.1 (−0.1) |
| 50 | 54.0 | 52.9 (−2.0) | 54.6 (+1.0) | 54.6 (+1.1) |
| 60 | 44.1 | 44.5 (+0.9) | 45.8 (+4.0) | 46.0 (+4.4) |
| 70 | 36.0 | 35.5 (−1.5) | 37.2 (+3.2) | 37.1 (+2.9) |
| 80 | 27.0 | 28.7 (+6.4) | 30.8 (+14.1) | 30.6 (+13.5) |
| 90 | 15.7 | 16.5 (+4.9) | 16.4 (+4.4) | 16.3 (+3.9) |
| 100 | 8.1 | 8.8 (+7.9) | 9.1 (+12.1) | 8.9 (+9.9) |
| average | 50.6 | 50.2 (−0.8) | 51.3 (+1.3) | 51.8 (+2.3) |

Tables 5.5, 5.6 and 5.7 report the retrieval effectiveness of PorterCC and PorterOptimal. On both WEST and WSJ, PorterCC and PorterOptimal outperform KSTEM and Porter. The results also show that the optimal partition algorithm is better than the connected component algorithm. On WEST, PorterOptimal is better than PorterCC using both natural language and structured queries while on WSJ PorterCC and PorterOptimal perform equally well. Though the improvements in retrieval effectiveness are not very impressive, we consider the results encouraging because corpus analysis reduced the amount of expansion by 50% or more. The fact that corpus analysis significantly reduced the amount of expansion and resulted in small improvements in retrieval performance suggests that it successfully prevented many

Table 5.6. Enhancing Porter by corpus analysis on WEST (structured queries)

| | Precision (% change) – 34 queries | | | |
|---|---|---|---|---|
| Recall | KSTEM | Porter | PorterCC | PorterOptimal |
| 0 | 87.8 | 88.5 (+0.9) | 88.4 (+0.7) | 88.3 (+0.6) |
| 10 | 80.9 | 80.7 (−0.2) | 80.7 (−0.2) | 81.2 (+0.3) |
| 20 | 75.5 | 74.6 (−1.1) | 74.8 (−0.9) | 75.3 (−0.3) |
| 30 | 72.0 | 70.7 (−1.9) | 71.3 (−1.0) | 72.1 (+0.1) |
| 40 | 63.8 | 63.7 (−0.2) | 64.0 (+0.4) | 64.5 (+1.1) |
| 50 | 57.2 | 57.4 (+0.3) | 58.6 (+2.5) | 58.6 (+2.5) |
| 60 | 49.2 | 49.7 (+0.9) | 50.5 (+2.7) | 50.4 (+2.4) |
| 70 | 41.5 | 41.9 (+1.0) | 42.1 (+1.3) | 42.2 (+1.6) |
| 80 | 30.7 | 33.4 (+8.7) | 33.4 (+8.6) | 33.3 (+8.5) |
| 90 | 19.7 | 20.1 (+2.3) | 20.4 (+3.7) | 20.4 (+3.8) |
| 100 | 9.3 | 9.1 (−1.7) | 9.4 (+1.2) | 9.4 (+1.2) |
| average | 53.4 | 53.6 (+0.4) | 54.0 (+1.0) | 54.1 (+1.4) |

incorrect conflations made by the Porter stemmer. This is advantageous in interactive applications where expanded queries are accessible to end users.

Table 5.8 shows the t-test results comparing the baseline stemmers, PorterCC and PorterOptimal. The t-test indicates that PorterOptimal is significantly better than both KSTEM and Porter on WEST using natural language queries. On WEST, using structured queries, and on WSJ, PorterOptimal is significantly better than KSTEM but not significantly better than Porter. There is no significant difference between PorterOptimal and PorterCC on any of the three data sets, but PorterOptimal achieved more significant results over the baseline stemmers than PorterCC did. The t-test and the average precision figures in Tables 5.5, 5.6 and 5.7 show that corpus analysis does help stemming. They also show that optimal partition algorithm produces better equivalence classes than the connected component algorithm.

#### 5.3.2.1 Parameter Setting

As we discussed before, there are a number of parameters in class generation. We now investigate how the parameter settings affect retrieval performance. Table 5.9 shows the effect of the window size and the *em* threshold on the performance of

Table 5.7. Enhancing Porter on WSJ

| | Precision (% change) – 66 queries | | | |
|---|---|---|---|---|
| Recall | KSTEM | Porter | PorterCC | PorterOptimal |
| 0 | 84.0 | 82.2 (−2.1) | 81.8 (−2.5) | 81.9 (−2.4) |
| 10 | 51.7 | 53.0 (+2.6) | 52.7 (+2.0) | 53.0 (+2.6) |
| 20 | 45.0 | 45.2 (+0.5) | 45.8 (+1.9) | 45.9 (+1.9) |
| 30 | 39.2 | 40.5 (+3.3) | 40.8 (+4.2) | 40.9 (+4.4) |
| 40 | 34.7 | 35.8 (+3.2) | 36.6 (+5.5) | 36.7 (+5.7) |
| 50 | 30.0 | 31.2 (+4.1) | 31.2 (+4.0) | 31.2 (+3.9) |
| 60 | 25.1 | 26.2 (+4.3) | 26.4 (+5.1) | 26.3 (+4.6) |
| 70 | 20.8 | 22.1 (+6.2) | 22.1 (+6.0) | 21.8 (+4.6) |
| 80 | 16.4 | 17.1 (+4.7) | 17.2 (+5.2) | 17.0 (+4.1) |
| 90 | 11.3 | 11.8 (+4.6) | 12.0 (+6.5) | 11.9 (+5.9) |
| 100 | 2.3 | 2.7 (+17.2) | 2.7 (+19.5) | 2.7 (+18.6) |
| average | 32.8 | 33.4 (+2.1) | 33.6 (+2.5) | 33.6 (+2.5) |

Table 5.8. Significance tests on performance difference between baseline stemmers, PorterCC and PorterOptimal

| Collection | positive pairs (p_value) |
|---|---|
| WEST natural language | PorterCC > Porter (0.013)<br>PorterOptimal > KSTEM (0.013)<br>PorterOptimal > Porter (0.012) |
| WEST structure | PorterOptimal > KSTEM (0.027) |
| WSJ | PorterCC > KSTEM (0.011)<br>PorterOptimal > KSTEM (0.007) |

PorterCC on WEST using natural language queries. As we can see, many sets of parameter values achieved good performance. The window size and the *em* threshold are not independent: with a larger window, we need to increase the *em* threshold in order to achieve good performance. This is understandable because the co-occurrence statistics depend on the window size: a larger window size means more co-occurrences for the same pair of words. As long as we use a reasonably large (at least 50 words) window and an appropriate *em* threshold for that window, retrieval effectiveness seems to be independent of the window size. But in practice, window size matters because the time to collect the co-occurrence data is proportional to the window size.

In order to minimize computational cost and achieve good retrieval effectiveness, for the rest of the experiments we use a window size of 100 words and an *em* threshold of 0.01.

Table 5.9. The effect of window size and *em* threshold on retrieval effectiveness on WEST using natural language queries. 11 point average precision is used

| | *em* threshold | | | | |
|---|---|---|---|---|---|
| window size | 0.005 | 0.01 | 0.02 | 0.03 | 0.05 |
| 50 | 51.0 | 51.5 | 51.0 | 50.4 | 49.6 |
| 100 | 51.0 | 51.3 | 51.6 | 51.1 | 50.6 |
| 200 | 50.9 | 51.1 | 51.4 | 51.4 | 50.9 |
| 500 | 50.7 | 50.9 | 51.1 | 51.2 | 51.4 |

Table 5.10 shows the effect of the $\delta$ value of the optimal partition algorithm on retrieval effectiveness. The optimal partition algorithm is applied on the PorterCC equivalence classes (using the default window size 100 and *em* threshold 0.01). We get some improvement using $\delta$ value 0.0075. Even if we can do slightly better using an even larger $\delta$ value (0.01 or 0.015), we will use 0.0075 as the default $\delta$ value. The reason is that a high $\delta$ value will prevent many good conflations. Though a high $\delta$ value is not a problem on WEST, it could be a problem on other collections.

Table 5.10. The effect of $\delta$ values on retrieval effectiveness on WEST using natural language queries

| $\delta$ | 11 pt precision |
|---|---|
| 0.003 | 51.3 |
| 0.005 | 51.4 |
| 0.0075 | 51.8 |
| 0.01 | 52.0 |
| 0.015 | 52.2 |
| 0.02 | 51.9 |

### 5.3.3 Using Multiple Stemmers

Though the Porter stemmer is an aggressive stemmer, it occasionally fails to bring related word variants in an equivalence classes. Examples of word pairs that Porter stemmer fails to conflate are "abdomen"/"abdominal", and "searcher"/"search". Since the equivalence class refinement algorithms only break up the initial equivalence classes and never merge them, these word pairs are in different equivalence classes after applying the algorithms. Many of the related word pairs missed by Porter, however, are conflated by KSTEM. If we use both Porter and KSTEM to construct the initial equivalence classes, we will miss fewer good conflations and, it is hoped, achieve better stemming than using the Porter stemmer alone.

On the WEST collection, merging the equivalence classes of Porter and KSTEM results in 25496 classes with average class length 1.96 (compared to 1.9 of Porter). We get slightly longer and fewer equivalence classes than using Porter alone. For example, two equivalence classes of Porter "bribe bribed bribes bribing", "briber bribery" and two equivalence classes of KSTEM "bribe bribed briber bribes bribing", "bribery" are merged into a single equivalence class "bribe bribed briber bribes bribing bribery".

Applying the connected component and optimal partition algorithms on the set of initial equivalence classes constructed using both Porter and KSTEM, we obtained two sets of equivalence classes *PorterKSTEM-CC* and *PorterKSTEM-Optimal*. They have slightly larger average class lengths than PorterCC and PorterOptimal on WEST (1.26 and 1.23 of PorterKSTEM-CC and PorterKSTEM-Optimal vs. 1.24 and 1.07 of PorterCC and Porter-Optimal). Table 5.11 shows the retrieval performance of PorterKSTEM-CC and PorterKSTEM-Optimal on WEST using natural language queries. As can be seen, the difference in retrieval performance between the two ways of constructing the initial equivalence classes (Porter alone vs both Porter and KSTEM) is very small. This can be explained by the fact that KSTEM is a very conservative stemmer. Using KSTEM in addition to Porter only reduces about 700 equivalence classes on WEST. That does not give us enough additional conflations

to have an impact on retrieval performance. In fact, only 25 words in the WEST queries (repeated query words are counted multiple times) are expanded differently by PorterCC and PorterKSTEM-CC.

Table 5.11. Enhancing the initial equivalence classes constructed using both KSTEM and Porter on WEST (natural language queries)

| | Precision (% change) – 34 queries | | | | | |
|---|---|---|---|---|---|---|
| Recall | PorterCC | PorterOptimal | | PorterKSTEM-CC | | PorterKSTEM-Optimal | |
| 0 | 85.6 | 86.3 | (+0.8) | 87.1 | (+1.8) | 85.8 | (+0.3) |
| 10 | 79.1 | 80.4 | (+1.6) | 79.4 | (+0.4) | 80.6 | (+1.9) |
| 20 | 74.0 | 75.8 | (+2.4) | 73.9 | (−0.2) | 75.5 | (+2.0) |
| 30 | 71.3 | 72.6 | (+1.8) | 71.2 | (−0.2) | 72.3 | (+1.4) |
| 40 | 60.4 | 61.1 | (+1.2) | 60.6 | (+0.3) | 61.2 | (+1.3) |
| 50 | 54.6 | 54.6 | (+0.1) | 54.1 | (−0.9) | 54.5 | (−0.1) |
| 60 | 45.8 | 46.0 | (+0.4) | 45.7 | (−0.3) | 46.0 | (+0.3) |
| 70 | 37.2 | 37.1 | (−0.3) | 36.8 | (−1.0) | 37.0 | (−0.6) |
| 80 | 30.8 | 30.6 | (−0.5) | 30.3 | (−1.5) | 30.5 | (−0.7) |
| 90 | 16.4 | 16.3 | (−0.4) | 16.8 | (+2.6) | 16.8 | (+3.0) |
| 100 | 9.1 | 8.9 | (−2.0) | 9.1 | (−0.1) | 9.0 | (−1.0) |
| average | 51.3 | 51.8 | (+0.9) | 51.4 | (+0.1) | 51.8 | (+0.9) |

In the future, it would be more interesting to use other combinations of existing stemming algorithms than KSTEM and Porter. Presumably, corpus analysis applied on the equivalence classes constructed using two more aggressive stemmers (e.g. Lovins and Porter) would produce better retrieval performance than using KSTEM and Porter.

### 5.3.4 Ngram Stemmers

#### 5.3.4.1 Trigram Matching

In the previous experiments, corpus analysis was applied on a set of equivalence classes constructed using existing stemming algorithms. Using existing stemmers has the problem that if they miss some important conflations, we have no way to recover the conflations. Also, this method requires that an initial stemmer be available. To

address the problems, we now use a more aggressive approach to constructing the initial equivalence classes: two words are in the same equivalence class if their first three characters are the same. Thus "color", "cola" and "collect" are initially in the same equivalence class. We label this approach to constructing initial equivalence classes *initial trigram matching*, or *trigram matching* for short. The reason we do not use fewer than 3 characters is that we would end up with unmanageably large equivalence classes. The reason we do not use more than 3 characters is that we would miss the conflations of short words such as "ant"/"ants", "aim"/"aims" and "bag"/"bags". Obviously trigram matching captures almost all potentially good conflations but makes many mistakes. We will show that the mistakes can be handled by corpus analysis.

The equivalence classes constructed by trigram matching are very long. The average class length is 18.5 on WEST and 25.8 on WSJ. On WSJ, 165 classes have more than 100 members. The largest equivalence class on WSJ, with initial characters "con", has 1,124 members. There are over 6 million word pairs whose co-occurrence data we need to collect from WSJ, but only 278,459 pairs ever co-occur in the collection. Applying the connected component algorithm, the average class length is reduced to 1.45 on WEST and 1.32 on WSJ. Some equivalence classes are still too long: 50 classes have more than 20 members on WEST. Since the optimal partition algorithm can not handle long equivalence classes, we run the approximation version of the optimal partition algorithm on the connected component classes. The average class length is reduced to 1.34 on WEST and 1.25 on WSJ after applying the optimal partition algorithm.

Table 5.12 shows the retrieval performance of WEST natural language queries using the trigram optimal partition equivalence classes. Though the performance is worse than the baseline stemmers, it is reasonable considering the crude way of constructing the initial equivalence classes. An examination of the expanded queries shows that many expansion words are only remotely related to the query

words. These words happen to have the same 3 initial characters as some query word but are not morphologically related to the query word. Examples are "application"/"approval", "process"/"prosecution" and "discrimination"/"discharge". But this simple minded approach also results in good conflations ordinarily missed by existing stemmers. Examples are "maintain"/"maintenance", "share"/"shareholder" and "corp"/"corporation".

Table 5.12. Enhancing trigram by corpus analysis on WEST using natural language queries

| | Precision (% change) – 34 queries | | | | |
|---|---|---|---|---|---|
| Recall | KSTEM | Porter | | trigram-optimal | |
| 0 | 84.1 | 84.3 | (+0.3) | 86.3 | (+2.6) |
| 10 | 79.9 | 78.2 | (−2.1) | 75.4 | (−5.6) |
| 20 | 75.3 | 72.8 | (−3.3) | 70.1 | (−7.0) |
| 30 | 71.4 | 70.1 | (−1.8) | 66.3 | (−7.2) |
| 40 | 61.2 | 60.2 | (−1.7) | 55.7 | (−9.0) |
| 50 | 54.0 | 52.9 | (−2.0) | 48.0 | (−11.1) |
| 60 | 44.1 | 44.5 | (+0.9) | 40.5 | (−8.2) |
| 70 | 36.0 | 35.5 | (−1.5) | 31.8 | (−11.8) |
| 80 | 27.0 | 28.7 | (+6.4) | 25.8 | (−4.3) |
| 90 | 15.7 | 16.5 | (+4.9) | 12.6 | (−19.9) |
| 100 | 8.1 | 8.8 | (+7.9) | 7.5 | (−7.8) |
| average | 50.6 | 50.2 | (−0.8) | 47.3 | (−6.7) |

#### 5.3.4.2 Prefix Restriction

Most incorrect conflations made by trigram matching involve words with common prefixes. Our solution to this problem is prefix restriction: more characters beyond the 3 initial characters are used to determine class membership for words with common prefixes. Suppose words $a$ and $b$ have the same prefix $p$. We force the $em$ score between $a$ and $b$ to be zero (even if they co-occur very often) if either of the following conditions is true:

1. $length(a) \geq length(p)+4$ or $length(b) \geq length(p)+4$, and the first $3+length(p)$ characters of $a$ and $b$ are not the same.

2. $length(a) = length(b) = length(p) + 3$ and the first $1 + length(p)$ characters are not the same.

If two or more prefixes are applicable, the longest is used. A prefix is defined as an initial substring shared by more than 100 words in the WSJ collection. Figure 5.4 lists the all prefixes.

```
acc air all ant app ass aut bac bal ban bar bea bel bio bla blo boo
bra bre bri bro cal can cap car cas cat cha che chi cho cla clo col
com con cor cou cra cre cro cur dec def del dem dep des det dis div
ele emb enc ent exc exp ext fin fla flo for fra fre gen gra gro han
har hea hig hom hyp imm imp inc ind inf ins int inv lea man mar mat
met mic mid mil min mis mon mul non out ove par pat pen per pho pla
pol pos pre pri pro qua qui rea rec red ref reg rei rel rem rep res
ret rev sal sca sch scr sea sec sem sha shi sho sno spa spe spi spo
squ sta ste sti sto str stu sub sup sur tel the thr tra tre tri una
unc und une unf uni unp unr uns war whi win wor
anti back coll comm comp conc conf cons cont conv coun disa disc disp
dist elec fore hand high inte intr micr mini mult nonc over para pres
prop reco rein rest stra subs supe tele tran unco unde unre
contr count elect inter micro multi super trans under
counter
```

Figure 5.4. List of English prefixes

By forcing their *em* score to be zero, we discourage the decision of putting $a$ and $b$ in one equivalence class. This does not mean $a$ and $b$ will definitely be in different classes after applying the class refinement algorithms. They can still end up in one class by association with other words.

Besides prefix restriction, there is no change to the procedure of equivalence class generation. We label this modified trigram approach *initial ngram matching*, or *ngram matching* for short, because more than 3 initial characters may be used to determine class membership. The corresponding set of equivalence classes generated by the class refinement algorithms are named *ngramCC* and *ngramOptimal*.

The statistics about ngramCC and ngramOptimal equivalence classes are shown in Tables 5.2 and 5.3. On WEST, the average class length of ngramCC is 1.32 and its expansion factor is 3.99. The average class length of ngramOptimal is 1.11 and its expansion factor is 2.97. On WSJ, the average class length of ngramCC is 1.23 and its expansion factor is 2.85. The average class length of ngramOptimal is 1.07 and its expansion factor is 2.28. The amount of expansion made by ngramOptimal is even less than the conservative KSTEM.

Table 5.13. The performance of ngramCC and ngramOptimal on WEST using natural language queries

| | Precision (% change) – 34 queries | | | |
|---|---|---|---|---|
| Recall | KSTEM | Porter | ngramCC | ngramOptimal |
| 0 | 84.1 | 84.3 (+0.3) | 87.8 (+4.5) | 87.2 (+3.7) |
| 10 | 79.9 | 78.2 (−2.1) | 78.4 (−1.8) | 79.4 (−0.6) |
| 20 | 75.3 | 72.8 (−3.3) | 73.8 (−2.0) | 75.8 (+0.6) |
| 30 | 71.4 | 70.1 (−1.8) | 69.3 (−2.9) | 71.2 (−0.4) |
| 40 | 61.2 | 60.2 (−1.7) | 60.7 (−0.9) | 61.6 (+0.7) |
| 50 | 54.0 | 52.9 (−2.0) | 52.9 (−2.0) | 54.9 (+1.6) |
| 60 | 44.1 | 44.5 (+0.9) | 44.0 (−0.3) | 44.7 (+1.4) |
| 70 | 36.0 | 35.5 (−1.5) | 35.7 (−0.9) | 37.1 (+2.9) |
| 80 | 27.0 | 28.7 (+6.4) | 27.2 (+0.8) | 28.6 (+5.9) |
| 90 | 15.7 | 16.5 (+4.9) | 15.5 (−1.1) | 16.6 (+5.9) |
| 100 | 8.1 | 8.8 (+7.9) | 8.5 (+4.5) | 9.0 (+10.7) |
| average | 50.6 | 50.2 (−0.8) | 50.4 (−0.5) | 51.5 (+1.6) |

The retrieval effectiveness of ngramCC and ngramOptimal is shown in Tables 5.13, 5.14 and 5.15. They perform at least as well as the baseline stemmers. NgramOptimal outperforms the baseline stemmers on all three data sets. NgramCC is worse than ngramOptimal on WEST but is about equally good on WSJ. It is interesting to compare ngramOptimal with PorterOptimal. Although they start with very different initial stemmers–a well designed linguistic stemmer and a crude initial character matching algorithm, they are comparable in terms of retrieval effectiveness. Tables 5.16, 5.17 and 5.18 show that PorterOptimal is slightly better on WEST

Table 5.14. The performance of ngramCC and ngramOptimal on WEST using structured queries

| | Precision (% change) – 34 queries | | | |
|---|---|---|---|---|
| Recall | KSTEM | Porter | ngramCC | ngramOptimal |
| 0 | 87.8 | 88.5 (+0.9) | 89.0 (+1.5) | 88.1 (+0.4) |
| 10 | 80.9 | 80.7 (−0.2) | 80.5 (−0.5) | 81.0 (+0.1) |
| 20 | 75.5 | 74.6 (−1.1) | 74.9 (−0.8) | 75.7 (+0.3) |
| 30 | 72.0 | 70.7 (−1.9) | 71.0 (−1.4) | 71.9 (−0.2) |
| 40 | 63.8 | 63.7 (−0.2) | 63.6 (−0.3) | 63.7 (−0.1) |
| 50 | 57.2 | 57.4 (+0.3) | 56.6 (−1.0) | 58.0 (+1.5) |
| 60 | 49.2 | 49.7 (+0.9) | 49.1 (−0.4) | 49.7 (+0.9) |
| 70 | 41.5 | 41.9 (+1.0) | 40.9 (−1.5) | 41.6 (+0.3) |
| 80 | 30.7 | 33.4 (+8.7) | 31.9 (+3.8) | 33.2 (+7.9) |
| 90 | 19.7 | 20.1 (+2.3) | 19.6 (−0.2) | 20.6 (+5.0) |
| 100 | 9.3 | 9.1 (−1.7) | 9.9 (+7.1) | 10.5 (+13.2) |
| average | 53.4 | 53.6 (+0.4) | 53.4 (−0.1) | 54.0 (+1.1) |

(0.6% using natural language queries, 0.3% better using structured queries) and ngramOptimal is slightly better on WSJ (0.2%).

Table 5.19 shows the t-test result on the differences between the baseline stemmers and the ngram stemmers. The t-test indicates that ngramOptimal is significantly better than ngramCC on WEST and significantly better than KSTEM on WSJ. There is no significant difference between a baseline stemmer and a ngram stemmer on WEST. Both the t-test and average precision figures show that the ngram stemmers are at least as good as the baseline stemmers. They also show the optimal partition algorithm is superior to the connected component algorithm.

Figure 5.5 shows some example ngramCC equivalence classes on WEST. Figure 5.6 gives the classes produced by the optimal partition algorithm.

The ngram approach has two advantages over the traditional approach to stemming. It requires no linguistic knowledge and therefore the process of creating a ngram stemmer can be fully automated. To retarget it to a new language is very easy. In contrast, traditional stemmers require costly human labor to code the linguistic rules for the specific languages. The other advantage of the ngram approach, as we

Table 5.15. The performance of ngramCC and ngramOptimal on WSJ

| | Precision (% change) – 66 queries | | | |
|---|---|---|---|---|
| Recall | KSTEM | Porter | ngramCC | ngramOptimal |
| 0 | 84.0 | 82.2 (−2.1) | 84.9 (+1.1) | 82.0 (−2.3) |
| 10 | 51.7 | 53.0 (+2.6) | 52.9 (+2.3) | 53.2 (+2.9) |
| 20 | 45.0 | 45.2 (+0.5) | 45.6 (+1.3) | 46.2 (+2.6) |
| 30 | 39.2 | 40.5 (+3.3) | 40.3 (+3.0) | 41.2 (+5.3) |
| 40 | 34.7 | 35.8 (+3.2) | 35.7 (+2.9) | 36.9 (+6.2) |
| 50 | 30.0 | 31.2 (+4.1) | 31.3 (+4.4) | 31.8 (+6.0) |
| 60 | 25.1 | 26.2 (+4.3) | 26.3 (+4.7) | 26.6 (+5.6) |
| 70 | 20.8 | 22.1 (+6.2) | 21.4 (+2.7) | 21.4 (+2.6) |
| 80 | 16.4 | 17.1 (+4.7) | 16.9 (+3.4) | 16.8 (+2.6) |
| 90 | 11.3 | 11.8 (+4.6) | 11.7 (+4.4) | 11.6 (+3.3) |
| 100 | 2.3 | 2.7 (+17.2) | 2.6 (+12.8) | 2.6 (+12.6) |
| average | 32.8 | 33.4 (+2.1) | 33.6 (+2.6) | 33.7 (+2.7) |

```
marked marketer marketers marker markers marketed markets marketing
 -mark marks market
anchored anchors anchor anchoring anchorage anchorages ancient
 -ancestor ancestors ancestory ancestry ancestral
engraving engravings engraver engravers engagement engagements
 -engageable engaging engaged engage
payday payroll paychecks payers paycheck payer payout payouts payrolls
```

Figure 5.5. Example ngramCC classes on WEST

discussed before, is that it can capture conflations which are missed by traditional stemmers.

The main disadvantage of the ngram approach is that it occasionally produces equivalence classes containing morphologically unrelated words. For example, in Figures 5.5 and 5.6, "mark" and "market" are in the same class. The two words have a high degree of co-occurrence in WEST because documents about trade mark infringement often mention market share. This type of association usually hurts retrieval although it sometimes results in potentially beneficial conflations such as "payroll"/"paycheck" and "punitive"/"punishment". Since this problem is less common

Table 5.16. Comparing PorterOptimal and ngramOptimal on WEST using natural language queries

| | Precision (% change) – 34 queries | | |
|---|---|---|---|
| Recall | PorterOptimal | ngramOptimal | |
| 0 | 86.3 | 87.2 | (+1.1) |
| 10 | 80.4 | 79.4 | (−1.2) |
| 20 | 75.8 | 75.8 | (+0.0) |
| 30 | 72.6 | 71.2 | (−2.0) |
| 40 | 61.1 | 61.6 | (+0.8) |
| 50 | 54.6 | 54.9 | (+0.5) |
| 60 | 46.0 | 44.7 | (−2.8) |
| 70 | 37.1 | 37.1 | (+0.0) |
| 80 | 30.6 | 28.6 | (−6.7) |
| 90 | 16.3 | 16.6 | (+2.0) |
| 100 | 8.9 | 9.0 | (+0.8) |
| average | 51.8 | 51.5 | (−0.6) |

in the ngramOptimal classes than in the ngramCC classes, the retrieval performance of ngramOptimal is better than that of ngramCC.

## 5.4 Spanish Experiments

As we discussed before, an advantage of the ngram approach to stemming is that it can be easily ported to a new language. In this section we apply the ngram approach on Spanish. To simplify the software design and the experiments, accents are removed from the Spanish characters and upper case characters are converted to lower case in the TREC4-SPANISH collection. We use exactly the same procedure as in the English experiments to generate the ngramCC and ngramOptimal equivalence classes except for one minor change: a prefix is now defined as an initial character string shared by at least 200 words rather than 100 words to accommodate the richer morphology of Spanish. The default parameter values (window size 100, *em* threshold 0.01 and $\delta$ 0.0075) are used in the Spanish experiments.

The baseline stemmer used in the Spanish experiments is a Porter style stemmer for Spanish implemented at CIIR of UMass and used by UMass in the Spanish experiments of TREC conferences [3]. Since the stemmer has no formal name, we

Table 5.17. Comparing PorterOptimal and ngramOptimal on WEST using structured queries

| | Precision (% change) – 34 queries | | |
|---|---|---|---|
| Recall | PorterOptimal | ngramOptimal | |
| 0 | 88.3 | 88.1 | (−0.2) |
| 10 | 81.2 | 81.0 | (−0.2) |
| 20 | 75.3 | 75.7 | (+0.6) |
| 30 | 72.1 | 71.9 | (−0.4) |
| 40 | 64.5 | 63.7 | (−1.2) |
| 50 | 58.6 | 58.0 | (−1.0) |
| 60 | 50.4 | 49.7 | (−1.5) |
| 70 | 42.2 | 41.6 | (−1.3) |
| 80 | 33.3 | 33.2 | (−0.5) |
| 90 | 20.4 | 20.6 | (+1.1) |
| 100 | 9.4 | 10.5 | (+11.9) |
| average | 54.1 | 54.0 | (−0.3) |

name it *S-Porter*. TREC4-SPANISH has 221,095 unique word variants. S-Porter generates 100,788 equivalence classes with average class length 2.2. The expansion factor of S-Porter is 20.4. The average class length and the expansion factor of ngramCC is 1.2 and 8.1. The average class length and the expansion factor of ngramOptimal are 1.07 and 4.1. The average class lengths of the ngram stemmers are very small because most of the infrequent words form singleton classes and the majority of the words in the collection occur infrequently. The expansion factors of the Spanish ngram stemmers are significantly larger than those of the English ngram stemmers. This is understandable because Spanish is richer in morphology than English. Figures 5.7, 5.8 and 5.9 give some equivalence classes of S-Porter, ngramCC and ngramOptimal on TREC4-SPANISH.

Table 5.20 shows the retrieval performance of S-Porter, ngramCC and ngramOptimal. As in the English experiments, the ngram stemmers perform at least as well as a traditional stemmer. NgramCC and ngramOptimal are 1.8% and 3.3% better than S-Porter. The t-test indicates no significant difference between S-Porter and the ngram stemmers. The results seem to suggest that corpus analysis and ngram

Table 5.18. Comparing PorterOptimal and ngramOptimal on WSJ

| | Precision (% change) – 66 queries | | |
|---|---|---|---|
| Recall | PorterOptimal | ngramOptimal | |
| 0 | 81.9 | 82.0 | (+0.1) |
| 10 | 53.0 | 53.2 | (+0.3) |
| 20 | 45.9 | 46.2 | (+0.7) |
| 30 | 40.9 | 41.2 | (+0.9) |
| 40 | 36.7 | 36.9 | (+0.5) |
| 50 | 31.2 | 31.8 | (+2.0) |
| 60 | 26.3 | 26.6 | (+1.0) |
| 70 | 21.8 | 21.4 | (−1.9) |
| 80 | 17.0 | 16.8 | (−1.5) |
| 90 | 11.9 | 11.6 | (−2.4) |
| 100 | 2.7 | 2.6 | (−5.0) |
| average | 33.6 | 33.7 | (+0.2) |

Table 5.19. Significance test on performance differences between baseline stemmers and ngram stemmers

| Collection | positive pairs (p_value) |
|---|---|
| WEST natural language | ngramOptimal > ngramCC (0.01) |
| WEST structure | ngramOptimal > ngramCC (0.02) |
| WSJ | ngramOptimal > KSTEM (0.03) |

matching are general techniques that are potentially useful for stemming for a number of languages.

## 5.5 Portability

The experimental results in the previous sections show that corpus analysis based on co-occurrence information helps stemming, but do not answer the questions whether stemming is corpus specific and how corpus specific it is.

To answer these questions, we use the equivalence classes generated from the co-occurrence data of one collection to expand the queries on another. We first examine the case that the two collections have similar content. Our conjecture is that, in this case, the equivalence classes generated from a different collection should do equally well. The experimental results support the conjecture. We generate ngramOptimal

```
marked
marketer marketers
marker markers
marketed markets marketing mark marks market
anchored anchors anchor anchoring anchorage anchorages
ancient
ancestor ancestors ancestory
ancestry ancestral
engraving engravings engraver engravers
engagement engagements engageable
engaging engaged engage
payers payer
payout payouts
payday payroll paychecks paycheck payrolls
```

Figure 5.6. Example ngramOptimal classes on WEST

```
mexica mexico mex mexia mexicas mexicos
navieras navales naves naval nava navo navares navismo
 -navidad nave naviera navidades navar navas
```

Figure 5.7. Example S-Porter equivalence classes on TEC4-SPANISH

classes from the WSJ87 data (0.13 GB) and apply them on WSJ91 queries. Table 5.21 shows that the retrieval performance of using WSJ87 classes (WSJ87ngramOptimal) is even slightly better than that of using WSJ91 classes (ngramOptimal).

We then repeat the experiment on two dissimilar collections: the ngramOptimal classes of WSJ are applied on WEST queries. Since WSJ is a collection of newspaper articles with a focus on financial news and WEST is collection of documents of legal

```
mexicana mexicano mexicanos mexicanas mexico
navegaba navigator navio nave navegacion navego naves
 -navieras naviero navegantes navieros navegan navios
 -naval navales naviera navimin naveguen navegaron
 -navegar navegables
```

Figure 5.8. Example ngramCC equivalence classes on TREC4-SPANISH

```
mexicana mexicano mexicanos mexicanas mexico
navales naval nave naves navego naviera navimin
```

Figure 5.9. Example ngramOptimal equivalence classes on TREC4-SPANISH

Table 5.20. Retrieval performance on TREC4-SPANISH

| | Precision (% change) – 25 queries | | |
|---|---|---|---|
| Recall | S-Porter | ngramCC | ngramOptimal |
| 0 | 79.4 | 80.5 (+1.4) | 82.1 (+3.4) |
| 10 | 48.9 | 49.6 (+1.4) | 50.0 (+2.4) |
| 20 | 39.3 | 39.6 (+0.8) | 41.1 (+4.6) |
| 30 | 31.8 | 33.7 (+6.0) | 33.8 (+6.1) |
| 40 | 26.4 | 26.8 (+1.7) | 27.8 (+5.2) |
| 50 | 21.6 | 21.9 (+1.3) | 20.8 (−3.9) |
| 60 | 16.4 | 16.9 (+2.7) | 17.1 (+4.3) |
| 70 | 12.6 | 12.6 (+0.2) | 12.8 (+2.1) |
| 80 | 9.1 | 9.2 (+1.1) | 9.5 (+4.7) |
| 90 | 4.9 | 5.0 (+1.4) | 5.2 (+6.5) |
| 100 | 0.7 | 0.5 (−32.5) | 0.5 (−22.4) |
| average | 26.5 | 26.9 (+1.8) | 27.3 (+3.3) |

cases, they should have dissimilar word usage. For example, the word "recoverable" as in the WEST query "Are punitive damages recoverable under 42 U.S.C. 2000E?" is related to words "recover" and "recovered" in WEST and has a high degree of co-occurrence with them. These words are added to the query by WEST equivalence classes. But these words are not added to the query by WSJ classes because "recoverable" is not closely related to these words in WSJ as indicated by a low degree of co-occurrence. Our conjecture is that the dissimilarity in word usage will cause the retrieval performance of WSJ equivalence classes to be worse than that of WEST classes on the WEST queries. The retrieval results in Tables 5.22 and 5.23 support this conjecture. The retrieval performance of WSJ classes is comparable to those of Porter and KSTEM, but not as good as that of WEST classes.

The results show that stemming is somewhat corpus specific. The portability of the equivalence classes depends on the similarity of the corpora.

Table 5.21. Using WSJ87 classes on WSJ91

| | Precision (% change) – 60 queries | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Recall | KSTEM | Porter | | ngramOptimal | | WSJ87ngramOptimal | |
| 0 | 75.8 | 75.5 | (−0.4) | 76.4 | (+0.7) | 76.2 | (+0.5) |
| 10 | 59.8 | 61.6 | (+2.9) | 60.8 | (+1.6) | 60.9 | (+1.9) |
| 20 | 51.5 | 51.6 | (+0.3) | 52.4 | (+1.9) | 52.8 | (+2.7) |
| 30 | 44.5 | 44.7 | (+0.4) | 47.0 | (+5.5) | 47.1 | (+5.9) |
| 40 | 39.1 | 39.2 | (+0.4) | 41.0 | (+5.0) | 41.1 | (+5.3) |
| 50 | 33.0 | 34.2 | (+3.5) | 35.4 | (+7.2) | 35.9 | (+8.7) |
| 60 | 27.8 | 29.1 | (+4.5) | 28.6 | (+2.7) | 29.2 | (+5.0) |
| 70 | 22.6 | 23.9 | (+6.0) | 22.4 | (−0.6) | 22.7 | (+0.8) |
| 80 | 17.7 | 18.4 | (+4.1) | 17.4 | (−1.4) | 17.4 | (−1.8) |
| 90 | 12.2 | 13.2 | (+8.5) | 12.6 | (+3.8) | 12.9 | (+5.6) |
| 100 | 3.8 | 4.4 | (+15.8) | 4.5 | (+17.1) | 4.4 | (+14.0) |
| average | 35.3 | 36.0 | (+2.1) | 36.2 | (+2.8) | 36.4 | (+3.4) |

## 5.6 Efficiency Issues

In this section we discuss the computational costs of applying corpus analysis to stemming. Specifically, we will discuss two sources of cost: the cost of generating the equivalence classes and the cost of retrieval. Timing figures used in this section are based on experiments on the WSJ collection on a SUN4 workstation.

### 5.6.1 Cost of Equivalence Class Generation

The following is a breakdown of the time spent on the steps in Section 5.2.2. Step 1 takes 20 minutes of CPU time to collect the unique word variants in WSJ. This step can be skipped if the unique word variants are extracted from the indexed database. Step 2 takes 1 minute of CPU time to generate the initial equivalence classes. For ngram stemmers, this step is not necessary. Step 3 takes 1 hour 10 minutes of CPU time to collect the co-occurrence data for the word pairs if the initial equivalence classes are generated by Porter. This step takes 1 hour 30 minutes if the initial equivalence classes are generated by trigram matching. Step 4 takes less than 1 minute if the connected component algorithm is used to refine the initial equivalence classes. This step takes 6 minutes if the optimal partition algorithm is used. All

Table 5.22. Using WSJ equivalence classes on WEST natural language queries

| | Precision (% change) – 34 queries | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Recall | KSTEM | Porter | | ngramOptimal | | WSJngramOptimal | |
| 0 | 84.1 | 84.3 | (+0.3) | 87.2 | (+3.7) | 85.4 | (+1.6) |
| 10 | 79.9 | 78.2 | (−2.1) | 79.4 | (−0.6) | 78.8 | (−1.3) |
| 20 | 75.3 | 72.8 | (−3.3) | 75.8 | (+0.6) | 75.5 | (+0.2) |
| 30 | 71.4 | 70.1 | (−1.8) | 71.2 | (−0.4) | 69.4 | (−2.8) |
| 40 | 61.2 | 60.2 | (−1.7) | 61.6 | (+0.7) | 59.8 | (−2.2) |
| 50 | 54.0 | 52.9 | (−2.0) | 54.9 | (+1.6) | 53.5 | (−1.0) |
| 60 | 44.1 | 44.5 | (+0.9) | 44.7 | (+1.4) | 43.5 | (−1.3) |
| 70 | 36.0 | 35.5 | (−1.5) | 37.1 | (+2.9) | 35.8 | (−0.7) |
| 80 | 27.0 | 28.7 | (+6.4) | 28.6 | (+5.9) | 27.8 | (+3.1) |
| 90 | 15.7 | 16.5 | (+4.9) | 16.6 | (+5.9) | 16.4 | (+4.3) |
| 100 | 8.1 | 8.8 | (+7.9) | 9.0 | (+10.7) | 9.3 | (+14.7) |
| average | 50.6 | 50.2 | (−0.8) | 51.5 | (+1.6) | 50.5 | (−0.3) |

steps considered, the most expensive method, ngramOptimal, takes about two hours to generate the final equivalence classes on WSJ. Most of the time is spent collecting the co-occurrence data. A better implementation can cut this cost. Even with the current implementation, two hours of processing time on a $500^{+}$ MB collection is reasonable.

One way to cut the time on collecting the co-occurrence data is to use data from a subset of the collection. Table 5.24 shows the retrieval effectiveness of using parts of the WSJ collection to generate the ngramOptimal classes. Using 10% of WSJ (ngramOptimal-0.1), the retrieval performance is slightly worse than using the whole collection (ngramOptimal). Using 50% of WSJ (ngramOptimal-0.5), the retrieval performance is as good as using all WSJ. This means that it is possible to use a subset of a large collection to generate the equivalence classes without losing much performance. So far we are not sure how much text is necessary in order to achieve the same performance as we get from using the whole collection. But we can state that for collections larger than 500 MB, the time on collecting the co-occurrence data can be cut by at least 50%. For even larger collections (multiple GBs in size), the time can probably be cut further.

Table 5.23. Using WSJ equivalence classes on WEST structured queries

| | Precision (% change) – 34 queries | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Recall | KSTEM | Porter | | ngramOptimal | | WSJngramOptimal | |
| 0 | 87.8 | 88.5 | (+0.9) | 88.1 | (+0.4) | 85.9 | (−2.1) |
| 10 | 80.9 | 80.7 | (−0.2) | 81.0 | (+0.1) | 80.6 | (−0.4) |
| 20 | 75.5 | 74.6 | (−1.1) | 75.7 | (+0.3) | 77.2 | (+2.2) |
| 30 | 72.0 | 70.7 | (−1.9) | 71.9 | (−0.2) | 71.8 | (−0.2) |
| 40 | 63.8 | 63.7 | (−0.2) | 63.7 | (−0.1) | 64.4 | (+1.0) |
| 50 | 57.2 | 57.4 | (+0.3) | 58.0 | (+1.5) | 58.3 | (+1.9) |
| 60 | 49.2 | 49.7 | (+0.9) | 49.7 | (+0.9) | 49.9 | (+1.3) |
| 70 | 41.5 | 41.9 | (+1.0) | 41.6 | (+0.3) | 40.4 | (−2.6) |
| 80 | 30.7 | 33.4 | (+8.7) | 33.2 | (+7.9) | 31.4 | (+2.2) |
| 90 | 19.7 | 20.1 | (+2.3) | 20.6 | (+5.0) | 20.0 | (+1.8) |
| 100 | 9.3 | 9.1 | (−1.7) | 10.5 | (+13.2) | 9.8 | (+5.8) |
| average | 53.4 | 53.6 | (+0.4) | 54.0 | (+1.1) | 53.6 | (+0.4) |

### 5.6.2 Cost of Retrieval

With query based stemming (query expansion using equivalence classes), retrieval time varies with the stemmer used. We found that there is a strong correlation between the expansion factor and retrieval time. For example, INQUERY takes 2 hours 27 minutes of CPU time to finish 66 WSJ queries expanded by Porter and 1 hour 23 minutes of CPU time to finish the same queries expanded by the ngramOptimal stemmer. The expansion factors of the two stemmers on WSJ are 4.5 and 2.28. Since corpus analysis significantly reduces the expansion factor, it can result in much faster retrieval. This is especially desirable for interactive applications.

Query based stemming is, however, less efficient than stemming at document indexing time. It takes only 42 minutes of CPU time to finish the 66 WSJ queries on a WSJ database pre-stemmed by Porter. We think the efficiency gap is largely due to the fact that the version of INQUERY (2.1) we used is not well tuned for query based stemming. For one thing, it merges position information in evaluating the #SYN operator though position information is often unnecessary. For the other, its algorithm to merge inverted lists is particularly inefficient for long equivalence classes. To merge $n$ inverted lists, the algorithm takes $n-1$ rounds. The $i$th round

Table 5.24. Using equivalence classes for 10% and 50% of WSJ on WSJ

| | Precision (% change) – 66 queries | | | | |
|---|---|---|---|---|---|
| Recall | ngramOptimal | ngramOptimal-0.1 | | ngramOptimal-0.5 | |
| 0 | 82.0 | 82.4 | (+0.5) | 83.2 | (+1.4) |
| 10 | 53.2 | 52.6 | (−1.1) | 53.7 | (+1.0) |
| 20 | 46.2 | 45.7 | (−1.1) | 46.0 | (−0.4) |
| 30 | 41.2 | 40.5 | (−1.9) | 41.1 | (−0.3) |
| 40 | 36.9 | 36.5 | (−1.1) | 37.0 | (+0.3) |
| 50 | 31.8 | 31.3 | (−1.6) | 31.8 | (−0.1) |
| 60 | 26.6 | 26.5 | (−0.4) | 26.6 | (+0.1) |
| 70 | 21.4 | 21.2 | (−0.8) | 21.3 | (−0.5) |
| 80 | 16.8 | 16.6 | (−1.2) | 16.8 | (−0.3) |
| 90 | 11.6 | 11.6 | (+0.0) | 11.6 | (−0.5) |
| 100 | 2.6 | 2.5 | (−1.5) | 2.6 | (−0.1) |
| average | 33.7 | 33.4 | (−0.8) | 33.8 | (+0.3) |

merges the merged result for the first $i$ inverted lists with the $i+1$th inverted list. For $n$ inverted lists of length $l$ each, it needs $O(n^2 l)$ operations. A better algorithm used in the merge sort algorithm [11] needs only $O(nl \log n)$ operations. We expect a significant cut in retrieval time with limited modifications to the retrieval system. With a well tuned retrieval system, query based stemming in combination with the corpus analysis technique may be even more efficient than indexing time stemming using an aggressive stemmer such as Porter because corpus analysis can result in much smaller equivalence classes and shorter inverted lists for the query words.

## 5.7 Summary

The experimental results in this chapter on both English and Spanish test collections validate the hypotheses 1 and 2. Corpus based stemming does offer slightly more effective retrieval than traditional stemming. A stemming algorithm based on simple initial character matching and corpus analysis is competitive with traditional stemmers in terms of retrieval effectiveness. Corpus analysis can significantly reduce the number of words in the expanded queries and speed up retrieval. Timing figures show that applying corpus analysis to stemming is also practical.

# Chapter 6

## Corpus Analysis for General Query Expansion: Experiments

In this chapter we will conduct experiments to investigate the usefulness of corpus analysis for general query expansion. In a sense this chapter is an extension of the previous chapter. Unlike stemming, we do not require expansion words to be morphological variants of the query words. In Section 6.1 we will present query expansion experiments using the corpus based approach. In Section 6.2 we will present experiments using the corpus-query based approach. In both sections we will use actual queries to expose the problems with these approaches. In Section 6.3 we will summarize the experiments and draw some conclusions about the value of corpus analysis for general query expansion.

### 6.1 Corpus Based Approach

#### 6.1.1 Data Preparation

The experiments in this section are conducted on the TREC4 collection. Before we report the experimental results, we will first describe how we prepare the data for the experiments.

##### 6.1.1.1 Query Word Extraction

As we discussed in Chapter 4, the corpus based approach expands a query word by adding its nearest neighbors. We need to define what a query word is. In stemming, we define query words as individual terms in a query. This definition has a problem for general query expansion. For example, if a query contains "United States", we do not want to expand "United" and "States" as two query words. Rather, we want

to treat them as a single query word. If a query is described in natural language, we need to process the query text properly and recognize the query words. This is a tricky problem because sometimes several words should be treated as a phrase. Fortunately the TREC4 queries we will work on are already in a parsed format [1]. Therefore we avoid the problem of recognizing the query words. Figure 6.1 shows one of the TREC4 queries (in the INQUERY query language). The query contains both phrases and single terms. Removing stop words, query operators and weights, the query words in the query are "affirmative", "action", "affected", "construction", "industry", "affirmative action" and "construction industry". A phrase in a query is treated as single query word rather than several words.

```
#WSUM (1.0
   1.0 How 1.0 has 1.0 affirmative 1.0 action 1.0 affected 1.0 the
   1.0 construction 1.0 industry
   1.0 #PHRASE(affirmative action) 1.0 #PHRASE(construction industry)
   ) ;
```

Figure 6.1. An example parsed query

#### 6.1.1.2 Concept Recognition

Past research work suggests that nouns and noun phrases are more informative than other types of words and phrases and are better units for query expansion [29]. For this reason, we will use nouns and noun phrases for query expansion in the experiments. The nearest neighbors of a query word are therefore nouns and noun phrases. To conform with the terminology used by other researchers, we call these nouns and noun phrases *expansion concepts*, or *concepts* in short. In our experiments, a concept is a single noun or a sequence of consecutive nouns recognized by a part speech tagger, *Jtag* [73]. Plural nouns are converted to the singular form. Figure 6.2

shows a sample document tagged by the tagger. Concepts are put in the brackets ([ ]).

```
The/AT [bill/NN] has/HVZ been/BEN reworked/VBN
since/CS it/PPS was/BEDZ introduced/VBN ,/,
in/IN [order/NN] to/TO meet/VB some/DTI
[employer/NN objections/NNS] ./. But/CC the/AT
[measure/NN] still/RB is/BEZ opposed/VBN by/IN
the/AT [construction/NN industry/NN] ,/, which/WDT
argues/VBZ that/CS it/PPS would/MD impose/VB
[unionism/NN] and/CC higher/JJ [costs/NNS] on/IN
much/AP of/IN the/AT [industry/NN] 's/$
[work/NN] ./.
```

Figure 6.2. A sample tagged document with concepts bracketed

#### 6.1.1.3 Co-occurrence Counting and *em* Calculation

To avoid problems with long documents, which are often about several different topics, documents are broken up into non-overlapping, fixed size text windows. We call these text windows *passages*. Co-occurrences are determined using passages rather than documents. In our experiments, passage size is 300 words, which has been shown to be an appropriate passage size by previous work in document retrieval using passage level evidence [8].

The frequency of a query word or a concept in the collection is defined as the number of passages containing it. The co-occurrence frequency of a query word and a concept is the number of passages containing both. The *em* score of a query word $w$ and a concept $c$ is calculated as,

$$em(w,c) = max(\frac{n_{wc} - En(w,c)}{n_w + n_c}, 0)$$

$$En(w,c) = n_w n_c / N$$

where $N$ is the number of passages in the collection, $n_w$, $n_c$ and $n_{wc}$ are the frequencies of $w$ and $c$ and their co-occurrence frequency.

Figure 6.3 lists some concepts for each query word in the example query in Figure 6.1 (the numbers are *em* scores with the query words). In query expansion, these concepts will be added to the query and combined with the query words using the weighted OR operator (see Chapter 4). The *em* scores are used as weights inside the operator. For example, the query word "affirmative action" is expanded to

```
#WOR(1.0  1.0     #PHRASE(affirmative action)
          0.13    #PHRASE(action program)
          0.046   minority
          0.04    quota
          0.039   #PHRASE(clarence thomas)
          0.039   discrimination
          ...
     )
```

where #WOR is the weighted OR operator, #PHRASE is an INQUERY operator to construct a phrase from single terms.

### 6.1.2 Experimental Results

Table 6.1 shows the retrieval performance of the corpus based approach as compared with the baseline queries on TREC4. The baseline queries are the unexpanded ad hoc queries used by CIIR UMass in the TREC4 conference. In the table, nn-5 means adding the top 5 concepts (nearest neighbors) for each query word, nn-10 adding the top 10 concepts and so forth. Adding 5 nearest neighbors for each query word produces a 4.2% percent improvement over the base line. The t-test indicates that the performance difference between the baseline and nn-5 is statistically significant (p_value=0.0017). Improvements at high recall points are mainly responsible for the improvement. The precision at recall point 0 is almost unchanged. This is understandable because query expansion is largely a recall device.

```
affirmative: affirmation 0.08, action-program 0.04, minority 0.03,
discrimination 0.028, appeals-court 0.024

action: summary 0.097, notice 0.093, dc 0.08, rule 0.07, washington 0.07

affected:   entity 0.069, cfr 0.064, rin 0.057, action-date-fr 0.056,
timetable 0.054

construction: project 0.045, building 0.035, facility 0.03,
housing 0.02, site 0.02

industry: market 0.073, company 0.069, analyst 0.054, stock 0.052,
share 0.051

affirmative-action: action-program 0.13, minority 0.046, quota 0.04,
clarence-thomas 0.039, discrimination 0.039

construction-industry: wage-law 0.037, fringe-benefit-information 0.036,
construction-industry-wage-determination 0.036, wage-determination 0.035
```

Figure 6.3. Nearest neighbors of the query words of the query shown in Figure 6.1

In Chapter 4, we said that using the nearest neighbors is a tradeoff between efficiency and effectiveness. We expected that the retrieval performance would keep increasing as more nearest neighbors are used though after a certain point the rate of improvement would be too small to be useful. But the data in Table 6.1 tells a different story. As more concepts are used, the average precision actually drops slightly. The performance loss may have something to do with the weights of the concepts we assign to them. Please note that we use the *em* score to estimate the probability that a concept is about a query word. The estimation, though reasonable, may be too crude. A casual examination of the expansion concepts and the *em* scores seems to show that the quality of a concept is not proportional to its *em* score with the query word. Rather, the quality of a concept seems to drop more quickly than its *em* score. In other words, concepts with small *em* scores deserve smaller weights than we assign to them. This problem becomes more severe when more concepts are used.

Table 6.1. Retrieval performance of the corpus based query approach on TREC4

| | Precision (% change) – 49 queries | | | | |
|---|---|---|---|---|---|
| Recall | baseline | nn-5 | nn-10 | nn-20 | nn-40 |
| 0 | 71.0 | 71.5 (+0.8) | 67.9 (−4.4) | 65.6 (−7.6) | 64.7 (−8.8) |
| 10 | 49.3 | 50.0 (+1.3) | 49.9 (+1.0) | 50.0 (+1.3) | 50.4 (+2.1) |
| 20 | 40.4 | 41.9 (+3.8) | 42.2 (+4.6) | 42.7 (+5.9) | 42.9 (+6.2) |
| 30 | 33.3 | 34.5 (+3.5) | 34.8 (+4.5) | 34.7 (+4.2) | 34.7 (+4.3) |
| 40 | 27.3 | 28.8 (+5.3) | 29.2 (+6.8) | 28.9 (+5.8) | 29.0 (+6.2) |
| 50 | 21.6 | 24.0 (+11.1) | 24.0 (+10.8) | 23.8 (+10.3) | 24.1 (+11.4) |
| 60 | 14.8 | 16.8 (+13.7) | 17.1 (+15.5) | 17.2 (+16.2) | 17.3 (+17.0) |
| 70 | 9.5 | 10.3 (+8.9) | 10.4 (+9.4) | 10.7 (+12.7) | 10.8 (+14.0) |
| 80 | 6.2 | 6.8 (+10.7) | 6.9 (+12.0) | 7.2 (+16.2) | 7.2 (+17.0) |
| 90 | 3.1 | 3.5 (+10.8) | 3.6 (+14.6) | 3.6 (+12.7) | 3.4 (+8.5) |
| 100 | 0.4 | 0.5 (+13.2) | 0.5 (+19.1) | 0.5 (+21.2) | 0.5 (+16.4) |
| average | 25.2 | 26.2 (+4.2) | 26.0 (+3.4) | 25.9 (+2.9) | 25.9 (+2.9) |

A suitable transformation from the *em* score to a weight could avoid the problem. The performance figures in Table 6.1 also show that as more concepts are used, the performance change is not uniform across all recall points. Though the precision at recall point 0 drops significantly, precision at other recall points generally improves when more concepts are used. This is understandable because when more concepts are used, the expanded queries can match with more relevant documents and hence improve recall.

Using the weighted OR operator to combine evidence from the query words and the expansion concepts are better than using two alternative operators, INQUERY's synonym and weighted SUM operators, for the same purpose. Table 6.2 shows the retrieval performance of using the 3 operators for query expansion. In the table, the results of the weighted OR, weighted SUM and synonym operators are denoted nn-5, nn-5-wsum and nn-5-syn respectively. In each case the top 5 concepts are used for each query word. The synonym operator, which treats the occurrences of the expansion concepts in the documents exactly the same as the occurrences of the original query word, results in a significant performance loss over the base line. This means that weighting the expansion concepts is necessary. The fact that weighted OR is slightly

better than weighted SUM suggests that it characterizes the relationship between the original query words and their nearest neighbors better.

Table 6.2. Comparing the retrieval performance of WOR, WSUM and SYN to combine query words and nearest neighbors

| | Precision (% change) – 49 queries | | | |
|---|---|---|---|---|
| Recall | baseline | nn-5 | nn-5-wsum | nn-5-syn |
| 0 | 71.0 | 71.5 (+0.8) | 72.8 (+2.6) | 64.4 (−9.3) |
| 10 | 49.3 | 50.0 (+1.3) | 49.9 (+1.2) | 43.4 (−12.0) |
| 20 | 40.4 | 41.9 (+3.8) | 41.2 (+2.0) | 37.2 (−7.9) |
| 30 | 33.3 | 34.5 (+3.5) | 34.3 (+3.0) | 29.8 (−10.6) |
| 40 | 27.3 | 28.8 (+5.3) | 28.1 (+2.9) | 24.8 (−9.3) |
| 50 | 21.6 | 24.0 (+11.1) | 22.4 (+3.4) | 19.3 (−10.7) |
| 60 | 14.8 | 16.8 (+13.7) | 15.6 (+5.5) | 14.6 (−1.1) |
| 70 | 9.5 | 10.3 (+8.9) | 9.9 (+4.3) | 9.4 (−0.2) |
| 80 | 6.2 | 6.8 (+10.7) | 6.3 (+3.1) | 6.4 (+4.3) |
| 90 | 3.1 | 3.5 (+10.8) | 3.1 (−0.6) | 3.5 (+10.3) |
| 100 | 0.4 | 0.5 (+13.2) | 0.5 (+8.1) | 0.9(+109.1) |
| average | 25.2 | 26.2 (+4.2) | 25.8 (+2.6) | 23.1 (−8.4) |

### 6.1.3 Analysis

The experimental results show that the corpus based approach is indeed helpful in improving retrieval performance. But the improvement (4.2%) is too small to be cost effective for such an expensive corpus based approach.

We think our results establish a upper bound (or nearly a upper bound) for any approach that exploits corpus analysis for general query expansion for several reasons. Firstly our results are based on a very large collection (TREC4, 2 Gigabytes). Most earlier studies used small collections, which may be a cause for the retrieval failure in them. Secondly, we use a better metric to choose expansion concepts and a better model to combine evidence from query words and expansion concepts than earlier studies. Thirdly, we expand a phrase in a query as a whole rather than expanding the terms in the phrase independently as in most earlier studies. Lastly, we emphasize

adding more content bearing nouns and noun phrases to the queries rather than just words.

The incorrect expansion concepts can be roughly broken down into the following categories:

1. Expansion concepts of ambiguous query words. For example, the meaning of "plant" in a query about "nuclear power plants and their rate of production" is different from its meaning in the query about "conversion of plant material to energy". The nearest neighbors of "plant" are related to different meanings of "plant". Those related to one meaning are "production", "reactor", "GM", "worker" and so on. Those related to the other meaning are "tree", "crop" and so on. The other example is the word "intelligence" in the query "search for extraterrestrial life and intelligence". Its nearest neighbors include such words as "CIA", "AI" and "expert system".

2. Expansion concepts related to the query words in general but not in the context of the query. As an example, we consider the word "DNA" in the query "as a result of DNA testing, are more defendants being absolved or convicted of crimes". Its nearest neighbors include such words as "gene", "DNA sequence", "chromosome", "NIH", "DNA molecule" and so on. In general, these concepts are related to "DNA". But for the above query about the forensic application of DNA test, they are not good expansion words for "DNA". Adding them to the query will bring documents about "DNA" in the biological and medical contexts closer to the top of the ranked output and hurt retrieval performance.

3. Expansion concepts of non-content bearing query words. The word "result" in the above query is obviously not a content bearing word. Its nearest neighbors include "quarter", "loss", "profit", "share" and so on.

Though all the above categories hurt retrieval performance, categories 1 and 2 cause the most damage. Incorrect expansion concepts in categories 1 and 2 are more

common than those in category 3. The expansion concepts of non-content query words are usually totally unrelated to the query words and are generally random. Therefore, they generally have a small impact on the ranked output for the query. The expansion concepts in categories 1 and 2, however, are related to the query words in some way and are less random. As the result, they are more capable of bringing non-relevant documents ahead of relevant documents in the ranked output.

Categories 1 and 2 are caused by the same problem: query words are treated independently and the specific query context is ignored. This problem limits the value of the corpus based approach for query expansion.

## 6.2 Corpus-query Based Approach

To address the problem with the corpus based approach, the corpus-query based approach employs query context in addition to corpus analysis for query expansion. It selects expansion concepts based on their co-occurrence with all query words rather than with individual query words. In this section we conduct experiments using the corpus-query based approach on TREC3 and TREC4.

### 6.2.1 Description of Experiments

As in the corpus based approach, we break documents into fixed size passages. Each passage has 300 words. The procedure used to recognize concepts from passages before is also used here. The frequency of a query word and the co-occurrence frequency of a query word and a concept are counted the same as before. Unlike the corpus based experiments, query words are now defined as terms in the queries to simplify experiments.

We use the following function $f(c, Q)$ to measure the suitability of a concept $c$ for query $Q$. We suppose the query words in $Q$ are $w_1, w_2, ...w_n$.

$$f(c,Q) = \prod_{w_i \text{ in Q}} (0.01 + co\_degree(c,w_i))^{idf(w_i)}$$

$$co\_degree(c,w) = max(\frac{n_{cw} - En(c,w) - 1}{n_c}, 0)$$

$$En(c,w) = \frac{n_w n_c}{N}$$

$$idf(w) = min(1.0, log_{10}(N/n_w)/5)$$

where $N$ is the number of passages in the collection, $n_w$, $n_c$ and $n_{wc}$ are the frequencies of $w$ and $c$ and their co-occurrence frequency. The function $f$ is similar to the one in Section 4.3 and has similar motivations.

The top 30 concepts are used for each query. The way we use the expansion concepts is (in INQUERY's query language):

$$\begin{array}{lcl} Q_{new} & = & \#\text{WSUM}(1.0\ 1.0\ Q\ 1.0\ Q\prime) \\ Q\prime & = & \#\text{WSUM}(1.0\ 1.0\ c_1\ 1.0\ c_2\ ...\ 1.0\ c_{30}) \end{array}$$

The formulas mean that the 30 expansion concepts form an auxiliary query $Q\prime$ with equal weights. The original query and the auxiliary query are combined with equal weights to form the expanded query.

We should mention that the corpus-query based approach we just described is very similar to Phrasefinder [29]. The major difference is in the functions used for concept selection. Phrasefinder used standard document-query similarity for concept selection. We have pointed out the problem with this function in Chapter 2. The corpus-query based approach uses a better motivated function for concept selection.

### 6.2.2 Experimental Results and Analysis

Tables 6.3 and 6.4 show the retrieval performance of the corpus-query based approach (labeled as corpus-query) as compared with the unexpanded baseline queries. The corpus-query based approach produces significant improvements in retrieval effectiveness over the baseline queries on both collections: 17% on TREC3 and 13.7%

on TREC4. The t-test indicates that the improvements are statistically significant (p_value=0.00001 on TREC3 and p_value=0.00016 on TREC4).

Table 6.3. Retrieval performance of the corpus-query based approach on TREC3

| | Precision (% change) – 50 queries | | |
|---|---|---|---|
| Recall | baseline | Phrasefinder | corpus-query |
| 0 | 82.2 | 79.4 (−3.3) | 85.3 (+3.8) |
| 10 | 57.3 | 60.1 (+4.8) | 65.1 (+13.5) |
| 20 | 46.2 | 50.4 (+9.1) | 54.7 (+18.5) |
| 30 | 39.1 | 43.3 (+10.7) | 46.8 (+19.9) |
| 40 | 32.7 | 36.9 (+12.8) | 40.0 (+22.1) |
| 50 | 27.5 | 31.8 (+15.9) | 34.6 (+25.9) |
| 60 | 22.6 | 26.1 (+15.1) | 28.4 (+25.2) |
| 70 | 18.0 | 20.6 (+14.0) | 23.0 (+27.3) |
| 80 | 13.3 | 15.8 (+18.6) | 17.4 (+30.7) |
| 90 | 7.9 | 9.4 (+18.7) | 10.7 (+34.4) |
| 100 | 0.5 | 0.8 (+60.9) | 0.7 (+36.9) |
| average | 31.6 | 34.1 (+7.8) | 37.0 (+17.0) |

Since the corpus-query based approach is similar to Phrasefinder except that the functions to choose the expansion concepts are different, we also include the Phrasefinder results in the tables. In the Phrasefinder queries, 30 concepts are added into each query and are weighted in proportion to their rank position. Phrases containing only terms in the original query are weighted more heavily than those containing terms not in the original query. Despite their similarity, the performance of our corpus-query based approach is significantly better than that of Phrasefinder (p_value=0.005 on TREC3 and p_value=0.003 on TREC4). The results indicate that our function $f$ for concept selection is better than that of Phrasefinder.

An examination of the expansion concepts reveals that the corpus-query based approach largely solves the problem with the corpus based approach. In Figure 6.4 we list the top 30 concepts for the TREC4 query 213, "as a result of DNA testing, are more defendants being absolved or convicted of crimes". These concepts are obviously better than the concepts added by the corpus based approach for the word

Table 6.4. Retrieval performance of the corpus-query based approach on TREC4

| | Precision (% change) – 49 queries | | |
|---|---|---|---|
| Recall | baseline | Phrasefinder | corpus-query |
| 0 | 71.0 | 68.6 (−3.3) | 70.4 (−0.8) |
| 10 | 49.3 | 48.6 (−1.6) | 54.3 (+10.0) |
| 20 | 40.4 | 40.0 (−1.0) | 45.0 (+11.6) |
| 30 | 33.3 | 33.9 (+1.8) | 37.7 (+13.4) |
| 40 | 27.3 | 28.0 (+2.5) | 32.6 (+19.4) |
| 50 | 21.6 | 23.9 (+10.3) | 27.4 (+26.7) |
| 60 | 14.8 | 18.8 (+27.1) | 20.8 (+40.7) |
| 70 | 9.5 | 11.8 (+24.7) | 13.6 (+43.3) |
| 80 | 6.2 | 8.1 (+31.0) | 8.2 (+33.8) |
| 90 | 3.1 | 4.2 (+33.6) | 4.2 (+34.2) |
| 100 | 0.4 | 0.6 (+24.0) | 0.6 (+36.7) |
| average | 25.2 | 26.0 (+3.4) | 28.6 (+13.7) |

"DNA". Subtle difference in word usage sometimes can have a significant impact on the relevance of a document. The concepts "DNA-pattern", "DNA-evidence" and "DNA-profile" seem to be synonyms with "DNA-sequence", a nearest neighbor of "DNA" found by the corpus based approach. But "DNA-sequence" is more likely to be used in biological and medical documents and is a poor indicator of relevance for the query. In contrast, "DNA-pattern", "DNA-evidence" and "DNA-profile" are more likely to be used in articles about DNA in law suits and are good indicators of relevance. A similar phenomenon is observed by Riloff in the context of information extraction [49].

### 6.2.3 How Valuable is Global Co-occurrence Information?

Experimental results show that the corpus-query based approach can significantly improve retrieval performance. But as we pointed out in Section 4.2.4, many of the expansion concepts used by the corpus-query based approach are likely to be from the top ranked documents retrieved for the original query. The improvement in retrieval performance may be largely due to the concepts from the top ranked documents. Our conjecture is that the performance of the corpus-query based approach largely

```
dna-pattern  dna-testing  lifecodes dna-test-result
dna-lab   dna-evidence   dna-test   dna-profile
dna-identification   cellmark cellmark-diagnostics
defense-attorneys-challenging-reliability   bureau-expert
lawyer-peter-neufeld   new-york-city-murder-case   michael-baird
procedures-track-record   dna-laboratory   oregon-rape-case
mark-storolow   laboratory-gelatin   supermarket-merchandise
thomas-caskey   procedures-lifecodes   price-code-bar
lifecodes-corp   tests-reliability   maine-case
rape-conviction   dna-strand
```

Figure 6.4. Concepts selected by corpus–query based approach for TREC4 query 213

depends on the top ranked documents (passages). To confirm the conjecture, we need to see how well the corpus-query based approach works if it is applied to the documents ranked low for the original query. The experiments are carried out on TREC4. The procedure to expand a query $Q$ is this:

1. Run INQUERY to rank all passages in the collection for $Q$.
2. Remove the top $n$ passages from the collection. This results in a slightly smaller collection consisting of $1084310 - n$ passages.
3. Apply the corpus-query based approach on the reduced collection to expand $Q$.

The above procedure is repeated for each TREC4 query. Figure 6.5 shows the retrieval performance of the corpus-query based approach on TREC4 when the top ranked passages are removed. The performance drops as more top ranked passages are removed. When 500 passages are removed for each query, the performance is only 2.5% better than the baseline (Table 6.5). Note that 500 passages only represent a small fraction of the $1^+$ million passages in the collection. For 99% of the query words, at least 50% of their occurrences are in passages other than the top 500 ones. For 92% of the query words, at least 90% of their occurrences are in passages other than the top 500 ones. In other words, when we exclude the top 500 passages, we still have sufficient data about most of the query words.

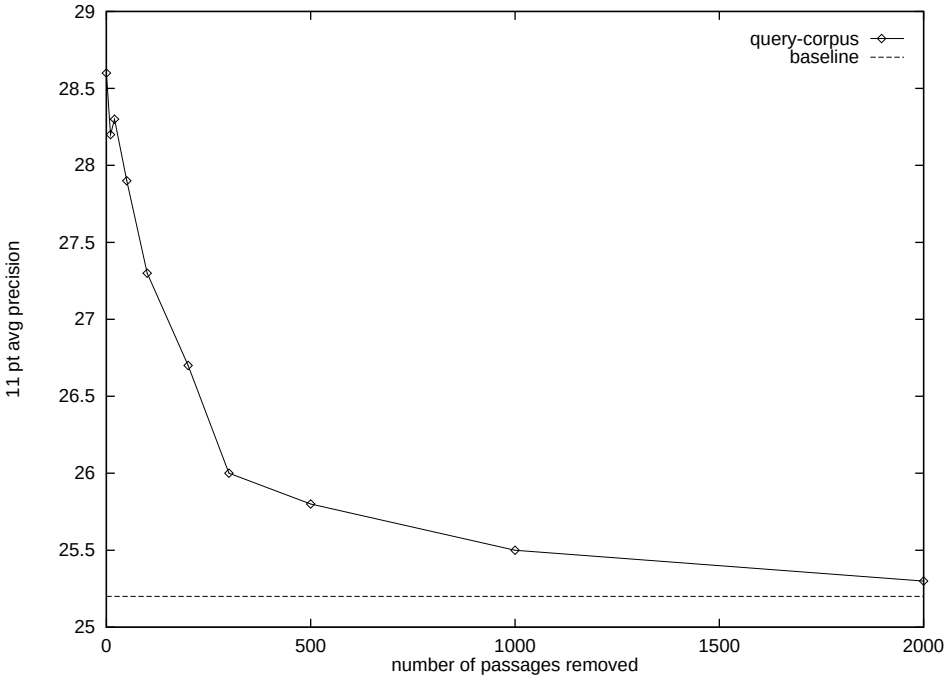


Figure 6.5. Retrieval performance of the corpus-query based approach when top ranked passages are removed

The results demonstrate that the performance of the corpus-query based approach largely depends on the information in the top ranked passages (documents). The vast amount of data outside the top ranked passages has little value for query expansion. The results also seem to suggest that the corpus-query based approach, as well as similar approaches such as Phrasefinder [29] and Concept Based Query Expansion [47], is largely a local method despite the fact it uses global co-occurrence information. This conclusion has a major implication: for effective query expansion, we can just use the information in the top ranked documents. Using the top ranked documents is more efficient. Though we lose the benefit of the global information, which is limited anyway, we avoid the potential harm caused by a large amount of noise in the global information. So using the top ranked documents may be more effective too.

## 6.3 Summary

The results in this chapter demonstrate that corpus analysis based on global co-occurrence information has limited value for general query expansion. The results

Table 6.5. Retrieval performance of the corpus-query based approach on TREC4 when 500 top ranked passages are removed for each query

| | Precision (% change) – 49 queries | | |
|---|---|---|---|
| Recall | baseline | remove-500-passages | |
| 0 | 71.0 | 68.7 | (−3.2) |
| 10 | 49.3 | 50.9 | (+3.1) |
| 20 | 40.4 | 42.0 | (+4.2) |
| 30 | 33.3 | 33.4 | (+0.3) |
| 40 | 27.3 | 28.2 | (+3.2) |
| 50 | 21.6 | 22.8 | (+5.3) |
| 60 | 14.8 | 16.9 | (+14.2) |
| 70 | 9.5 | 10.6 | (+11.7) |
| 80 | 6.2 | 6.7 | (+9.0) |
| 90 | 3.1 | 3.3 | (+4.5) |
| 100 | 0.4 | 0.5 | (+7.9) |
| average | 25.2 | 25.8 | (+2.5) |

also show that despite the fact the corpus-query based approach makes use of global co-occurrence information, its success largely depends on the information in the top ranked documents. The global co-occurrence information outside the top ranked documents has little value for query expansion. We therefore confirm hypotheses 3 and 4.

# Chapter 7

# Local Context Analysis: Experiments

## 7.1 Introduction

The results in Chapter 6 show that global co-occurrence information has very limited value for general query expansion. In this chapter we will conduct experiments to investigate local methods, which make use of the information in the top ranked documents retrieved for the original query for query expansion. The local methods are local feedback [2, 13], a technique which has been around for a long time, and local context analysis, a new technique proposed in this thesis. We will focus on local context analysis and use local feedback primarily for the purpose of comparison. To compare the local methods with the global methods, results of Phrasefinder and the corpus-query based approach on some collections are also reported. The experiments are carried out on TREC3, TREC4, TREC5, WEST, TIME, TREC5-SPANISH, and TREC5-CHINESE. We will use the original queries as the baseline. On WEST, we will use the natural language query set [1].

The rest of this chapter is organized as follows: Section 7.2 lays out the procedure for the experiments. Section 7.3 reports and analyzes the results obtained on the English collections. Section 7.4 reports results on the Spanish and Chinese collections. Section 7.5 discusses some efficiency issues of local context analysis. Section 7.6 summarizes the results and draws some conclusions.

[1] Due to the difference in the versions of INQUERY used, the WEST results in this chapter are not directly comparable with those in Chapter 4.

## 7.2 Description of Experiments

### 7.2.1 Local Feedback

The local feedback experiments are based on the local feedback procedure used by the Cornell group in the TREC conferences [5]. It represents one of the most successful techniques used in the TREC conferences. To expand a query on a collection, an initial retrieval is run for the query on the collection. The most frequent 50 terms and 10 phrases (pairs of adjacent non-stop words) from the top ranked documents or passages are added to the query. The terms in the expanded query are then reweighted using the Rocchio [51] formula with $\alpha : \beta : \gamma = 1 : 1 : 0$. The default number of documents or passages used for feedback is 10, though we sometimes change the number to observe the effect on the retrieval performance.

### 7.2.2 Local Context Analysis

Below are the steps taken by local context analysis to expand a query $Q$ on a collection:

1. Use a standard IR system (INQUERY) to retrieve the top $n$ ranked passages. A passage is a text window of fixed size. There are two reasons that we use passages rather than whole documents. Long documents are usually about multiple topics. Breaking them into smaller pieces avoids the problem of using words from unrelated parts of the documents for query expansion. It is also more efficient to use passages by eliminating the cost of processing the unnecessary parts of the documents.

   The default passage size is 300 words, though we sometimes vary the passage size to observe the effect on retrieval performance.

2. Concepts in the top ranked passages are ranked according to the formula $f(c, Q)$ discussed in Section 4.3.

Different numbers of passages are used to observe the effect on retrieval. In the experiments on the English collections, the default definition of a concept is a noun or a sequence of nouns recognized by Jtag. The same procedure for concept recognition in Chapter 6 is used here. Alternatively, we use terms and pairs of terms in the top ranked passages as concepts. The default $\delta$ value in the formula $f(c, Q)$ is 0.1. Other values are used to see the effect on retrieval performance.

3. Add $m$ top ranked concepts to $Q$ using the following weighting formula:

$$\begin{aligned} Q_{new} &= \#\text{WSUM}(1.0\ 1.0\ Q\ w\ Q\prime) \\ Q\prime &= \#\text{WSUM}(1.0\ w_1\ c_1\ w_2\ c_2\ ...\ w_m\ c_m) \end{aligned}$$

where $c_i$ is the $i$th ranked concept. We call $Q\prime$ the auxiliary query. The default parameter setting is $m = 70$, $w = 2.0$ and $w_i = 1.0 - 0.9 * i/70$.

## 7.3 Experimental Results

### 7.3.1 Local Context Analysis

Table 7.1 shows the performance of local context analysis on the TREC4, TREC3 and WEST. The results labeled TREC4, TREC3 and WEST are obtained using the default parameter values. The results labeled WEST2 are obtained by letting the weight of the auxiliary query be 1.0 instead of the default 2.0.

Local text analysis performs very well on TREC3 and TREC4. All runs produce significant improvements over the baseline. The best run on TREC4 (100 passages) is 23.5% better than the baseline. The best run on TREC3 (200 passages) is 24.4% better than the baseline. The t-test indicates that the improvements are statistically significant (p_value=0.00000006 on TREC4 and p_value=0.000005 on TREC3).

On WEST, the improvements over the baseline are not as good as on TREC3 and TREC4. With too many passages, the performance of local context analysis is even worse than the baseline. But as we will discuss later, the results are still better than local feedback and Phrasefinder. The best run is 3.0% better than the

Table 7.1. Performance of local context analysis using 11 point average precision. The expansion concepts are downweighted by 50% for WEST2

| number of passages used | TREC4 | TREC3 | WEST | WEST2 |
|---|---|---|---|---|
| baseline | 25.2 | 31.6 | 53.8 | 53.8 |
| 10 | 29.5(+17) | 36.6(+16) | 54.8(+1.9) | 55.9(+3.8) |
| 20 | 29.9(+18.6) | 37.5(+18.9) | 55.4(+3.0) | 56.5(+5.0) |
| 30 | 30.2(+19.8) | 38.7(+22.6) | 54.5(+1.3) | 55.6(+3.4) |
| 50 | 30.4(+20.6) | 38.9(+23.2) | 54.2(+0.7) | 55.8(+3.7) |
| 100 | 31.1(+23.5) | 38.9(+23.3) | 54.2(+0.8) | 55.6(+3.3) |
| 200 | 31.0(+23.0) | 39.3(+24.4) | 53.1(-1.3) | 54.6(+1.6) |
| 300 | 30.7(+21.8) | 39.1(+23.7) | 52.7(-2.0) | 54.4(+1.2) |
| 500 | 29.9(+18.6) | 38.3(+21.3) | 52.1(-3.2) | 53.6(-0.4) |
| 1000 | 29.0(+15) | 37.6(+19) | 51.7(-3.9) | 53.7(-0.1) |
| 2000 | 27.9(+10.7) | 36.6(+16.0) | 51.7(-3.9) | 53.7(-0.1) |

baseline. The t-test indicates the improvement is not statistically significant. The high baseline of the WEST collection (53.8% average precision) suggests that the original queries are of very good quality and we should give them more emphasis. Better results are obtained by reducing the weight of auxiliary query $Q\prime$ from 2.0 to 1.0 (WEST2). After downweighting the expansion concepts, the best run (20 passages) is 5.0% better than the baseline. The t-test indicates the improvement is statistically significant (p_value=0.03).

It is interesting to see how the number of passages used affects retrieval performance. To see it more clearly, we plot the performance curve of local context analysis on TREC4 in Figure 7.1. Initially, increasing the number of passages quickly improves performance. The performance peaks at a certain point. After staying relatively flat for a period, the performance curve drops slowly when more passages are used. For TREC3 and TREC4, the optimal number of passages is around 100, while on WEST, the optimal number of passages is around 20. This is not surprising because the first two collections are an order of magnitude larger and have far more relevant documents per query than WEST. Currently we do not know how to automatically determine the optimal number of passages to use. Fortunately, unlike local feedback,

local context analysis is very robust and seems relatively insensitive to the number of the passages used, especially for large collections like the TREC collections. On the TREC collections, between 30 and 300 passages produces very good retrieval performance.

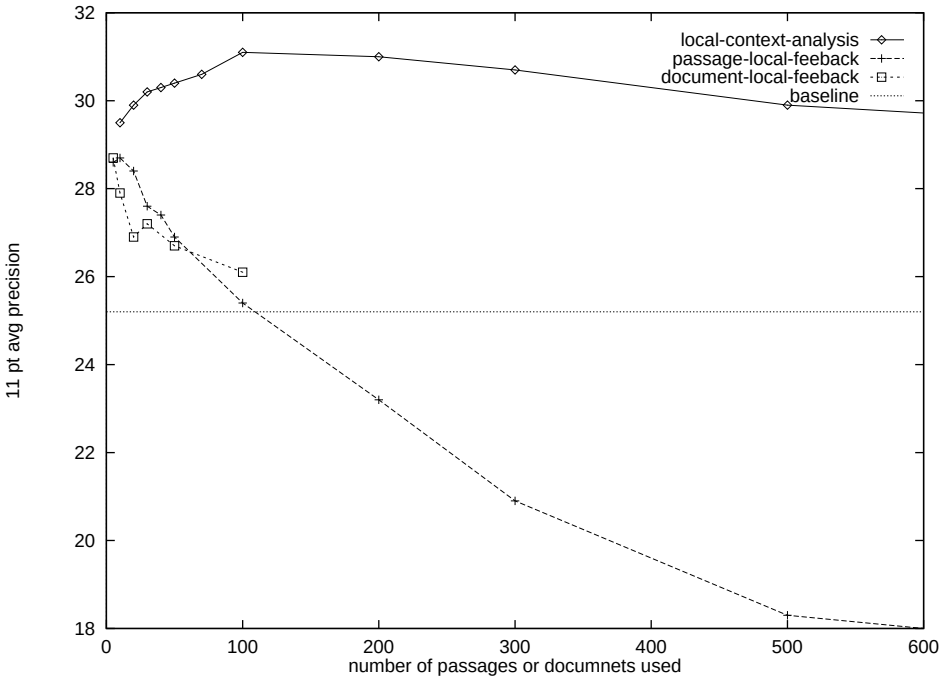


Figure 7.1. Performance curves of local context analysis and local feedback on TREC4.

Figure 7.2 shows the top 30 concepts chosen by local context analysis for TREC4 query 214 "What are the different techniques used to create self induced hypnosis".

| | | |
|---|---|---|
| hypnosis | brain-wave | ms.-burns |
| technique | pulse | reed |
| brain | ms.-olness | trance |
| hallucination | process | circuit |
| van-dyck | behavior | suggestion |
| case | spiegel | finding |
| hypnotizables | subject | van-dyke |
| patient | memory | application |
| katie | muncie | approach |
| study | point | contrast |

Figure 7.2. Local context analysis concepts for query 214

Table 7.2. Retrieval Performances on TREC3. 10 documents are used for local feedback (lf-10doc). 100 passages are used for local context analysis (lca-100p).

| | Precision (% change) – 50 queries | | | | |
|---|---|---|---|---|---|
| Recall | base | Phrasefinder | corpus-query | lf-10doc | lca-100p |
| 0 | 82.2 | 79.4 (−3.3) | 85.3 (+3.8) | 82.5 (+0.4) | 87.0 (+5.9) |
| 10 | 57.3 | 60.1 (+4.8) | 65.1 (+13.5) | 64.9 (+13.3) | 65.5 (+14.3) |
| 20 | 46.2 | 50.4 (+9.1) | 54.7 (+18.5) | 56.1 (+21.5) | 57.2 (+23.8) |
| 30 | 39.1 | 43.3 (+10.7) | 46.8 (+19.9) | 48.3 (+23.5) | 48.4 (+23.8) |
| 40 | 32.7 | 36.9 (+12.8) | 40.0 (+22.1) | 41.6 (+26.9) | 42.7 (+30.4) |
| 50 | 27.5 | 31.8 (+15.9) | 34.6 (+25.9) | 36.8 (+34.1) | 37.9 (+38.0) |
| 60 | 22.6 | 26.1 (+15.1) | 28.4 (+25.2) | 30.9 (+36.7) | 31.5 (+39.3) |
| 70 | 18.0 | 20.6 (+14.0) | 23.0 (+27.3) | 25.2 (+40.0) | 25.6 (+42.1) |
| 80 | 13.3 | 15.8 (+18.6) | 17.4 (+30.7) | 19.4 (+45.7) | 19.4 (+45.7) |
| 90 | 7.9 | 9.4 (+18.7) | 10.7 (+34.4) | 11.5 (+44.3) | 11.7 (+47.3) |
| 100 | 0.5 | 0.8 (+60.9) | 0.7 (+36.9) | 1.2(+143.5) | 1.4(+177.0) |
| average | 31.6 | 34.1 (+7.8) | 37.0 (+17.0) | 38.0 (+20.5) | 38.9 (+23.3) |

An assumption made by local context analysis is that the top ranked documents contain a reasonable number of relevant documents. We now discuss how dependent the performance of local context analysis is on the assumption. Since we use top ranked passages instead of documents, our discussion is based on how many "relevant" passages are in the top ranked passages. A relevant passage is a passage from a relevant document. Table 7.5 shows the number of relevant passages in the top 100 passages for those TREC4 queries whose performance change using local context analysis is greater than 1.0%. The performance change is the absolute value of the difference between the 11 point precisions of the original query and the expanded query. The top 100 passages are used for each query by local context analysis. Though the data show no hard threshold on the number of relevant passages below or above which local context analysis will hurt or improve retrieval performance, the queries improved tend to have more relevant passages than those hurt by local context analysis. The improved queries have 29.7 relevant passages per query on average while those hurt only have 8.6 relevant passages per query. These figures

Table 7.3. Retrieval performance on TREC4. 10 documents are used for local feedback (lf-10doc). 100 passages are used for local context analysis (lca-100p).

| | Precision (% change) – 49 queries | | | | |
|---|---|---|---|---|---|
| Recall | base | Phrasefinder | corpus-query | lf-10doc | lca-100p |
| 0 | 71.0 | 68.6 (−3.3) | 70.4 (−0.8) | 68.4 (−3.6) | 73.2 (+3.2) |
| 10 | 49.3 | 48.6 (−1.6) | 54.3 (+10.0) | 52.8 (+7.0) | 57.1 (+15.7) |
| 20 | 40.4 | 40.0 (−1.0) | 45.0 (+11.6) | 43.2 (+7.0) | 46.8 (+16.0) |
| 30 | 33.3 | 33.9 (+1.8) | 37.7 (+13.4) | 36.0 (+8.0) | 39.9 (+19.8) |
| 40 | 27.3 | 28.0 (+2.5) | 32.6 (+19.4) | 29.8 (+9.2) | 35.3 (+29.1) |
| 50 | 21.6 | 23.9 (+10.3) | 27.4 (+26.7) | 24.5 (+13.2) | 29.9 (+38.4) |
| 60 | 14.8 | 18.8 (+27.1) | 20.8 (+40.7) | 19.7 (+33.4) | 23.6 (+59.8) |
| 70 | 9.5 | 11.8 (+24.7) | 13.6 (+43.3) | 14.8 (+56.9) | 17.9 (+89.1) |
| 80 | 6.2 | 8.1 (+31.0) | 8.2 (+33.8) | 10.8 (+74.7) | 11.8 (+91.0) |
| 90 | 3.1 | 4.2 (+33.6) | 4.2 (+34.2) | 6.4(+104.6) | 5.7 (+80.2) |
| 100 | 0.4 | 0.6 (+24.0) | 0.6 (+36.7) | 0.9 (+93.3) | 0.8 (+88.2) |
| average | 25.2 | 26.0 (+3.4) | 28.6 (+13.7) | 27.9 (+11.0) | 31.1 (+23.5) |

show that local context analysis still depends on the relevance of some top ranked documents (passages) but not as strongly as local feedback.

From the above discussion, it becomes apparent that if we can predict with high accuracy whether the top ranked passages have enough relevant passages, we would be able to do a better job using local context analysis to improve retrieval performance. Rather than expand all queries, we would only expand those which are likely to have enough top ranked relevant passages. One method we have tried is based on the number of matched query terms in the top ranked passages. The basic idea is that if the top ranked passages contain few query terms, they are unlikely to be relevant and we will not expand the query. Unfortunately, this simple method does not work. Data obtained on the TREC collections show that the queries improved and the queries hurt by local context analysis have roughly the same average number of matched query terms (4.0) in a top ranked passage when top 10 passages are considered for each query. Whether more refined methods will work awaits further investigation.

Table 7.4. Retrieval Performances on WEST. 10 documents are used for local feedback (lf-10doc). 100 passages are used for local context analysis (lca-100p). Expansion words are downweighted by 50% for local feedback and local context analysis.

| | Precision (% change) – 34 queries | | | | |
|---|---|---|---|---|---|
| Recall | base | phrasefinder | corpus-query | lf-10doc | lca-100p |
| 0 | 88.0 | 83.9 (−4.7) | 90.1 (+2.4) | 81.9 (−7.0) | 92.1 (+4.7) |
| 10 | 80.0 | 74.5 (−6.9) | 79.8 (−0.3) | 76.9 (−4.0) | 84.3 (+5.4) |
| 20 | 77.5 | 67.2 (−13.3) | 73.9 (−4.5) | 71.4 (−7.8) | 78.5 (+1.3) |
| 30 | 74.1 | 64.3 (−13.2) | 68.4 (−7.7) | 68.2 (−7.9) | 73.9 (−0.1) |
| 40 | 62.9 | 54.6 (−13.2) | 62.5 (−0.7) | 60.8 (−3.3) | 61.8 (−1.7) |
| 50 | 57.5 | 49.5 (−14.0) | 56.6 (−1.6) | 56.8 (−1.2) | 56.8 (−1.2) |
| 60 | 49.7 | 44.6 (−10.1) | 50.7 (+2.1) | 50.1 (+0.8) | 50.7 (+2.2) |
| 70 | 41.5 | 37.4 (−9.9) | 43.8 (+5.5) | 42.1 (+1.3) | 44.2 (+6.4) |
| 80 | 32.7 | 30.0 (−8.2) | 37.2 (+13.7) | 33.1 (+1.1) | 36.4 (+11.2) |
| 90 | 19.3 | 18.4 (−4.6) | 22.0 (+14.2) | 21.8 (+13.0) | 22.6 (+17.1) |
| 100 | 8.6 | 8.7 (+0.3) | 9.1 (+5.7) | 9.3 (+7.8) | 10.0 (+15.3) |
| average | 53.8 | 48.5 (−9.9) | 54.0 (+0.4) | 52.0 (−3.3) | 55.6 (+3.3) |

### 7.3.2 Local Context Analysis vs Local Feedback

Table 7.6 shows the retrieval performance of local feedback using top ranked documents instead of the top ranked passages. The results labeled WEST2 are obtained by downweighting the expansion terms and phrases by 50%. Local feedback does very well on TREC3 and TREC4. The best run on TREC3 produces a 20.5% improvement over the baseline, close to the 24.4% improvement of the best run of local context analysis. On TREC4, the best run produces a 14.0% improvement over the baseline, a substantial increase, but lower than the 23.5% improvement of the best run of local context analysis. The t-test indicates the improvements are significant (p_value=0.0002 on TREC3 and p_value=0.0046 on TREC4). On WEST, local feedback using top ranked documents does not work at all. Without downweighting the expansion terms and phrases, it results in a significant performance loss over all runs. Downweighting the expansion concepts by 50% only reduces the amount of loss.

Table 7.5. Numbers of relevant passages in the top 100 passages for TREC4 queries. 100 passages are used for each query by local context analysis (lca).

| queries improved | | | queries hurt | | |
|---|---|---|---|---|---|
| baseline | lca | relevant passages | baseline | lca | relevant passages |
| 3.2 | 4.3 | 3 | 1.8 | 0.3 | 1 |
| 1.7 | 2.8 | 4 | 4.1 | 1.4 | 2 |
| 2.7 | 11.1 | 4 | 24.9 | 19.5 | 5 |
| 2.7 | 5.6 | 5 | 20.2 | 19.0 | 12 |
| 56.5 | 63.0 | 5 | 23.2 | 21.9 | 15 |
| 1.5 | 3.3 | 6 | 21.6 | 17.8 | 17 |
| 5.7 | 9.2 | 6 | | | |
| 5.2 | 9.0 | 8 | | | |
| 5.3 | 17.6 | 14 | | | |
| 20.0 | 37.3 | 15 | | | |
| 28.1 | 46.5 | 16 | | | |
| 17.9 | 27.3 | 17 | | | |
| 20.6 | 25.0 | 19 | | | |
| 28.7 | 35.2 | 19 | | | |
| 42.5 | 53.2 | 22 | | | |
| 33.3 | 39.8 | 23 | | | |
| 49.5 | 51.0 | 24 | | | |
| 54.2 | 59.1 | 24 | | | |
| 25.9 | 36.5 | 25 | | | |
| 32.4 | 47.4 | 27 | | | |
| 23.5 | 30.1 | 29 | | | |
| 18.8 | 43.3 | 31 | | | |
| 16.8 | 31.1 | 33 | | | |
| 18.1 | 29.0 | 37 | | | |
| 29.8 | 39.6 | 37 | | | |
| 31.9 | 43.5 | 42 | | | |
| 33.7 | 35.6 | 47 | | | |
| 34.7 | 35.8 | 48 | | | |
| 44.6 | 65.9 | 49 | | | |
| 60.4 | 67.3 | 50 | | | |
| 27.2 | 29.5 | 55 | | | |
| 40.8 | 48.4 | 56 | | | |
| 50.6 | 59.3 | 56 | | | |
| 42.3 | 56.1 | 63 | | | |
| 50.2 | 57.8 | 64 | | | |
| 53.0 | 63.4 | 87 | | | |

Table 7.6. Performance of document level local feedback using 11 point average precision. Expansion terms and phrases are downweighted by 50% for WEST2

| number of documents used | TREC4 | TREC3 | WEST | WEST2 |
|---|---|---|---|---|
| baseline | 25.2 | 31.6 | 53.8 | 53.8 |
| 5 | 28.7(+14.0) | 36.6(+16.0) | 49.6(-7.8) | 52.6(-2.2) |
| 10 | 27.9(+11.0) | 38.0(+20.5) | 49.8(-7.5) | 52.0(-3.3) |
| 20 | 26.9(+6.8) | 37.6(+19.1) | 46.2(-14.2) | 48.7(-9.5) |
| 30 | 27.2(+8.2) | 37.7(+19.4) | 44.1(-18.0) | 47.5(-11.6) |
| 50 | 26.7(+6.2) | 37.7(+19.3) | 40.0(-25.6) | 44.5(-17.2) |
| 100 | 26.1(+3.5) | 36.6(+15.8) | 35.1(-34.7) | 40.0(-25.7) |

Local context analysis is better than local feedback on all three collections. The t-test indicates the best run of local context analysis is significantly better than the best run of local feedback on TREC4 and WEST (p_value=0.036 and 0.0001). On TREC3, though local context analysis is slightly better, the t-test does not indicate a significant difference. Tables 7.2, 7.3 and 7.4 compare the performances of local feedback and local context analysis at different recall points on the three collections. The top 10 documents are used for local feedback and the top 100 passages are used for local context analysis in these tables. In Table 7.4 for WEST, the expansion concepts are downweighted by 50% for both local feedback and local context analysis.

The performance of local feedback depends heavily on the number of documents used. On all three collections, the performance drops quickly when more documents are used after the optimal number. The sensitivity to the number of documents used seems to be related to the number of relevant documents in a collection. On TREC3, which has the largest number of relevant documents per query, the performance drop is the slowest. Increasing the number of documents from 10 to 100 results in a small (3.7%) performance loss. On TREC4, which has fewer relevant documents per query, the performance drops more quickly. The performance drop from 5 to 100 documents is significant (9.1%). On WEST, which has the fewest number of relevant documents per query, the performance drop from 10 to 100 documents is huge (29%).

In comparison, local context analysis is less sensitive to the number of passages used. The difference is more clearly shown in Figure 7.1.

Though both methods significantly improve retrieval performance on TREC4, a query by query comparison of the best runs shows that local context analysis is more consistent than local feedback. Of 49 TREC4 queries, local feedback hurts 21 and improves 28, while local context analysis hurts 11 and improves 38. Of the queries hurt by local feedback, 5 queries have a more than 5% percent loss in average precision. In the worst case, the average precision of one query is reduced from 24.8% to 4.3%. Of those hurt by local context analysis, only one has a more that 5% percent loss in average precision. Local feedback also tends to hurt queries with poor performance. Of 9 queries with baseline average precision less than 5%, local feedback hurts 8 and improves 1. In contrast, local context analysis hurts 4 and improves 5. Its tendency to hurt "bad" queries and queries with few relevant documents (such as the WEST queries) shows that local feedback is not robust and very sensitive to the number of relevant documents in the top ranked documents whereas local context analysis does not suffer from these limitations.

When top ranked passages (text windows of 300 words) rather than top ranked documents are used for local feedback, we get slightly different results, as shown in Table 7.7. The performance of using top ranked passages is similar to that of using top ranked documents on TREC4. On TREC3, the performance of using top ranked passages is slightly worse. On WEST, using 10 passages even produces a 4.1% improvement over the baseline. This is understandable because WEST documents are very long. Using passages eliminates the terms from the unrelated passages.

From Figure 7.1 we can see another interesting difference between local feedback and local context analysis in the way the number of passages or documents used affects their performance. The performance of local feedback and local context analysis peaks at quite different numbers of passages. For local feedback, the performance peaks when relatively few passages (around 10) are used. For local context analysis, the

Table 7.7. Performance of passage level local feedback using 11 point average precision

| number of passages used | TREC4 | TREC3 | WEST |
|---|---|---|---|
| baseline | 25.2 | 31.6 | 53.8 |
| 5 | 28.6(+13.6) | 34.7(+9.8) | 55.1(+2.4) |
| 10 | 28.7(+13.9) | 35.5(+12.4) | 56.0(+4.1) |
| 20 | 28.4(+12.7) | 36.4(+15.2) | 52.9(-1.7) |
| 30 | 27.6(+9.6) | 37.0(+17.3) | 53.9(+0.1) |
| 50 | 26.9(+6.8) | 36.6(+15.8) | 52.2(-3.0) |
| 100 | 25.4(+0.8) | 35.8(+13.5) | 49.5(-8.1) |

peak performance occurs when significantly more passages (100) are used. From 10 to 100, using more passages hurts the performance of local feedback but improves that of local context analysis. The difference can be explained by the different assumptions concerning the top ranked documents (passages) made by the two techniques. As we discussed in Chapter 4, local feedback is based on the assumption that a large percentage of the top ranked documents are relevant. Since this assumption is more likely to be violated with more top ranked documents, using more documents (passages) hurts the performance of local feedback. Local context analysis is based on the assumption that a reasonably large number of top ranked documents are relevant. Since this assumption is more likely to be true with more documents (passages), using more passages improves the performance of local context analysis. The difference in the assumptions also explains why local context analysis is less likely to hurt queries with poor initial retrieval performance. Since local context analysis does not assume a high percentage of relevant documents in the top ranked documents, it is more effective in separating the concepts in the relevant documents from the concepts in the non-relevant documents when the percentage of relevant documents is low.

It is interesting to note that although both local context analysis and local feedback find concepts from top ranked documents (passages), the overlap of the concepts chosen by them is very small. On TREC4, the average number of unique terms in the expansion concepts per query is 58 for local feedback and 78 for local

context analysis (using the default parameter setting). The number of overlapping terms is only 17.6 terms per query. This means local context analysis and local feedback are two quite different query expansion techniques. Some queries expanded quite differently are improved by both methods. For example, the overlap for the query "What are the different techniques used to create self-induced hypnosis" is 19 terms, yet both methods improve the query significantly. The expansion concepts added by the two methods to the query are shown in Figures 7.2 and 7.3. This small overlap is understandable because the concepts chosen by local feedback and local context analysis have different properties. In general, concepts selected by local feedback have a high frequency in the top ranked documents while those selected by local context analysis co-occur with all query words in the top ranked documents.

| | | |
|---|---|---|
| hypnot | hypnotiz | 19960500 |
| psychosomat | psychiatr | immun |
| mesmer | franz | suscept |
| austrian | dyck | psychiatrist |
| shesaid | tranc | professor |
| hallucin | 18th | centur |
| hilgard | 11th | unaccept |
| 19820902 | syndrom | exper |
| physician | told | patient |
| hemophiliac | strang | cortic |
| ol | defic | muncie |
| spiegel | diseas | imagin |
| suggest | dyke | feburar |
| immunoglobulin | reseach | fresco |
| person | numb | katie |
| psorias | treatment | medicin |
| 17150000 | ms | franz-mesmer |
| austrian-physician | psychosomat-medicin | |
| hypnot-state | fight-immun | intern-congress |
| late-18th | diseas-fight | hypnotiz-peopl |
| ms-ol | | |

Figure 7.3. Local feedback terms and phrases for TREC4 query 214. Terms are stemmed.

### 7.3.3 Local Context Analysis vs Global Methods

In this section we compare local context analysis with two global methods: the corpus-query based approach and Phrasefinder. The retrieval performances of the three techniques are shown in Tables 7.2, 7.3 and 7.4. On all three collections, local context analysis is better than the global methods.

On TREC3, the improvements over the baseline by Phrasefinder, the corpus-query based approach and local context analysis are 7.8%, 17.0%, and 23.3% respectively. The t-test indicates local context analysis is significantly better than both Phrasefinder (p_value=0.0003) and the corpus-query based approach (p_value=0.024).

On TREC4, the improvements over the baseline by Phrasefinder, the corpus-query based approach and local context analysis are 3.4%, 13.7%, and 23.5% respectively. Again, the t-test indicates local context analysis is significantly better than both Phrasefinder (p_value=0.00001) and the corpus-query based approach (p_value=0.0008).

On WEST, Phrasefinder is 9.9% worse than the baseline, the corpus-query based approach is 0.4% better than the baseline and local context analysis is 3.3% better than the baseline. The t-test indicates local context analysis is significantly better than Phrasefinder (p_value=0.000006) but not than the corpus-query based approach.

We have discussed the impact of the number of passages used on the performance of local context analysis. We now revisit this issue. If all the passages in a collection are used, local context analysis is essentially the corpus-query based approach. In other words, the corpus-query based approach is a special case (an extreme one) of local context analysis. The performance curve of local context analysis in 7.1 implies that this special case of local context analysis would be worse than using a few hundred passages. As we conjectured in Chapter 6, the harm caused by the noise in the global data outweighs its benefit. As the result, global methods are not as effective as local context analysis. To illustrate this point, we consider a TREC4 query "Is there data available to suggest that capital punishment is a deterrent to crime?".

Phrasefinder added some non-relevant concepts such as "Iraqi leader", "world order" "Iraqi army" and "shields" to the query because they co-occur with query words "available", "suggest", "capital", "crime" and "deterrent" in the global collection. This is not a problem for local context analysis because these concepts are unlikely to occur in the top ranked passages.

### 7.3.4 Parameter Variation

We now experiment with different parameter values and see how the performance of local context analysis is affected. The parameters we consider are the passage size, the $\delta$ value (in function $f(c, Q)$) and the number of concepts added to each query and the concept definition.

#### 7.3.4.1 Passage Size and Number of Passages Used

In Table 7.8, we list the retrieval performance of local context analysis on WEST using different passage sizes. We experimented with passage sizes 100, 200, 300, 400 and 500 words. For each passage size, we used the top 10, 20, 30, 40, 50, 100, 200 and 300 passages for query expansion. As we can see from the table, local context analysis produces improvements over the baseline for a wide variety of passage sizes and numbers of passages used. In general, with a larger window size, optimal performance occurs with a smaller number of passages. Performance seems to be independent of the passage size.

Though the experiments on WEST show that the performance is independent of the passage size, we think neither too large a passage size nor too small a passage size is desirable. If the passage size is too small, there will be fewer matched words between a query and the passages. A passage matching only the non-content words in the query may be ranked highly for the query. Consequently, the quality of the retrieved passages is affected. Since the WEST queries are short, this is not a serious problem. But this may be a serious problem for long queries such as the TREC3

Table 7.8. Effect of passage size and number of passages used on retrieval performance of local context analysis on WEST

| | Number of passages used | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| passage size | 10 | 20 | 30 | 40 | 50 | 100 | 200 | 300 |
| 100 | 55.3 | 55.9 | 55.8 | 56.3 | 56.6 | 55.4 | 54.8 | 54.7 |
| | +2.8 | +3.8 | +3.7 | +4.7 | +5.2 | +3.0 | +1.8 | +1.6 |
| 200 | 55.3 | 56.7 | 57.0 | 57.3 | 57.0 | 56.1 | 55.1 | 54.6 |
| | +2.8 | +5.5 | +5.9 | +6.5 | +6.0 | +4.4 | +2.5 | +1.5 |
| 300 | 55.9 | 56.5 | 55.6 | 55.7 | 55.8 | 55.6 | 54.6 | 54.4 |
| | +3.8 | +5.0 | +3.4 | +3.6 | +3.7 | +3.3 | +1.6 | +1.2 |
| 400 | 56.5 | 56.7 | 55.7 | 55.5 | 55.8 | 55.9 | 54.5 | 53.7 |
| | +4.9 | +5.4 | +3.6 | +3.1 | +3.8 | +4.0 | +1.3 | -0.1 |
| 500 | 56.0 | 56.4 | 56.3 | 56.7 | 56.2 | 55.9 | 55.0 | 54.2 |
| | +4.1 | +4.8 | +4.7 | +5.5 | +4.5 | +4.0 | +2.2 | +0.7 |

queries. If the passages are too long, the top ranked passages may contain extraneous text and the performance may be affected. The adverse effect of very small and very large passage sizes may be observed if we repeat the experiments on the TREC collections. Unfortunately, we have not done that because we do not have enough disk space to carry out more experiments on the large TREC collections. To avoid the potential problems with small and large passage sizes, we choose 300 words as the default passage size.

We think, however, that the best method is to segment long documents into passages such that each passage is about a topic. Some techniques that automatically segment long documents or text streams by topics are presented in [25] and [43]. We plan to investigate the application of these techniques for local context analysis in future work.

#### 7.3.4.2 $\delta$ Value

Table 7.9 shows the effect of the $\delta$ value on the performance of local context analysis on TREC4. We can see that the average precision is relatively insensitive to the $\delta$ value. But precision at individual recall points is affected. In general, a small $\delta$ value is good for precision and a large $\delta$ is good for recall.

Table 7.9. Effect of $\delta$ values on performance of local context analysis on TREC4

| | Precision (% change) – 49 queries | | | | | |
|---|---|---|---|---|---|---|
| recall | 0.001 | 0.01 | 0.05 | 0.1 | 0.2 | 0.3 |
| 0 | 77.8 | 75.7 (−2.7) | 73.4 (−5.6) | 73.2 (−5.9) | 72.9 (−6.3) | 73.1 (−6.0) |
| 10 | 55.1 | 54.9 (−0.2) | 57.0 (+3.6) | 57.1 (+3.7) | 56.8 (+3.2) | 56.6 (+2.7) |
| 20 | 46.7 | 47.4 (+1.6) | 47.7 (+2.2) | 46.8 (+0.3) | 46.6 (−0.1) | 46.5 (−0.5) |
| 30 | 40.0 | 40.4 (+1.0) | 40.0 (+0.1) | 39.9 (−0.2) | 39.6 (−0.9) | 39.4 (−1.5) |
| 40 | 34.8 | 35.4 (+1.6) | 35.4 (+1.7) | 35.3 (+1.4) | 34.8 (+0.1) | 34.6 (−0.7) |
| 50 | 29.4 | 30.1 (+2.4) | 30.2 (+2.6) | 29.9 (+1.7) | 30.1 (+2.4) | 30.1 (+2.2) |
| 60 | 22.5 | 23.1 (+2.6) | 23.7 (+5.3) | 23.6 (+5.2) | 24.1 (+7.3) | 24.1 (+7.1) |
| 70 | 15.9 | 17.0 (+6.6) | 17.7(+11.2) | 17.9(+12.3) | 17.6(+10.7) | 17.6(+10.3) |
| 80 | 11.0 | 11.6 (+6.0) | 11.7 (+7.2) | 11.8 (+7.3) | 11.9 (+8.3) | 11.9 (+8.2) |
| 90 | 5.1 | 5.3 (+4.6) | 5.6(+11.3) | 5.7(+11.9) | 5.9(+15.4) | 5.9(+16.8) |
| 100 | 0.8 | 0.9(+11.2) | 0.8 (+7.5) | 0.8 (+7.4) | 0.9(+11.9) | 0.9(+11.6) |
| avg | 30.8 | 31.1 (+0.8) | 31.2 (+1.3) | 31.1 (+0.9) | 31.0 (+0.7) | 30.9 (+0.4) |

To show the role that $\delta$ plays in the formula $f(c, Q)$, we consider the case that $Q$ has only two words $w_1$ and $w_2$. To further simplify the discussion, we ignore the *idf* values of $w_1$ and $w_2$ in $f(c, Q)$. Under these simplifications, we have

$$\begin{aligned} f(c,Q) &= (\delta + co\_degree(c,w_1))(\delta + co\_degree(c,w_2)) \\ &= co\_degree(c,w_1)co\_degree(c,w_2) + \delta(co\_degree(c,w_1) + co\_degree(c,w_2)) \\ &\quad +\delta^2 \end{aligned}$$

The formula consists of three items: The first item, $co\_degree(c, w_1)co\_degree(c, w_2)$ emphasizes co-occurrence with both query words. The second item, $\delta(co\_degree(c, w_1)+ co\_degree(c, w_2))$, emphasizes co-occurrence with individual query words. The third item, $\delta^2$, has no effect on the ranking of the concepts. The relative weights of the first and second items are controlled by the $\delta$ value. With a small $\delta$, concepts co-occurring with all query words are ranked higher. With a large $\delta$, concepts having significant co-occurrences with individual query words are ranked higher. The experimental results seem to imply that concepts co-occurring with all query words are good for precision and concepts co-occurring with individual query words are good for recall.

#### 7.3.4.3 Number of Concepts to Use

In the above experiments, we add 70 concepts to each query. Adding so many concepts obviously slows the retrieval process. We now examine how the performance of local context analysis is affected if we use fewer concepts for query expansion. We add 30 concepts with equal weights to each query. A property of the INQUERY weighted SUM operator (#WSUM) is that the fewer items that are inside the operator, the larger the belief value it returns. To offset the larger belief value produced by the auxiliary query because of the decrease in the number of concepts used, we set the weight of the auxiliary query to 1.0 rather than the default 2.0. The retrieval performance is shown in Table 7.10. The performance using 30 concepts is close to that using 70 concepts. The performance at low recall points is better and the performance at high recall points is worse than using 70 concepts. This means that the concepts ranked below 30 are somewhat useful in retrieving more relevant documents but can bring non-relevant documents to top of the ranked output.

Table 7.10. Using 70 concepts of local context analysis vs using 30 concepts on TREC4

| | Precision (% change) – 49 queries | | |
|---|---|---|---|
| Recall | baseline | lca-70-cpt | lca-30-cpt |
| 0 | 71.0 | 73.2 (+3.2) | 73.8 (+4.1) |
| 10 | 49.3 | 57.1 (+15.7) | 57.3 (+16.1) |
| 20 | 40.4 | 46.8 (+16.0) | 48.0 (+19.1) |
| 30 | 33.3 | 39.9 (+19.8) | 40.1 (+20.5) |
| 40 | 27.3 | 35.3 (+29.1) | 35.0 (+28.0) |
| 50 | 21.6 | 29.9 (+38.4) | 29.6 (+37.0) |
| 60 | 14.8 | 23.6 (+59.8) | 23.0 (+55.2) |
| 70 | 9.5 | 17.9 (+89.1) | 16.4 (+73.5) |
| 80 | 6.2 | 11.8 (+91.0) | 10.7 (+74.1) |
| 90 | 3.1 | 5.7 (+80.2) | 5.5 (+74.9) |
| 100 | 0.4 | 0.8 (+88.2) | 0.7 (+67.1) |
| average | 25.2 | 31.1 (+23.5) | 30.9 (+22.9) |

#### 7.3.4.4 One Term and Two Term Concepts

In the above experiments concepts are defined as nouns and noun phrases in the documents. We now experiment with an alternative definition of concepts. We define concepts as single terms and pairs of adjacent terms (excluding stop words) in the documents. The experiments are carried out on TREC4. We can imagine that there will be a huge number of concepts in the collection with this new concept definition. In order to reduce the number of concepts, we stem the concepts using the aggressive Porter stemmer [46].

Figure 7.4 compares the retrieval performance of the old and the new concept definitions. In general, the old definition (nouns and noun phrases) is better than the new definition (terms and pairs of terms). The difference is, however, very small except when 200 or more passages are used. The peak performance of the two definitions is very close, as shown in Table 7.11.

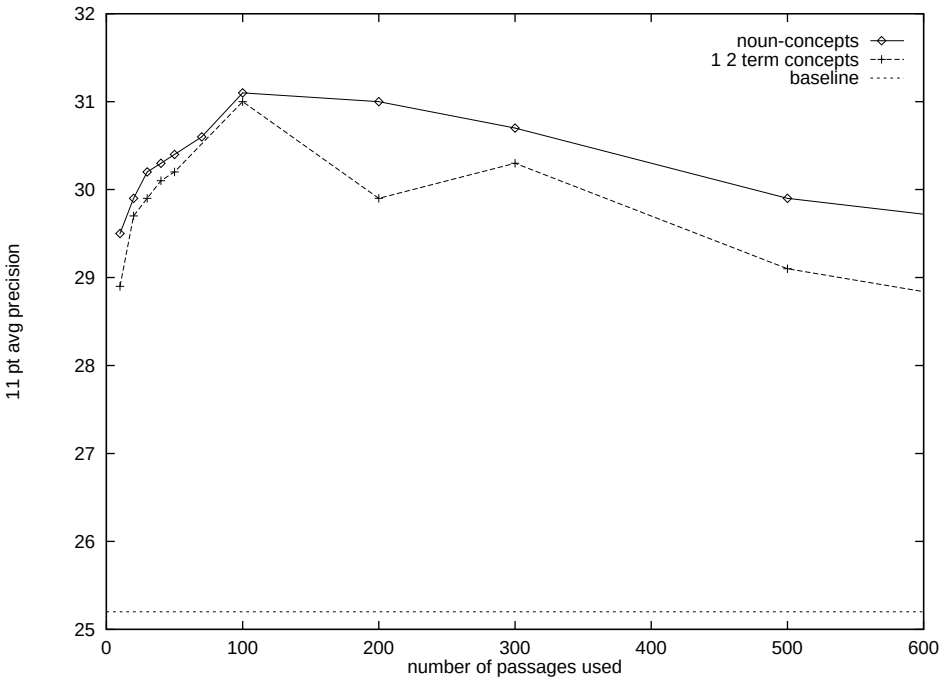


Figure 7.4. Performance of local context analysis using terms and pairs of terms vs using nouns and noun phrases on TREC4

Table 7.11. Nouns and noun phrases vs terms and pairs of terms as concepts of local context analysis on TREC4.

| | Precision (% change) – 49 queries | | |
|---|---|---|---|
| Recall | base | noun-concepts | 1-2term-concepts |
| 0 | 71.0 | 73.2 (+3.2) | 75.8 (+6.8) |
| 10 | 49.3 | 57.1 (+15.7) | 56.9 (+15.4) |
| 20 | 40.4 | 46.8 (+16.0) | 46.9 (+16.1) |
| 30 | 33.3 | 39.9 (+19.8) | 39.3 (+18.2) |
| 40 | 27.3 | 35.3 (+29.1) | 33.5 (+22.7) |
| 50 | 21.6 | 29.9 (+38.4) | 28.2 (+30.2) |
| 60 | 14.8 | 23.6 (+59.8) | 22.2 (+50.3) |
| 70 | 9.5 | 17.9 (+89.1) | 18.0 (+90.0) |
| 80 | 6.2 | 11.8 (+91.0) | 12.9 (+109.9) |
| 90 | 3.1 | 5.7 (+80.2) | 6.9 (+118.3) |
| 100 | 0.4 | 0.8 (+88.2) | 0.8 (+81.4) |
| average | 25.2 | 31.1 (+23.5) | 31.0 (+23.3) |

The main advantage of the new concept definition is that it does not require part of speech recognition and thus is more efficient. The other advantage is that we can easily apply it to languages for which we do not have the part of speech recognition software. The main disadvantage is that it produces less readable concepts. In contrast, nouns phrases are generally more readable and more desirable for interactive applications. The other disadvantage is that it results in significantly more concepts than the old definition and therefore consumes more space. The TREC4 collection contains 10 million unique terms and pairs of terms. In comparison, it contains only 4.9 million unique nouns and noun phrases.

### 7.3.5 Results on a Small Collection

The collections we have used so far are generally very large. Though most real world collections are large, some can be quite small. It would be preferable if local context analysis also works on small collections.

Table 7.12 shows the retrieval performance of local context analysis on TIME, a very small collection consisting of 423 documents. When 20 passages are used,

local context analysis produces a 4% improvement over the baseline queries. The t-test indicates that the improvement is statistically significant (p_value=0.045). The improvement becomes smaller when more passages are used. This is not surprising because TIME is such a small collection. The performance of using 100 passages is almost identical to the baseline. The results show that local context analysis works on collections of different sizes, from a few hundred to nearly one million documents.

Table 7.12. Retrieval performance of local context analysis on TIME. 20, 50 and 100 passages are used for local context analysis (lca-20p, lca-50p and lca-100p).

| | Precision (% change) – 83 queries | | | |
|---|---|---|---|---|
| Recall | baseline | lca-20p | lca-50p | lca-100p |
| 0 | 80.8 | 82.7 (+2.3) | 82.1 (+1.5) | 80.2 (−0.8) |
| 10 | 80.8 | 82.6 (+2.2) | 82.0 (+1.4) | 79.7 (−1.4) |
| 20 | 79.6 | 81.8 (+2.8) | 81.4 (+2.2) | 79.5 (−0.1) |
| 30 | 77.0 | 80.3 (+4.2) | 78.9 (+2.4) | 76.6 (−0.6) |
| 40 | 74.9 | 79.6 (+6.3) | 77.0 (+2.8) | 75.1 (+0.3) |
| 50 | 73.5 | 79.4 (+8.1) | 76.4 (+4.0) | 74.4 (+1.3) |
| 60 | 67.4 | 69.9 (+3.8) | 68.2 (+1.3) | 66.6 (−1.1) |
| 70 | 64.9 | 66.5 (+2.6) | 65.4 (+0.8) | 64.8 (−0.1) |
| 80 | 63.3 | 65.0 (+2.7) | 63.6 (+0.4) | 63.1 (−0.3) |
| 90 | 58.8 | 61.4 (+4.4) | 60.6 (+3.0) | 60.0 (+2.1) |
| 100 | 58.0 | 60.9 (+5.1) | 59.8 (+3.1) | 59.3 (+2.3) |
| average | 70.8 | 73.7 (+4.0) | 72.3 (+2.1) | 70.9 (+0.1) |

## 7.4 Multilingual Experiments

To see whether local context analysis works on non-English collections, experiments were carried out on a Chinese collection and a Spanish collection. We should mention that the experiments in this section were mostly carried out by other researchers at CIIR of UMass for the TREC5 conference.[2] As a result, some parameter values used in this section are different from the ones used in the previous section.

[2]Thanks to Lisa Ballesteros, who did all the Spanish experiments, and John Broglio and Hongmin Shu, who did the major part of the Chinese experiments.

The Chinese experiments were carried out on the TREC5-CHINESE collection. The query words in the Chinese queries are recognized using *Useg* [44], a Chinese segmenter based on the hidden Markov model. We define concepts as words in the documents recognized by the segmenter. Documents are broken into passages containing no more than 1500 Chinese characters. To expand a query, the top 30 concepts from the top 10 passages for the query are used. Concept $i$ is assigned the weight

$$w_i = 1.0 - (i - 1)/60$$

The weight of the auxiliary query is set to 1.0.

Table 7.13 shows the retrieval performance of local context analysis on the Chinese collection. The retrieval performance is 14.0% better than that of the unexpanded queries. Precision at all recall points is improved significantly. The t-test indicates the improvement is statistically significant (p_value=0.01).

Table 7.13. Performance of local context analysis on TREC5-CHINESE.

| | Precision (% change) – 19 queries | | |
|---|---|---|---|
| Recall | baseline | lca | |
| 0 | 69.1 | 74.9 | (+8.4) |
| 10 | 56.8 | 60.7 | (+6.8) |
| 20 | 49.2 | 56.2 | (+14.1) |
| 30 | 43.1 | 48.5 | (+12.4) |
| 40 | 37.2 | 44.2 | (+18.9) |
| 50 | 33.2 | 36.3 | (+9.3) |
| 60 | 26.9 | 31.2 | (+16.1) |
| 70 | 20.1 | 26.3 | (+31.2) |
| 80 | 16.8 | 21.9 | (+30.1) |
| 90 | 12.5 | 14.5 | (+16.2) |
| 100 | 3.7 | 5.7 | (+53.4) |
| average | 33.5 | 38.2 | (+14.0) |

The Spanish experiments were carried out on the TREC5-SPANISH collection. As in the English experiments, query words are terms in the queries and concepts

are nouns and noun phrases in the documents. The passage size is 200 words. The top 31 concepts from the top 20 passages are added to each query. The top concept is given the weight 1.0 with all subsequent concepts downweighted by 1/100 for each position further down the rank. The weight of the auxiliary query is 1.0. Table 7.14 shows the retrieval performance of local context analysis on the Spanish collection. Local context analysis produces a 13% improvement over the unexpanded queries. The t-test indicates the improvement is statistically significant (p_value=0.005).

Table 7.14. Performance of local context analysis on TREC5-SPANISH.

| | Precision (% change) – 25 queries | | |
|---|---|---|---|
| Recall | baseline | lca | |
| 0 | 85.5 | 79.6 | (−7.0) |
| 10 | 72.0 | 74.9 | (+4.2) |
| 20 | 55.2 | 66.8 | (+21.1) |
| 30 | 49.8 | 60.1 | (+20.8) |
| 40 | 45.4 | 51.2 | (+12.9) |
| 50 | 39.8 | 47.1 | (+18.3) |
| 60 | 33.5 | 40.5 | (+20.8) |
| 70 | 27.9 | 34.9 | (+24.7) |
| 80 | 22.0 | 28.2 | (+28.1) |
| 90 | 16.6 | 21.3 | (+27.9) |
| 100 | 1.8 | 3.5 | (+86.9) |
| average | 40.9 | 46.2 | (+13.0) |

## 7.5 Efficiency Issues

In this section we discuss the computational costs of local context analysis. We will first report the costs under the current implementation. Then we will discuss techniques that can be used in the future to reduce these costs.

Specific figures used in the discussion are based on the TREC4 collection, which has 2.0 Gigabytes of data. Experiments are carried out on a DEC Alpha workstation. The concepts used are nouns and noun phrases and the passage size is 300 words.

### 7.5.1 Current Implementation

A local context analysis database consists of the following data structures:

- The concept dictionary, which stores for each concept in the collection an integer coded identifier and its frequency in the collection (number of passages containing the concept).

  The TREC4 concept dictionary contains 4.9 million concepts and uses 167 MB of space.

- The term dictionary, which is analogous to the concept dictionary except it is for terms in the collection.

  The TREC4 term dictionary contains 1.05 million terms and uses 43 MB of space.

- The concept file, which sequentially stores for each passage the concepts that occur in the passage and the numbers of occurrences. To save space and speed up query expansion, concepts in the file are integer coded.

  The TREC4 concept file contains 1.08 million passages and uses 251 MB of space.

- The term file, which is analogous to the concept file except it is for terms. To save space, the occurrences of a term in the collection are represented as an inverted list.

  The TREC4 term file uses 213 MB of space.

Besides the above data structures, we need to build an INQUERY database which indexes the passages in the collection. On TREC4, the INQUERY database uses 850 MB of space.

The time to build the local context analysis database for TREC4 is about 4 hours of wall clock time. Most of the time (3 hours) is spent on parsing the collection and

part of speech tagging. After the collection is parsed and concepts are recognized, it takes only 1 hour to build the local context analysis database. Building the INQUERY database for the passages takes about 6 hours of wall clock time.

The total building time is about 10 hours on TREC4. The total disk usage to store the local context analysis database and the INQUERY database is about 1.5 GB. The peak disk usage is 4.0 GB, most of which is used to store temporary files. The peak memory usage is about 200 MB, most of which is used by a hash table to store the concepts. Considering the size of the collection, the costs are reasonable.

Query expansion for a query consists of two steps. In the first step, INQUERY retrieves passage IDs of the top ranked passages for the query. A passage ID is the sequential number of a passage in the collection. This step takes about 10 seconds per query. In the second step, concepts in the top ranked passages are ranked and the top ranked concepts are output for query expansion. This steps takes about 2 seconds per query when the top 100 passages are used. The total time to expand a query is therefore 12 seconds. The memory usage for query expansion is about 240 MB, most of which is used by the concept dictionary and the term dictionary.

### 7.5.2 Future Implementation

For each collection, the INQUERY retrieval system needs to build a database, which we call the *document database*, that indexes the documents in the collection. The current implementation of local context analysis builds a second INQUERY database, which we call the *passage database*, that indexes the passages in the collection. The passage database takes a lot of space and time to build. The sole purpose of the passage database is to retrieve the top ranked passages. If we can retrieve the top ranked passages from the document database, we can significantly cut the building costs of local context analysis: on TREC4, the total building time will be reduced from 10 hours to 4 hours, the total disk usage will be reduced from 1.5 GB to 0.7 GB and the peak disk usage will be reduced from 4.0 GB to 2.0 GB.

There several possible approaches that we can perform local context analysis without building a passage database. The first approach is to use the passages of the top ranked documents. With this approach, the passages used are no longer the best passages. Some of them may be totally unrelated to the query. The second approach is to rank the documents by their best passage (a feature provided by INQUERY) and use the best passage of each top ranked document for local context analysis. With this approach we can guarantee the quality of the passages used but we will miss some good passages because a document may contain several good passages. The third approach is to modify the retrieval system so that it can retrieve the top ranked passages from the document database. Experiments are required to determine the merits and problems of these approaches.

In the current implementation, we preprocess the raw text in a collection and generate the concept file and the term file so that we can save the cost of parsing the top ranked passages during query expansion. This speeds up query expansion but increases the space overhead. If disk space is in short supply, we can save disk space by parsing the top ranked passages at query expansion time.

As we mentioned before, the memory usage largely depends on the size of the concept dictionary. The TREC4 concept dictionary has 4.9 million concepts. In Figure 7.5, we plot the frequency distribution of the concepts on TREC4. As we can see from the figure, 3.3 million concepts occur only once, 0.67 million concepts occur only twice. These concepts are largely "noise" and are unlikely to have any impact on the overall performance. If we remove the concepts that occur only once or twice from the concept dictionary, we can reduce the concept dictionary size by more than 80%. The reduced concept dictionary only takes about 30 MB. If compression techniques are used, we can further reduce the memory usage. Experiments are needed to decide how many low frequency concepts can be removed and what the impact is on efficiency and effectiveness.

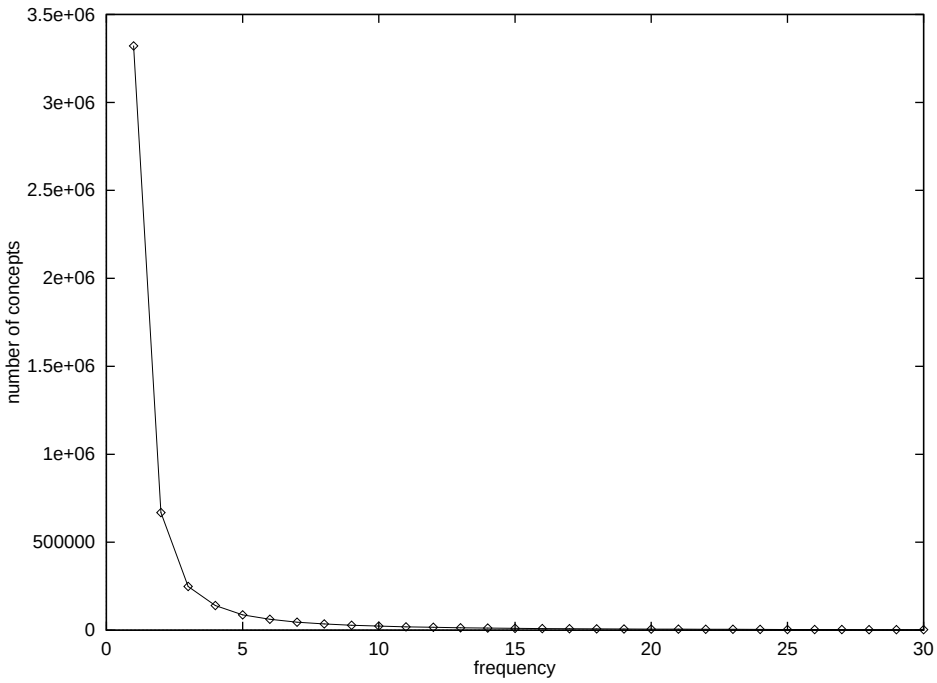


Figure 7.5. Frequency distribution of concepts on TREC4

## 7.6 Summary

The results in this chapter demonstrate that query expansion based on the analysis of the top ranked documents (passages) is more effective than query expansion based on global corpus analysis. The results also show that local context analysis, a query expansion technique which does such an analysis, is more effective than local feedback. Resource usage statistics on a large collection show that local context analysis is reasonably efficient. With more implementation effort, the efficiency can be improved further. We therefore confirm hypotheses 5 and 6.

Recently we have obtained a new set of results using local context analysis on the TREC5 collection. The results are shown in Table 7.15. Although the results are not as good as on other TREC collections, local context analysis is still better than local feedback. When 20 passages are used for each query, local context analysis produces a 9.2% improvement over the baseline. The t-test indicates the improvement is statistically significant (p_value=0.008). In comparison, local feedback produces almost no improvement. Using 100 passages, local context analysis only produces a

Table 7.15. Retrieval performance on TREC5. 10 documents are used for local feedback (lf-10doc). 20 and 100 passages are used for local context analysis (lca-20p and lca-100p).

| | Precision (% change) – 50 queries | | | |
|---|---|---|---|---|
| Recall | baseline | lf-10doc | lca-20p | lca-100p |
| 0 | 64.1 | 48.5 (−24.4) | 62.8 (−2.1) | 53.7 (−16.2) |
| 10 | 37.5 | 36.8 (−1.9) | 37.2 (−0.9) | 34.4 (−8.4) |
| 20 | 29.1 | 31.0 (+6.5) | 33.0 (+13.3) | 30.9 (+6.3) |
| 30 | 24.1 | 26.5 (+9.9) | 27.2 (+13.0) | 26.4 (+9.3) |
| 40 | 21.3 | 22.6 (+6.2) | 23.8 (+12.0) | 23.5 (+10.5) |
| 50 | 17.9 | 19.3 (+7.8) | 21.1 (+17.6) | 20.7 (+15.8) |
| 60 | 12.6 | 15.6 (+24.2) | 17.5 (+39.1) | 17.1 (+36.2) |
| 70 | 10.1 | 12.9 (+27.6) | 13.0 (+28.3) | 12.7 (+25.2) |
| 80 | 7.2 | 9.4 (+31.5) | 8.9 (+23.9) | 8.7 (+21.6) |
| 90 | 4.8 | 6.6 (+36.3) | 5.9 (+22.2) | 6.2 (+28.2) |
| 100 | 2.7 | 2.7 (−2.3) | 2.5 (−7.6) | 2.4 (−13.5) |
| average | 21.0 | 21.1 (+0.2) | 23.0 (+9.2) | 21.5 (+2.3) |

2.3% improvement, which is not statistically significant. This is different from TREC3 and TREC4 where the optimal number of passages is around 100 and using thousands of passages still produces a significant improvement in retrieval effectiveness. Also, unlike the old results, local context analysis is not robust on TREC5. It seriously hurts a number of queries. The above problems may be attributed to the peculiarities of the TREC5 queries. Three of the TREC5 queries have only one or two relevant documents in the whole collection. It is hard for local context analysis to improve them. Some queries are ill-formed from the retrieval point of view in that they use terms that are poor or even negative indicators of relevance. Examples are queries "Evidence of the existence of human life 10,000 years ago" and "Identify social programs for poor people in countries other than the U.S." ("U.S." is negative evidence of relevance). For such queries, local context analysis is likely to choose the wrong words for query expansion by requiring expansion words to co-occur with the "bad" query terms. Though our hypotheses are still valid, the TREC5 results emphasize the need to find

ways to decide when not to expand a query. This is an important topic in the future work.

# Chapter 8

# Conclusion

The goal of the research in this thesis was to address the word mismatch problem through automatic text analysis. We have investigated two text analysis techniques, corpus analysis and local context analysis, applied them in two domains of word mismatch, stemming and general query expansion, and presented the experimental results. In the final chapter, we summarize the research contributions of this thesis and describe future directions.

## 8.1 Contributions

The contributions of this research are:

- We established that automatic corpus analysis based on co-occurrence information can help to identify related and unrelated word variants specific to a corpus. A stemmer enhanced by corpus analysis is adaptable to the corpus being searched, significantly reduces the amount of expansion, makes fewer mistakes and improves retrieval effectiveness slightly.

- We proposed a retargetable stemming algorithm based on simple initial character matching and corpus analysis. The algorithm can be easily ported to a new language and offers retrieval performance competitive with the traditional stemming approach.

- We established that global corpus analysis has limited value for general query expansion. We also established that contrary to widely held belief, query expansion based on global co-occurrence information and query context largely depends on the information in the top ranked documents.

- We established that analysis of the top ranked documents is more effective and more efficient for query expansion than global corpus analysis. Specifically, we proposed a query expansion technique, called local context analysis, which outperforms global query expansion techniques in terms of effectiveness and efficiency.

- We demonstrated that local context analysis is more effective and more robust than local feedback because local context analysis does not make strong assumptions about the relevance of the top ranked documents.

## 8.2 Future Work

Though we have largely achieved the research goal set for the thesis, there are a number of directions in which the work in this thesis can pursued further:

- Overlapping classes for stemming. In this thesis, equivalence classes are non overlapping: a word belongs to exactly one class. For words with multiple primary meanings in a corpus, it may be more desirable to let them belong to multiple classes. It is not totally clear at the moment how to adapt the class refinement algorithms to handle overlapping classes.

  One method to deal with words with multiple primary meanings is to ask the users for information about the membership of the ambiguous query words. Another possible method is to use the top ranked documents to automatically disambiguate the ambiguous query words. But these methods may not be cost effective because stemming is supposed to be a simple and efficient procedure.

- Stemming for languages other than English and Spanish. We have successfully applied initial character matching (ngram) and corpus analysis to stemming for English and Spanish. It would be interesting to apply the ideas to stemming for other languages. Some languages have more complex morphology. For example, in Indonesian, both prefixes and suffixes can be added to a word. For such

languages, initial character matching is not adequate in constructing the initial equivalence classes. One possible solution is to use more complex heuristics (e.g. initial and ending character matching) to construct initial equivalence classes.

- Problem with ngram stemmers. Though we have shown that ngram stemmers perform at least as well as traditional stemmers, they occasionally make mistakes. The mistakes usually happen when two words have the same initial characters and are somewhat related, but are not morphologically related. One example is "market/mark" (articles about market share often mention trade mark infringement). Simple morphological constraints can help address this problem. One constraint might be: two words can not possibly be morphologically related if the longer word does not have a legal suffix ("et" is not a legal suffix).

- Global vs local information. We have shown that the corpus-query based approach as well as similar techniques such as Phrasefinder and Concept Based Query Expansion largely depends on the information in the top ranked documents. This may also be true for other global techniques. We are particularly interested in the dimensionality reduction techniques such as LSI because they have very appealing mathematical properties but are computationally prohibitive. Experiments are needed to confirm the conjecture. If it turns out to be true, it would be compelling evidence that we should abandon the costly practice of using global information for query expansion.

- More principled approaches of combining evidence from the original query words and the expansion concepts in local context analysis. Our current method of combining evidence is ad hoc. We need better ways of deriving the weights of the expansion concepts and deciding the optimal number of passages to use. An ideal solution is to find a method to automatically compute the degree of evidence that a concept supports a query based on the information in the top ranked documents.

One possible method is to assign weights to the expansion concepts based on their suitability values. Currently, the suitability function is only used for concept ranking. The reason we do not use the suitability function in weight assignment right now is that it is not normalized over query length. The longer the query, the smaller the suitability values. It is not totally clear at present how to normalize the suitability function so that the suitability values can be directly used as weights in the expanded query. Some form of regression on the suitability values and query parameters (e.g. query length) may be useful.

- Parameter tuning. Results in Section 7.3.4 show that while varying the $\delta$ value has little impact on average precision, it affects the precision at different recall points. Large $\delta$ values favor recall and small $\delta$ values favor precision. One area worth investigation is to establish the appropriate $\delta$ values for different applications. For applications where precision is more important (e.g. searching the Internet), we can use a small $\delta$ value. For applications where recall is more important (e.g. searching a database of legal texts for a lawyer), we can use a large $\delta$ value.

- When to expand and when not to expand? Though experiments show local context analysis improves average retrieval performance substantially, there are queries for which no expansion is better. It would be useful if we can automatically decide when to expand and when not to expand. Though the simple technique we tried based on the number of matched query terms in the top ranked documents (passages) does not work, more refined techniques may be successful. One possible method, for example, is to cluster the top ranked documents to see whether the original query is highly ambiguous (no clusters or several clusters equally similar to the query). If so, we do not expand the query. One study which used a similar technique for local feedback appeared to achieve some success [36]. We would like to know whether it will work for local context analysis.

- Use of local context analysis in interactive environments. Another solution to the above problem is to interact with the users. When a user types in a query, local context analysis provides the user with a list of candidate words (concepts) that may be added to the query. Based on his/her judgment, the user selects some of the words and discards the others. In this case, local context analysis functions as a dynamic thesaurus. Like a traditional thesaurus, it provides the users with alternative words for their queries. Unlike a traditional thesaurus, it is tailored for each query and avoids the problems with traditional thesauri as discussed in Section 4.1.

- Cluster based search and spreading activation. In this thesis we have compared local context analysis with local feedback. There are other techniques that attempt to utilize the information in the top ranked documents to improve retrieval performance. Cluster based search [13, 26] and spreading activation [14] are two examples. It would be interesting to compare local context analysis with these techniques in terms of retrieval performance. We have done some work on clustering the top ranked documents and using words from the best cluster (the one most similar to the query) for query expansion. Though the preliminary results are not encouraging, more effort in this direction is worthwhile.

- Efficiency issues. Though the query time cost of local context analysis is relatively small, it would be better if we could reduce it further. Remember that retrieval using local context analysis is a two stage process: evaluate the original query (first stage) and evaluate the expanded query (second stage). Since local context analysis needs only the top ranked documents, the first stage can be speeded up by using the techniques proposed by Eric Brown in [4]. Some of the techniques can cut the query evaluation time by more than 50% without affecting the retrieval effectiveness of the top ranked documents. The other place where we can cut the retrieval time is to make use of results of the first stage in the second

stage. Note that the original query is part of the expanded query, therefore the second stage can simply reuse the results for the original query established by the first stage if the two stages are carried out together. Under the current implementation, the two stages are carried out separately for simplicity. In an operational environment, it should be possible to do them together.

- Cross corpora query expansion. We did some cross corpora experiments about corpus based stemming in Chapter 5. Our concern then was on the negative effect of applying equivalence classes obtained on one collection for query expansion on the other.

  There may be also a positive side of cross corpora query expansion. As we discussed in Chapter 1, some collections may not use alternative words in their documents. For such collections, expanding a query on a different collection (presumably a collection known to use alternative words in its documents) may be advantageous. Experiments are needed to determine the merits and problems of cross corpora query expansion.

## Bibliography


[1] Allan, J., Ballesteros, L., Callan, J. P., Croft, W. B., and Lu, Z. Recent experiments with INQUERY. In *Proceedings of the Fourth Text REtrieval Conference (TREC-4)* (1996), D. Harman, Ed., NIST Special Publication.

[2] Attar, R., and Fraenkel, A. S. Local feedback in full-text retrieval systems. *Journal of the Association for Computing Machinery 24*, 3 (July 1977), 397–417.

[3] Broglio, J., Callan, J. P., Croft, W. B., and Nachbar, D. W. Document retrieval and routing using the INQUERY system. In *Proceedings of the Third Text REtrieval Conference (TREC-3)* (1995), D. Harman, Ed., NIST Special Publication 500-225, pp. 22–29.

[4] Brown, E. *Execution Performance Issues in Full-Text Information Retrieval.* PhD thesis, University of Massachusetts Amherst, 1995.

[5] Buckley, C., Singhal, A., Mitra, M., and Salton, G. New retrieval approaches using SMART : TREC 4. In *Proceedings of the TREC 4 Conference* (1996).

[6] Caid, B., Gallant, S., Carleton, J., and Sudbeck, D. HNC Tipster phase I final report. In *Proceedings of Tipster Text Program (Phase I)* (1993), pp. 69–92.

[7] Caid, W., Dumais, S., and Gallant, S. Learned vector-space models for document retrieval. *Information Processing and Management 31*, 3 (1995), 419–429.

[8] Callan, J. P. Passage-level evidence in document retrieval. In *Proceedings of ACM SIGIR International Conference on Research and Development in Information Retrieval* (1994), pp. 302–310.

[9] Callan, J. P., Croft, W., and Harding, S. The INQUERY retrieval system. In *Proceedings of the 3rd International Conference on Database and Expert Systems Applications* (1992), pp. 78–83.

[10] Church, K., and Hanks, P. Word association norms, mutual information, and lexicography. In *Proceedings of the 27th ACL Meeting* (1989), pp. 76–83.

[11] Cormen, T., Leiserson, C., and Rivest, R. *Introduction to Algorithms.* McGraw-Hill, 1990.

[12] Croft, W. B. Using Boolean queries with a clustered file organization. *Journal of the American Society for Information Science 30*, 6 (Nov. 1979), 358–360.

[13] Croft, W. B., and Harper, D. J. Using probabilistic models of document retrieval without relevance information. *Journal of Documentation 35* (1979), 285–295.

[14] Croft, W. B., Lucia, T. J., Cringean, J., and Willett, P. Retrieving documents by plausible inference: An experimental study. *Information Processing and Management 25*, 6 (1989), 599–614.

[15] Crouch, C. J., and Yang, B. Experiments in automatic statistical thesaurus construction. In *SIGIR 92* (1992), pp. 77–88.

[16] Deerwester, S., Dumais, S., Furnas, G., Landauer, T., and Harshman, R. Indexing by Latent Semantic Analysis. *Journal of the American Society for Information Science 41* (1990), 391–407.

[17] Dumais, S. Latent Semantic Indexing (LSI), TREC-3 report. In *Proceedings of the TREC 3 Conference* (1995), D. Harmann, Ed.

[18] Fuhr, N. Probabilistic datalog – a logic for powerful retrieval methods. In *Proceedings of the 18$^{th}$ Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (1995), pp. 282–290.

[19] Furnas, G., Deerwester, S., Dumais, S., Landauer, T., Harshman, R., Streeter, L., and Lochbaum, K. Information retrieval using a singular value decomposition model of latent semantic structure. In *SIGIR 88* (1988), pp. 465–480.

[20] Furnas, G. W., Landauer, T. K., Gomez, L. M., and Dumais, S. T. The vocabulary problem in human-system communication. *Commun. ACM 30*, 11 (Nov. 1987), 964–971.

[21] Grefenstette, G. Use of syntatic context to produce term association lists for text retrieval. In *SIGIR'92* (1992), pp. 89–97.

[22] Haines, D. *Adaptive Query Modification in a Probabilistic Information Retrieval Model.* PhD thesis, Department of Computer Science, University of Massachusetts Amherst, 1996.

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

[24] Harman, D. Overview of the Third Text REtrieval Conference (TREC-3). In *Proceedings of the Third Text REtrieval Conference (TREC-3)* (1995), D. Harman, Ed., NIST Special Publication 500-225, pp. 1–20.

[25] Hearst, M. Mini-paragraph segmentation of expository discourse. In *Proceedings of 32nd Meeting of the Association for Computational Linguistics* (June 1994).

[26] Hearst, M., and Pedersen, J. Reexamining the cluster hypothesis: Scatter/gather on retrieval results. In *Proceedings of the 19$^{th}$ Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (1996), pp. 76–84.

[27] Hull, D. Using statistical testing in the evaluation of retrieval experiments. In *Proceedings of the 13$^{th}$ Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (1993), pp. 329–338.

[28] Hull, D. A. Stemming algorithms: A case study for detailed evaluation. *Journal of the American Society for Information Science 47*, 1 (Jan. 1996), 70–84.

[29] Jing, Y., and Croft, W. An association thesaurus for information retrieval. In *Proceedings of RIAO 94* (1994), pp. 146–160.

[30] Kraaij, W. Viewing stemming as recall enhancement. In *Proceedings of the 19$^{th}$ Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (1996), pp. 40–48.

[31] Krovetz, R. Viewing morphology as an inference process. In *Proceedings of the 16$^{th}$ International Conference on Research and Development in Information Retrieval* (1993), pp. 191–202.

[32] Lesk, M. E. Word-word association in document retrieval systems. *American Documentation 20*, 27 (1969).

[33] Lewis, D., Croft, W. B., and Bhandaru, N. Language-oriented information retrieval. *International Journal of Intelligent Systems 4* (1989), 285–318.

[34] Lewis, D. D. *Representation and Learning in Information Retrieval*. PhD thesis, University of Massachusetts at Amherst, 1992.

[35] Lovins, J. Development of a stemming algorithm. *Mechnical Translation and Computational Linguisitics 11* (1968), 22–31.

[36] Lu, A., Ayoub, M., and Dong, J. Ad hoc experiments using EUREKA. In *Proceedings of the TREC 5 Conference* (1997), NIST, Ed. To appear.

[37] McCarn, D. Medline: An introduction to on-line searching. *Journal of the American Society for Information Science 31*, 3 (1980), 181–192.

[38] Meghini, C., and Straccia, U. A relevance terminological logic for information retrieval. In *Proceedings of the 19$^{th}$ Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (1996), pp. 197–205.

[39] Miller, G. Special issue: WordNet: An on-line lexical database. *International Journal of Lexicography 3*, 4 (1990).

[40] Minker, J., Wilson, G., and Zimmerman, B. An evaluation of query expansion by the addition of clustered terms for a document retrieval system. *Information Storage and Retrieval 8* (1972), 329–348.

[41] Pearl, J. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers, 1988.

[42] Peat, H. J., and Willet, P. The limitations of term co-occurrence data for query expansion in document retrieval systems. *Journal of the American Society for Information Science 42*, 5 (1991), 378–383.

[43] Ponte, J., and Croft, B. Text segmentation by topic. Unpublished manuscript.

[44] Ponte, J., and Croft, B. USeg: A retargetable word segmentation procedure for information retrieval. In *Symposium on Document Analysis and Information Retrieval* (1996).

[45] Popovic, M., and Willett, P. The effectiveness of stemming for natural-language access to Slovene textual data. *Journal of the American Society for Information Science 43*, 5 (1992), 384–390.

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

[47] Qiu, Y., and Frei, H. P. Concept based query expansion. In *SIGIR 93* (1993), pp. 160–169.

[48] Rice, J. *Mathematical Statistics and Data Analysis.* Duxbury Press, 1995.

[49] Riloff, E., and Lehnert, W. Information extraction as a basis for high-precision text classification. *ACM Transactions on Information Systems 12*, 3 (1994), 296–333.

[50] Robertson, S. E. The probability ranking principle in IR. *Journal of Documentation 33*, 4 (Dec. 1977), 294–304.

[51] Rocchio, J. J. *Relevance feedback in information retrieval.* Prentice-Hall, 1971, ch. 14.

[52] Ruge, G. Experiments on linguistically based term assoications. In *RIAO'91* (1991), pp. 528–545.

[53] Salton, G. Automatic term class construction using relevance–a summary of work in auotmatic pseudoclassification. *Information Processing and Management 16* (1980), 1–15.

[54] Salton, G. *Automatic Text Processing.* Addison Wesley, 1989.

[55] Salton, G., and Buckley, C. On the use of spreading activation methods in automatic information retrieval. In *Proceedings of the Eleventh International Conference on Research and Development in Information Retrieval* (1988), pp. 147–160.

[56] Salton, G., and Buckley, C. Improving retrieval performance by relevance feedback. *Journal of the American Society for Information Science 41* (1990), 288–297.

[57] Salton, G., and Lesk, M. *Computer evaluation of indexing and text processing.* Prentice-Hall, 1971, pp. 143–180.

[58] Schutze, H., and Pedersen, J. O. A cooccurrence-based thesaurus and two applications to information retrieval. In *Proceedings of RIAO 94* (1994), pp. 266–274.

[59] Singal, A., Buckley, C., and Mitra, M. Pivoted document length normalization. In *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (1996), pp. 21–29.

[60] Sparck Jones, K. *Automatic Keyword Classification for Information Retrieval.* Butterworth, 1971.

[61] Sparck Jones, K., and Bates, R. G. Research on automatic indexing 1974–1976. Tech. rep., Computer Laboratory, University of Cambridge, 1977.

[62] Sparck Jones, K., and Jackson, D. The use of automatically-obtained keyword classifications for information retrieval. *Information Processing and Management 5* (1970), 175–201.

[63] Tong, R., Appelbaum, L., and Askman, V. A knowledge representation for conceptual information retrieval. *International Journal of Intelligent Systems 4*, 3 (1989), 259–283.

[64] Turtle, H. R. *Inference Networks for Document Retrieval.* PhD thesis, University of Massachusetts at Amherst, 1990.

[65] Tuttle, M. S., Sheretz, D. D., Erlbaum, M. S., Olson, N., and Nelson, S. J. Implementing Meta-1: The first version of the UMLS metathesaurus. In *Proceedings of the 13th Annual Symposium on Computer Applications in Medical Care* (1989), pp. 483–487.

[66] van Rijbergen, C. J. A theoretical basis for the use of co-occurrence data in information retrieval. *Journal of Documentation* (June 1977), 106–119.

[67] van Rijsbergen, C. J. *Information Retrieval*, second ed. Butterworths, 1979.

[68] van Rijsbergen, C. J. Towards an information logic. In *Proceedings of the Twelfth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (New York, NY, 1989), N. J. Belkin and C. J. van Rijsbergen, Eds., ACM, pp. 77–86.

[69] Voorhees, E. Query expansion using lexical-semantic relations. In *Processings of the 17th ACM SIGIR Conference* (1994), pp. 61–69.

[70] Weaver, M. T., France, R. K., Chen, Q.-F., and Fox, E. A. Using a frame-based language for information retrieval. *International Journal of Intelligent Systems 4*, 3 (1989), 223–257.

[71] Wilkinson, R., Zobel, J., and Sacks-Dvis, R. Similarity measures for short queries. In *Proceedings of the TREC 4 Conference* (1996).

[72] Wong, S., Ziako, W., Raghavan, V., and Wong, P. On modeling of information retrieval concepts in vector spaces. *ACM Transactions on Information Systems 12*, 2 (June 1987), 299–321.

[73] Xu, J., Broglio, J., and Croft, B. The design and implementation of a part of speech tagger for English. Tech. Rep. IR52, CIIR, Computer and Information Science Department, University of Massachusetts, Amherst, MA 01003, 1994.

[74] Zipf, G. K. *Human Behavior and the Principle of Least Effort: An Introduction to Human Ecology.* Addison-Wesley, Reading, MA, 1949.