Natural Language Query Translation and Expansion in Information Retrieval

By

Duraid Ibrahim

A Thesis
Submitted to the Faculty of Graduate Studies
in Partial Fulfillment of the Requirements
for the Degree of

Master of Science

Department of Computer Science
University of Manitoba
Winnipeg, Manitoba

THE UNIVERSITY OF MANITOBA

FACULTY OF GRADUATE STUDIES

*****

COPYRIGHT PERMISSION PAGE

Natural Language Query Translation and Expansion in Information Retrieval

BY

Duraid Ibrahim

A Thesis/Practicum submitted to the Faculty of Graduate Studies of The University of Manitoba in partial fulfillment of the requirements of the degree of

Master of Science

ABSTRACT

Query formulation and expansion have long been explored for enhancing query effectiveness and solving the word mismatch problem in information retrieval systems. Most of the approaches are statistical in nature. They are based on the occurrence frequency of words in documents. In this thesis, we present a new approach based on natural language processing.

Given a natural language query, our approach will translate a natural language query into a Boolean query that is better, in terms of retrieval effectiveness, than the original query. The terms in the Boolean query are assigned weights based on their contribution to the semantic of the query, which is determined by its occurrence frequency and its syntactic dependency within the query. Furthermore, the resulting weighted Boolean query can be further improved by expanding the query terms with synonyms in a very restrictive fashion. This process is fully automated and does not require human intervention. Experiments run for TREC-4 queries showed consistent improvement over standard information retrieval ranking methods.

Acknowledgements

I would like to thank my thesis supervisor Dr. Dekang Lin for providing me with many suggestions. He was always available to answer questions at any time any day.

I also would like to thank my wife Jocelyne for her help, encouragement, and support with all the long hours and sleepless nights to finish this work.

CONTENTS

CHAPTER 1 - INTRODUCTION

1.1 Motivation

With the boundless amount of information stored in computer systems, retrieving and filtering specific stored information is becoming a very important issue among researchers and practitioners. There is a great demand for powerful information retrieval systems that can retrieve and filter such information in a very effective manner. The demand will continue to grow with the increasing popularity of the World Wide Web and computers in general.

Formulating precise and effective queries is not an easy task. Even experienced users may not have the time to create well-formulated queries. Users can use different words than authors for describing the same concept. For example, consider the query:

What is the effect of temperature on crops?

This query will retrieve all documents that contain the word *temperature*, the word *crops*, and the word *effect*. This seems logical but it is inadequate if we consider that there may be some documents that describe the effect of cold weather on crops. Such documents will not be retrieved because the query contains the word *temperature* while the documents contain the word *cold*. One way of improving the user's query effectiveness is by expanding the query (i.e. adding synonyms). Here is an example of the potential expansion of the word *temperature*:

temperature → cold, hot, mild, weather, ...

As can be seen from the above example, automatically adding words to the user's initial query (query expansion) is a very important area of research.

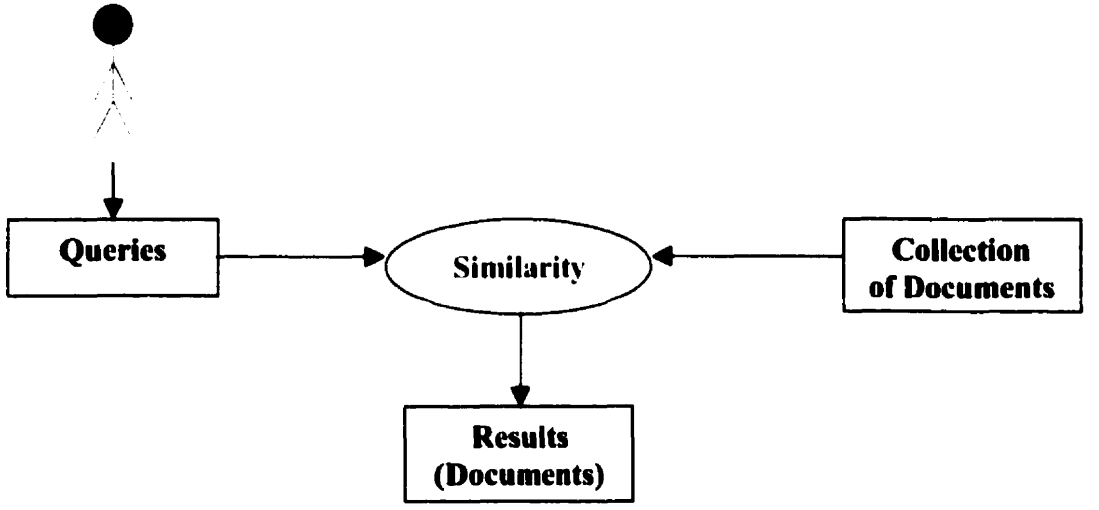
Another way to improve retrieval effectiveness is to use advanced features provided by information retrieval systems. Although some information retrieval systems include advanced features such as Boolean operators, many users do not bother to learn them and in some cases may not even know that they exist. A good example is the Internet search engine AltaVista. Very few users of AltaVista are aware that it comes with Boolean operators to form advanced and complex queries. Automatically transforming the user's natural language query into a more powerful query representation (e.g. Boolean form) improves retrieval performance and hides the complexity of the system from the user.

1.2 Information Retrieval

Information retrieval (IR) is a discipline that deals with the representation, storage, organization, indexing, searching, and accessing of information items. Information items in information retrieval are typically text or other unstructured forms [Salton and McGill 1983] [Kelledy 1997]. With the tremendous growth in science and technology, the number of publications has dramatically increased. The rate of increase of available information is becoming immeasurable. Though interest in this discipline is not new, as the amount of information available continues to grow, so does interest in IR.

Conceptually, an information retrieval system contains a collection of documents, queries, and a similarity function that matches a set of documents to a query. This is illustrated in Figure 1-1. A user specifies a query and submits it to the IR system. The IR system then retrieves all the documents that match the query using a similarity function.

Figure 1-1 Information Retrieval System - Conceptual Overview



In information retrieval, a document is a piece of information stored and labeled in a computer system. Examples of documents include text files, HTML files, newsgroup postings, and pictures or images. A group of documents that a user wants to retrieve from is called a collection.

A query is a sequence of words that describes and specifies the information the user wants. One form of a query is a natural language query. In this form, a user can specifies his or her query using plain English (or any other natural language). An example of such a query is "*which disk drive should I buy for my Mac?*" Another form of a query is the Boolean form. In a Boolean query, the words of the query are combined with Boolean operators such as AND and OR. Parentheses are usually provided to resolve the order of precedence. An example of a Boolean query is

(mandarin OR navel) AND orange

The goal of a query is to retrieve all documents that match the query. Words in the query are connected conjunctively (AND-ed) to reduce the result set and disjunctively (OR-ed) to broaden the scope of the query. To retrieve the matching documents, a similarity function is used. The similarity function determines how closely a document matches a query. A simple form of a similarity function might consider only documents that contain all the words in the query to be similar.

1.3 Performance of Information Retrieval Systems

The two standard metrics of IR systems performance are recall and precision. Neither of these two measures can be dealt with without considering the other:

- *Recall* is defined as the number of relevant documents retrieved divided by the total number of relevant documents in the collection. For example, suppose there are 80 documents that are relevant to query Q in the collection and an IR system that retrieves a total of 60 documents among which 40 are relevant to Q. The recall of this system for the query Q is 40/80 = 50%. Since any system can achieve a recall of 100% if all documents in a collection are retrieved, a system attempts to maximize both recall and precision simultaneously.

- *Precision* is defined as the number of relevant documents retrieved divided by the total number of documents retrieved. In the above example, an IR system has retrieved a total of 60 documents but only 40 of them are relevant to Q. In this case, the precision of this system for the query Q is 40/60 = 67%.

1.4 Objective of This Thesis

Traditional IR systems treat queries and documents as "bags of words". For a given query, the retrieved documents are ranked based upon the number of occurrences of the words in documents that match the query words. The retrieval methods are purely statistical. Some researchers have explored the use of Natural Language Processing (NLP) techniques, in addition to the statistical approaches, to enhance the retrieval performance. Unfortunately, the empirical results of using NLP techniques have generally not provided evidence of improved performance [Strzalkowski, Lin, and Perez-Carballo 1997]. On the other hand, some NLP techniques have proven to be effective. Examples of such techniques are the use of phrasal terms (e.g., joint venture) and proper noun extraction [Strzalkowski, Lin, and Perez-Carballo 1997]. Phrase normalization has also been used in information retrieval [Strzalkowski, Lin, and Perez-Carballo 1997]. Phrase normalization involves finding <head, modifier> pairs in text using syntactic analysis. For example, the phrases *information retrieval*, *retrieval of information*, and *retrieve more information* are reduced to the pair <retrieve, information>.

In this thesis, we study the enhancement of the query component of information retrieval. The goal of query enhancement is to improve retrieval effectiveness. Two methods will be considered to achieve this improvement. The first method is translating natural language queries into fuzzy Boolean queries. The second method is query expansion, which is applied on top of the query translation.

1.5 Outline of the Thesis

In this chapter we have introduced the components of information retrieval systems, their importance, and what components are of interest to this thesis. Chapter 2 will present background information and a literature review of query representation, formulation, and expansion, including the latest research and development in this area. Our approach for enhancing query performance will be presented in chapter 3. Chapter 4 presents the experimental results of our approach. Finally, Chapter 5 states our conclusion and outlines future work.

CHAPTER 2 - LITERATURE REVIEW

2.1 Introduction

As mentioned in the previous chapter, the objective of any information retrieval system is to store information (e.g., indexing) about a set of documents and retrieve all relevant documents for specific user queries. There are two goals to be achieved by IR systems. The first goal is to maximize the number of relevant documents retrieved in the result set for user queries (i.e. to maximize recall). The second goal is to improve the quality of the result set by placing the most relevant documents first. The two goals are achieved by identifying the importance of each word in documents and queries.

A document (or a query) consists of a set of words. The basic constituent of a text document and a query is the word. In information retrieval terminology, a single word or concept that appears in a document or a query is called a **term**. In the example, "*What is the effect of temperature on crops?*", the words *temperature*, *crops*, and *effect* would each be considered a term. Each term in a collection is assigned a weight. The weight represents the importance of the term among other terms in the collection in regards to content. Terms are used as content identifiers. Therefore, it is the terms in the document that have the highest weights that determine the documents' contents.

2.1.1 Term Weighting

The occurrence frequency of a term t in document d is called *term frequency*, *tf*. The total occurrence frequency of a term t in the collection is the total of the term frequencies of all documents in the collection [Salton and McGill 1983]:

$$Total\ tf = \sum_{d=1}^{n} tf_d$$

where n is the total number of documents in the collection.

The terms with very high and very low frequencies are ignored. Only terms with medium frequency are considered. The range of medium frequency is usually determined by an arbitrary threshold. The elimination of high frequency words could degrade recall because eliminating broad terms causes less documents to be retrieved. To add to the problem, the elimination of low frequency words could degrade both precision and recall. This is because missing some very specific and uncommon terms causes relevant documents to be missed.

The use of this absolute frequency measure sometimes hurts retrieval performance. A term might be important in one collection while irrelevant in another. For example, the term *computer* is considered irrelevant if all the collection is related to computers. This is because the term *computer* may appear in every single document in the collection. This makes the term *computer* a very poor content identifier, and hence, will not discriminate relevant documents.

Calculating the occurrence frequency of a term in a document, *tf*, is not enough to determine the importance of that term. The number of documents that contain that term should be considered

as well. The total number of documents in the collection that contain the term *t* is called *document frequency* - *df*. The importance of a term *t* is proportional to the occurrence frequency of *t* in a document, *tf*, and inversely proportional to the total number of documents in the collection that contain *t*. The *inverse document frequency* is defined as follows [Frakes and Baeza-Yates 1992]:

$$idf = \log_2 \frac{n}{df}$$

where *n* is the number of documents in the collection.

For example, consider a collection of 1000 documents. Assume the term *computer* occurred in 800 documents, the term *program* occurred in 400 documents, and the term *floppy* occurred in 100 documents. The inverse document frequency *idf* of *floppy* is the highest among the three words. More specifically:

$$\log_2\left(\frac{1000}{800}\right) < \log_2\left(\frac{1000}{400}\right) < \log_2\left(\frac{1000}{100}\right)$$

Hence, the terms with higher document frequency have a lower inverse document frequency. In other words, common terms would have low *idf* values while the highest *idf* value is assigned to terms unique to one document.

Thus, terms that are considered good content identifiers would appear frequently in individual documents and rarely across the collection. Therefore, the weight of a term *t* could be calculated as follows [Frakes and Baeza-Yates 1992]:

$$w_i = tf \times idf$$

In this function, t will have a high weight if it occurred in few documents. Specifically, relative frequency identifies the terms that are used very frequently in some documents (i.e. have high frequency) and have low frequency in the rest of the collection.

2.1.2 Stop List

A stop list is a list of words that have very high frequency. These words (also called *stop words*) are usually ignored due to their limited semantic content and, therefore, are very poor discriminators of relevant documents. Table 2-1 [Salton and McGill, 1983] lists some example stop words.

Table 2-1 Stop List

about	alone	an	are	becomes	beside
across	along	and	around	becoming	besides
after	already	another	as	been	between
afterward	also	any	at	before	beyond
again	although	anyhow	be	beforehand	both
against	always	anyone	became	behind	but
all	among	anything	because	being	by
almost	amongst	anywhere	become	below	can

Stop words comprise 40 to 50 percent of words in text. In some cases where a specialized collection is used, terms that might typically be considered important should be treated as a stop words because of their insignificance as a content identifier. The term *computer* would probably be a stop word if a computer related collection is used.

2.1.3 Stemming

Stemming is the process of removing prefixes and suffixes from words in a document or query [Frakes and Baeza-Yates 1992]. The goal of such an algorithm in information retrieval is to group words that share the same conceptual meaning. For example, the endings of words *walked*, *walker*, and *walking* are all stemmed to the word *walk*. A commonly used stemming algorithm is the Porter stemmer. There are advantages in using stemming algorithms. The first advantage of stemming is to minimize the use of disk space. In the previous example, instead of storing all the forms of the word *walk*, only the term *walk* is stored in the information retrieval database to represent all its forms. The other advantage is broadening the likelihood of word match in the retrieval process. On the other hand, stemming algorithms have a drawback. Stemming the word *porter* in *Porter stemmer* to *port* would cause an IR system to retrieve documents about boats or wine.

2.1.4 Vector Space Model

Documents (or queries) can be represented using terms. Only the meaningful (e.g. non-stopwords) are used to represent documents. In this model, each document is converted into a vector of terms. In other words, a document d in a collection can be represented as [Frakes and Baeza-Yates 1992]:

$$d = < t_1, t_2, t_3, \ldots, t_m >$$

where t_i is the weight of a term in the collection.

The terms represent the features of a vector. The length of the vector m is the total number of terms in the collection. In other words, the vector of a document contains all the meaningful,

stemmed, and distinct terms in the collection. If a term does not exist in a document, the value of that term is set to zero. For example, consider a document vector with three terms, *information*, *retrieval*, and *performance*. $d = <5, 2, 0>$ means that document d contains the term *information* and the term *retrieval* with weights 5 and 2, respectively. The document d does not contain the term *performance*. The *tf*idf* weighing scheme is often used with this model. The Boolean scheme can also be used with this model. In the Boolean scheme, the weight of a term is either 1 or 0 depending if the term exists in the document or not. Queries are also represented as vectors in the same way as documents. A query q can be defined as a vector $<t_1, t_2, ..., t_k>$, where t_i represents the weight, or importance, of term in query q.

2.1.5 Similarity and Ranking

Similarity is the measure of how similar a document and a query are. The retrieval process does not require a full match between the terms in query and document vectors. An IR system typically uses a similarity measure with a threshold to determine the similarity between a query vector and all the document vectors in the collection. The document vectors that score the best are retrieved first. Different similarity measures have been developed. One example is the inner product similarity measure. In this measure, the inner product is applied to the corresponding term vectors [Buckley, Mitra, Walz, and Cardie, 1997]:

$$S(D_i, Q_j) = \sum_{k=1}^{t} (d_{ik} * q_{jk})$$

In this formula, the similarity between a document D_i and a query Q_j is determined by the inner product of the terms in the document vector and query vector. Documents that are highly similar to the query are ranked higher. IR systems typically display the most similar documents first.

2.1.6 Query Expansion and Relevance Feedback

Query expansion is the process of building a new query from the initial user query. The new query is built by adding new terms from other documents in the collection or by adding synonyms from a thesaurus. The added words should be very effective and improve retrieval performance. Since query expansion broadens the scope of the query concepts, it results in improved system recall. Query expansion deals with the fundamental issue of word mismatch in information retrieval. As mentioned earlier, word mismatch happens when users select different words to describe concepts in their queries than authors do in describing the same concepts in their documents.

In *relevance feedback*, users are expected to provide the system with relevance judgment. When a query is submitted to the IR system, the user indicates the relevant documents from the result set. Then the system uses the terms in these relevant documents as expansion terms. The expansion terms are then added to the initial query. The newly formed query is submitted again to the IR system. The result set of this newly formed query is expected to contain more relevant documents than the initial result set.

The most commonly used relevance feedback procedure is the Rocchio's method. In this method, new terms are added from relevant documents and initial query terms are re-weighted [Frakes and Baeza-Yates 1992]:

$$Q_{new} = Q_{init} + \beta \sum_{i=1}^{n1} \frac{R_i}{n_1} - \gamma \sum_{i=1}^{n2} \frac{S_i}{n_2}$$

where

Q_{init} = the vector for the initial query
R_i = the vector for relevant document *i*
S_i = the vector for non-relevant document *i*
n_1 = the number of relevant documents
n_2 = the number of irrelevant documents

Terms that exist in the relevant documents and do not exist in the initial query are added to the query. Weights of initial query terms are increased if they appear in the relevant documents and reduced if they appear in the non-relevant documents. Terms in relevant documents contribute with positive weights while terms in non-relevant documents contribute with negative weights. Typically, negative-weight terms are not added to the query.

The problem with this method of expansion is that users usually do not have the time to provide the system with such feedback. To address this, IR systems developers provide a "pseudo" relevance feedback [Mitra, Singhal, and Buckley 1998] where users do not have to provide any feedback. In this approach, the initial query is presented to the IR system and the top *n* documents (e.g. $n = 20$) are used for the relevance feedback procedure. The top *n* documents are assumed to be the most relevant to the initial query. Therefore, under this assumption, these documents are considered the best candidates for expanding the terms of the initial query. The additional terms for expansion are taken for these documents using Rocchio's method.

2.2 The TREC Collection

Test collections are created for evaluating experimental information retrieval systems. They usually consist of a set of documents and queries. One of the most common collections used in evaluating information retrieval is the TREC collection.

TREC (Text REtrieval Conference) is an annual conference for evaluating IR systems. IR researchers around the world can participate in this conference. This conference has been held six times. To identify the conference and its collection, the number of the conference is used immediately after the word TREC. For example, TREC-4 means the fourth conference and collection. Since all participants use the same document collection (TREC), systems are evaluated by comparing and contrasting among them. The conference is funded by the National Institute of Standards and Technology (NIST) and the Defense Advanced Research Projects Agency (DARPA).

The TREC collection consists of a large number of queries and their relevant documents, all created by humans. It is diverse and covers multiple domains. The collection comes with relevance feedback that enumerates all the relevant documents for a given query. This enumeration is done by humans and is used to test the performance of IR systems.

TREC documents are obtained from different sources such as newspapers and abstracts of technical papers. Documents contain SGML tags to describe the different components of the document such as document number, title, and text. The following query (207) is representative of TREC-4 queries:

What are the prospects of the Quebec separatists achieving independence from the rest of Canada?

Here is a sample of a document from TREC-4 that is relevant to query 207:

```
<DOC>
<DOCNO> AP900623-0028 < DOCNO>
<FILEID>AP-NR-06-23-90 0322EDT< FILEID>
<FIRST>r i PM-Canada-Quebec 06-23 0733< FIRST>
<SECOND>PM-Canada-Quebec, 0764< SECOND>
<HEAD>Failure of Accord Causes Resentment But No Immediate Secession
Move< HEAD>
<HEAD>With PM-Canada< HEAD>
<HEAD>An AP News Analysis< HEAD>
<BYLINE>By JEFFREY ULBRICH< BYLINE>
<BYLINE>Associated Press Writer< BYLINE>
<DATELINE>MONTREAL (AP) < DATELINE>
<TEXT>
   The failure of the Meech Lake constitutional accord will be strongly
resented in much of French-speaking Quebec, but an immediate march toward
independence is unlikely.
   The proposed agreement would have met Quebec's principal conditions for
signing Canada's 1982 constitution. The most important and most symbolic of
its provisions was a clause calling Quebec a "distinct society."
   The "distinct society" clause had no immediate practical impact on the
province or its people. It served mainly to soothe bruised Quebecois feelings.
   The clause was also a guideline for the Canadian Supreme Court, requiring
it to take the province's special situation into account when ruling on
challenges to laws affecting Quebec.
   But the 1987 agreement was declared dead Friday night after it fell short
of ratification by all 10 provinces, which was required by this weekend.
   A widened political debate in the province seemed certain. An eventual
referendum on either independence, or more likely some sort of sovereignty-
association with Canada is possible.
   But a highly passionate response, or a violent separation, does not appear
to be in the cards.
   During the long debate, Quebec officials were careful not to make any
threats of secession or other future action during the long debate. They
refused even to acknowledge that any contingency plan was in the works should
the accord fail.
   Quebecois voters rejected a 1980 referendum that proposed talks with the
federal government on political autonomy for their province.
   The Meech Lake accord gave all provinces, not just Quebec, the right to opt
out of some federal-provincial spending programs. It gave provinces more say
in immigration _ particularly important in Quebec, where French-speaking
immigrants are desirable. It also covered nominations to the Supreme Court and
a new constitutional amending formula.
   The accord died because opposition prevented ratification in the provinces
of Manitoba and Newfoundland.
   In Manitoba, a Cree Indian leader who is a member of the provincial
legislature sucessfully submarined attempts to push through approval by
```

throwing up procedural obstacles. He forced adjournment of the legislature until Monday, two days past tonight's midnight deadline for passage.

That meant there could be no unanimous approval by the provinces.

In Newfoundland, Premier Clyde Wells, hearing the news from Manitoba, postponed a vote on Meech Lake indefinitely.

"It's a sad day for Canada," Prime Minister Brian Mulroney said after the death of the accord.

In fact, most Canadians were sick of the discussion and debate, which had occupied the headlines for the past several months, and with increasing intensity for the past several weeks.

Some Quebecois with nationalist or separatist feelings were happy, believing the rejection of the accord would be a step toward independence. Others were bitter.

"Try to sit with us the next time and see what our basic demands are," said Serge Dube, a power company executive. "The bare minimum was asked, and they trampled on our flag. We were told: We don't want you in Canada. No, Canada, we can't accept that."

"For a while it looked good," computer systems analyst Robert Gagnon said of the accord. "It looked like it was going to bring the country together."

Quebec Premier Robert Bourassa expressed "profound Disappointment" at the failure of the agreement but gave no indication what his next move would be.

Jacques Parizeau, leader of the separatist Parti Quebecois opposition party, offered to sit down with Bourassa, a Liberal, to discuss the future of the province.

Quebec's 6.5 million inhabitants represent a quarter of Canada's population. Montreal is its second city. Quebec's economy is an integral part of the Canadian economy and the province is the country's largest.

The Meech Lake debate already was having an impact on foreign investment and on the Canadian dollar because it raised fears about Canada's stability.

But widespread opinion among Quebecois was that there probably would be an expression of strong feeling this weekend on St. Jean Baptiste Day, the Quebec national day. Then, the conventional wisdom said, people would head off for vacation and generally forget about it.

One constitutional specialist _ asked before the measure died what the immediate impact of its failure would be _ offered a low-key response.

"They'll probably form a committee," he said.

</ TEXT>

<TEXT>

EDITOR'S NOTE: Jeffrey Ulbrich is the Associated Press bureau chief in Toronto.

</ TEXT>

</ DOC>

2.3 WordNet

WordNet is a general-purpose lexical database. WordNet was manually constructed by a team of researchers at the Cognitive Psychology Lab in Princeton University. Nouns, verbs, adjectives, and adverbs are organized into synonym sets called **synsets**. A synset of a word represents one of its senses. Each synset may have a list of synonyms of that sense. Synsets are connected with

each other using lexical relationships. The type of lexical relationship is dependant on the synset's part-of-speech. For example, nouns have *is-a* and *part-of* relationships. Figure 2-1 shows all the six senses of the word *swing* [Voorhees 1994].

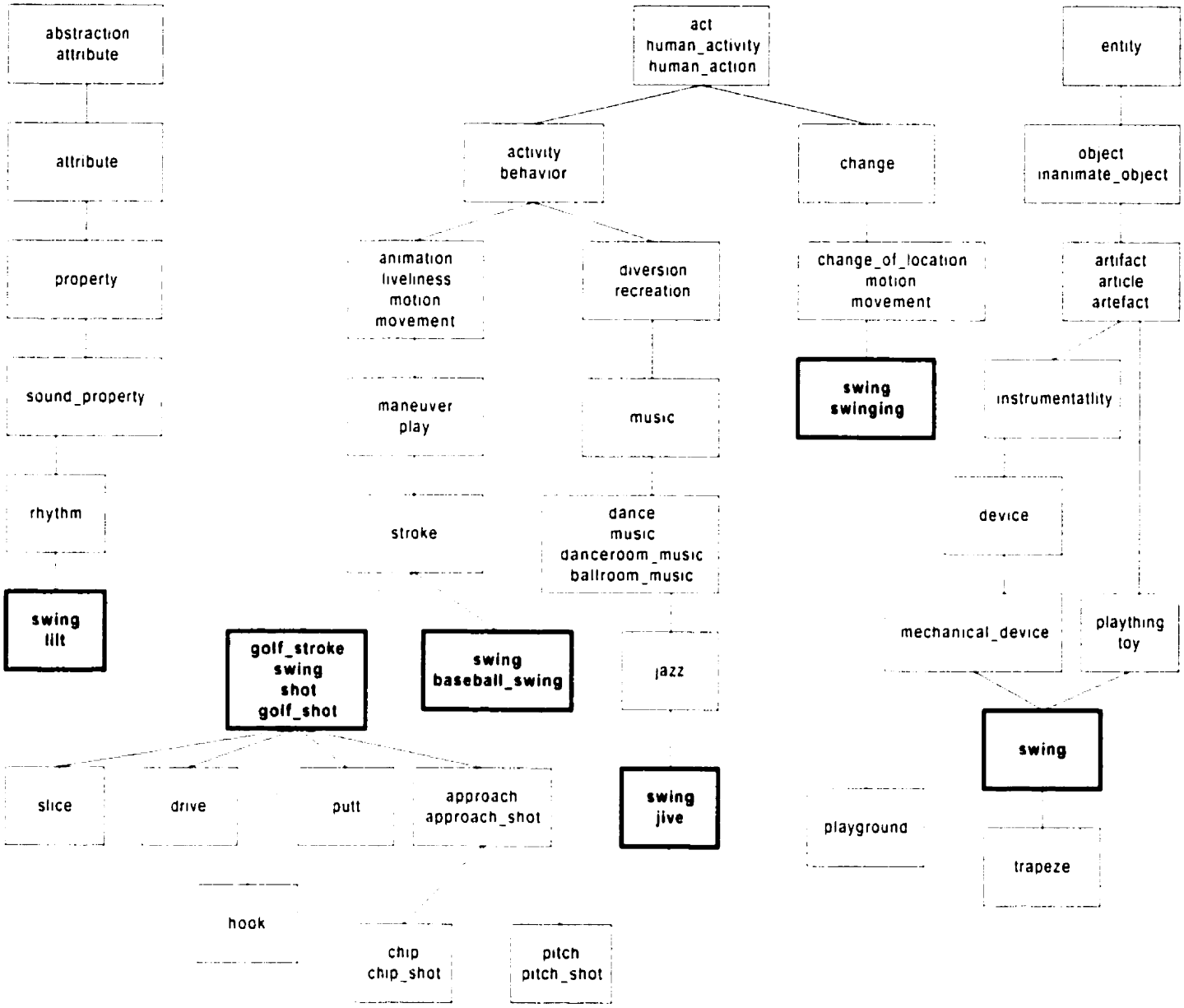
One can retrieve all senses of a specific word and its part-of-speech. Here is an example that shows the WordNet output for all the senses of the word *snow* as a noun:

```
Sense 1
snow, snowfall -- (precipitation falling from clouds in the form of ice crystals)
       => precipitation, downfall -- (the falling to earth of rain or snow or hail or
sleet or mist)

Sense 2
snow -- (a layer of snowflakes (white crystals of frozen water) covering the ground)
       => layer -- (a relatively thin sheetlike expanse or region lying over or under
another)

Sense 3
cocaine, cocain, coke, snow, C -- (a narcotic (alkaloid) extracted from coca leaves;
used as a surface anesthetic or taken for pleasure; can become addictive)
       => hard drug -- (a drug that is considered relatively strong)
```

Figure 2-1 The six senses of the word *Swing* in WordNet.



2.4 Short User Queries

As mentioned earlier, most users of IR systems do not have the time to spend formulating their queries. For this reason they tend to use short queries limited to a few key words related (in the users' opinion) to the information they need. For example, the average number of words of a query recorded on Internet search engines is two [Croft 1995]. Short queries tend to have limited coverage due to term mismatch between queries and documents. This problem decreases with longer queries. This is because with longer queries, the possibility of co-occurrence between words in the queries and documents is higher. It has been observed in TREC conferences that long queries perform better than short queries [Voorhees 1994]. In TREC-4 [Harman 1996], the score of the top ranked documents decreases with shorter, more realistic queries. Appendix A lists all queries from TREC-4. The average number of words per query (including stopwords) in TREC-3 is 105.28 while the average number of words per query in TREC-4 is 39.46.

Due to the wide usage of short queries in many applications, improving the retrieval performance of short queries has been of great interest to researchers and practitioners.

2.5 Query Enhancements

Researchers have been working on the problem of query expansion since the 1970s [Vlez, Weiss, Sheldon, and Gifford 1997]. There has been many research in this area. One of the earlier query expansion techniques made use of general-purpose thesauri [Frakes and Baeza-Yates 1992]. This approach did not show any improvement to the initial query retrieval performance. Results from expanding a query using general-purpose thesauri could degrade query performance,

producing results that are worse than the original query. This is partly due to word sense ambiguity. For example, the word *go* has more that 40 entries in the dictionary [Allen 1995], many of which may reflect a new sense of the word *go*. Some synonyms for the word *go* are *move*, *depart*, *pass*, *vanish*, *reach*, and *extend*. Many of these represent a different sense for the verb *go*. Even within a sense, expansion that includes all synonyms may negatively impact results.

Most of the work done in automatic query expansion or modification is based on term co-occurrence. If a query term occurs frequently in the relevant documents then the term weight is increased, while if the term occurs frequently in the non-relevant documents, its weight is decreased. Terms with high weight are usually added to the query.

There are two types of query expansion, *automatic* and *interactive*. Unlike interactive query expansion, automatic query expansion does not require intervention on the part of the user.

Automatic query expansion can be divided into two types, *global* and *local* [Xu and Croft 1996]. Global query expansion is done using thesauri. In this method, an IR system selects the words that are synonyms to the query terms from a thesaurus. The thesaurus could be manually or automatically constructed. An example of manually constructed thesaurus is WordNet. Automatically constructed thesauri are created by automatically analyzing word occurrences and relationships in the corpus as a whole. Expansion makes use out of this information to add new terms to queries. There is weak evidence that use of a general thesaurus improves the effectiveness of the search [Voorhees 1994].

Local query expansion is done using local document analysis. Relevance feedback is based on terms from the most highly ranked documents in the result set. This method involves selecting a set of terms only from the top *n* ranked documents retrieved by the original query. Queries are then re-weighted and expanded based on these terms.

2.5.1 Xu and Croft's Method

[Xu and Croft 1996] presented a new technique that is a combination of the global and local document analysis called Local Context Analysis. Query terms (concepts) being considered in this approach are noun groups. A noun group (phrase) is a single noun, two adjacent nouns, or three adjacent nouns. This approach differs from others in the definition of a concept.

Typically, concepts are all non-stop words in the query. This creates a restriction on the type of concepts that can be expanded. The main idea of this approach is to break the top ranked documents into passages (a passage is a text window of fixed size). Concepts are constructed from terms within the passages (300 words in size) rather than the entire document. The rationale for considering passages rather than entire documents is that if there are two concepts, one occurring at the beginning of the documents and the other occurring at the end, they are most likely unrelated. On the other hand, if two concepts occur relatively close to each other (within a passage), then there is a good chance that they are related.

Experiments were conducted for this approach and results compared with results of experiments conducted on the global and local analysis approaches. The local context analysis outperformed the other two. This approach uses the following algorithm:

1. Retrieve the top n ranked passages (in this experiment a 300 word window size was used)
2. Concepts (or noun phrases) in the top n passages are ranked according to the following function:

$$bel(Q,c) = \prod_{t \in Q} (\delta + \log(af(c,t_i)) \frac{idf_c}{\log(n)})^{idf_t}$$

where

$$af(c,t_i) = \sum_{j=1}^{j=n} ft_{ij} \cdot fc_j$$

$$idf_i = \max(1.0, \frac{\log_{10}(N / N_i)}{5.0})$$

$$idf_c = \max(1.0, \frac{\log_{10}(N / N_c)}{5.0})$$

c = a concept
ft_{ij} = the number of occurrences of t_i in p_j
fc_j = the number of occurrences of c in p_j
N = the number of passages in the collection
N_i = the number of passages containing t_i
N_c = the number of passages containing c
$\delta = 0.1$ in this paper to avoid zero bel value

In the above function, the *af* component gives more weight to concepts that co-occur more frequently with query terms. The idf_c component reduces the weight of concepts that occur frequently in the entire collection. The idf_i component gives more weight to infrequent query terms. Finally, the advantage of multiplication is to emphasize co-occurrence with all query

terms. This function for local context analysis weight measurement is a variant of the *tf*idf* measure.

In local context analysis experiments against the TREC-4 collection, 70 concepts were added to each query using the *bel* function. The best run was 23.5% better than the baseline when the number of passages used was 100.

2.5.2 Velez, Weiss, Sheldon, and Gifford's Method

[Velez, Weiss, Sheldon, and Gifford 1997] presented an algorithm for interactive query expansion. Given a query and a corpus, the algorithm first finds all documents matching the user's query. It then ranks the terms in these documents according to some weight functions, and finally displays (to user) the top ranked terms as suggestions for query expansion. The algorithm works as follows:

Let

C = corpus
q = query
r = number of matching documents to consider
n = number of suggestions to display
W_f = weight function

1. Select all documents $D(q) \in C$ that match the query q.
2. Select a subset $D_r(q)$ of the top r matching documents from $D(q)$
3. Construct the set of terms $T(q)$ from the documents $D_r(q)$ such that $T(q)$ contains all the terms in all the documents in $D_r(q)$
4. Construct the subset $S \in T(q)$ such that S contains the top n ranked terms in $T(q)$ using a weight function W_f.

5. Display the suggested terms S for the query q to the user.

The discriminating feature of this algorithm is the weight function. This algorithm uses three different weight functions. The first one is the standard document frequency measure:

$$w_{df}(q,t) = \sum_{d \in (D(q) \cap D(t))} df_{t,d}$$

In this weight function, the weight of term t is equal to the document frequency of the term t in the matching documents, $D_t(q)$.

The second weight function that was used to evaluate this algorithm is the standard term frequency measure:

$$w_{tf}(q,t) = \sum_{d \in (D(q) \cap D(t))} tf_{t,d}$$

The third function takes into consideration the number of occurrence of a term t in the corpus and the size of documents that a term t is assigned:

$$w_{tfx}(q,t) = \sum_{d \in (D(q) \cap D(t))} 0.5 + \frac{0.5(tf_{t,d})}{\max_{t \in d} tf_{t,d}} * \log(\frac{N}{|D(t)|})$$

The first component of this function reduces the weight of terms that have high frequency across the corpus. To do this the term frequency weight is adjusted with inverse document frequencies. The second component of the function handles document size. It normalizes term weights by document size to counteract the increase in the weight of terms that appear in large documents.

The algorithm was evaluated using concept recall and precision improvements. Concept recall is a measure of the ability of an algorithm to recommend terms that are semantically related to the user's information need. That is, a measure of how likely the automatically generated terms are to be accepted by humans.

In evaluating this algorithm, terms suggested by the algorithm were compared to terms suggested by humans. More precisely, the evaluation is based on the ratio of terms that are suggested by the algorithm as expansion terms for a given query versus human generated terms that are candidates for expansion of the query.

Figure 2-2 illustrates the experimental results. The graph represents the number of terms suggested by the algorithm divided by the total number of terms suggested by humans versus the number of suggested terms displayed. The figure illustrates the results using 8466 documents and 150 queries. Figure 2-3, on the other hand, illustrates the results using 84660 documents. Figure 2-3 illustrates the impact of using larger data sets on retrieval effectiveness.

Figure 2-2 Expansion terms suggested by the algorithm for 8K documents [Velez, Weiss, Sheldon, and Gifford 1997].

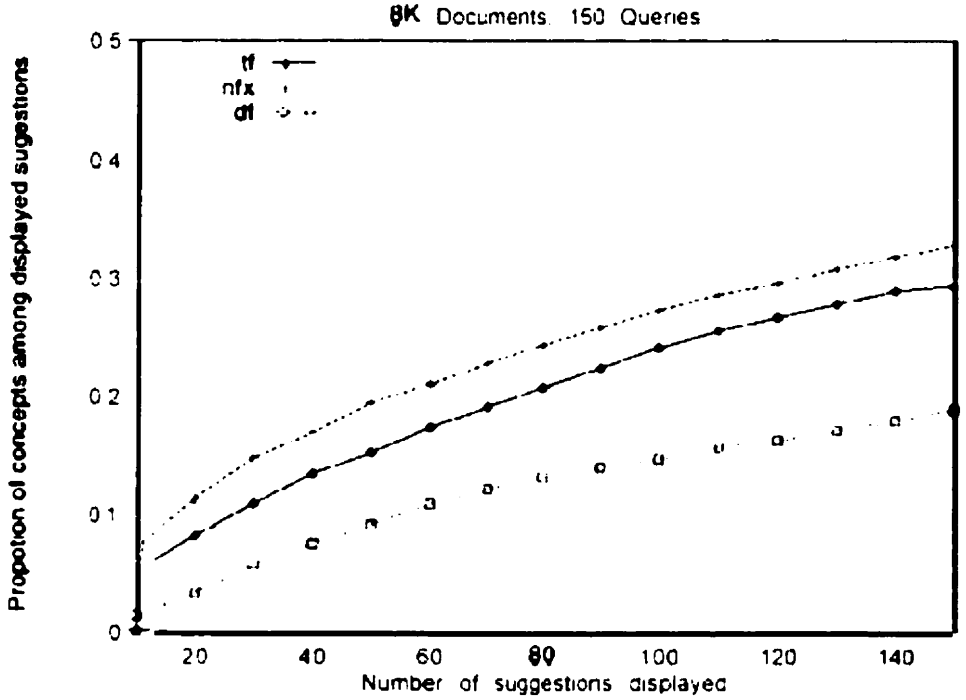
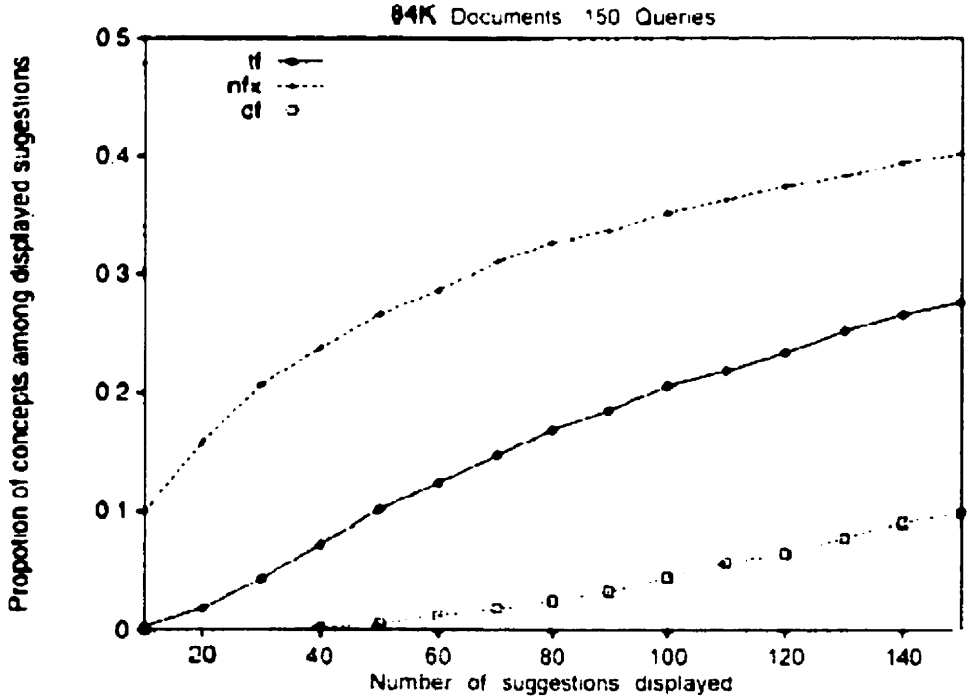


Figure 2-3 Expansion terms suggested by the algorithm for 84K documents [Velez, Weiss, Sheldon, and Gifford 1997].



Precision improvement measures the effect of adding suggested terms to the query. The experiments showed that adding useful expansion terms doesn't always improve the performance of a query. For example, adding terms such as budget, deficit, and loan to the topic economic projection decreased precision. The experimental results show that more than 50 percent of the human generated terms actually decrease precision.

2.5.3 Hearst's Method

[Hearst 1996] argues that the goal of an information retrieval system is to enhance the precision of the top ranked documents (say top 5-30 documents). This argument is based on the assumption that users are interested in the most relevant (top) documents in the result set. To achieve this increased precision on the top ranked documents two constraints (or filters) are applied.

The first one is the Boolean constraint. This filter insures that every document in the result set contains all query terms (or topics). A Boolean query in this approach consists of a conjunct of disjuncts. Each disjunct represents a topic in the query. The Boolean query is manually constructed by humans. For example, consider query 207 of the TREC collection:

What are the prospects of the Quebec separatists achieving independence from the rest of Canada?

The following Boolean query was created by the author for query 207:

Disjunct1: Quebec OR Canada
Disjunct2: independence OR separatists OR separation

The final Boolean query looks like this:

(Quebec OR Canada) AND (independence OR separatists OR separation)

Adding the topics (disjuncts) is a manual procedure while the conjunction (AND-ing) of the disjuncts is done automatically by the system. Since the separation of topics is performed by humans, it is easy to conclude that *Quebec* and *Canada* both belong to the same topic. Human intervention in term expansion is also allowed in this method. In this example the author added the term *separation* to the query to enhance retrieval effectiveness.

The second constraint is the proximity constraint. It is more restrictive than the Boolean constraint. While the Boolean constraint requires each document in the result set to have all the topics in the query, the proximity constraint requires all query topics to appear in at least one segment of every document in the result set. A segment is text window of size 100 to 300 words.

The algorithm in this method works as follows:

1. Documents are divided into segments.
2. The user specifies the disjuncts using a user interface provided by the IR system.
3. The system converts this into a Boolean query.
4. The Boolean query is processed and the proximity constraint is activated.
5. The system filters out all documents that do not have at least one segment that contains all the query topics.

6. The systems ranks the remaining documents using the standard vector space model.
7. If the resulting list of documents is not large enough the system appends the highest ranked documents that have not yet been added to the list. What constitutes "enough" is subjective.

In the experiments for this method, different document cut-off levels were used. The experimental results are not very promising when the TREC-4 collection is used. See Table 2-1 for illustration.

Table 2-1 TREC-4 Boolean/Proximity constraint results from [Hearst 1996]

Cut-off	No constraint (Baseline)	Boolean	% Improvement	*p*	Boolean Proximity	% Improvement	*p*
5	.38	.33	-13	.6	.29	-23	.3
10	.30	.31	3	.7	.34	13	.3
20	.24	.33	**38**	.03	.32	**33**	.04
30	.23	.35	**52**	.0006	.26	13	.3
100	.19	.21	11	.4	.16	-16	.3

As shown in Table 2-1, the best percentage of improvement in precision using the Boolean only is 52 at cut-off 30, while precision was at its best (33) at cut-off 20 for the Boolean proximity constraints together.

2.5.4 Mitra, Singhal, and Buckley's Method

[Mitra, Singhal, and Buckley 1998] presented a new method that combines the Boolean query and straightforward query terms to improve retrieval effectiveness. This approach was designed with the short queries in mind. The motivation of this work is the poor performance of the standard "pseudo" relevance feedback. Consider query 203 from the TREC collection:

What is the economic impact of recycling tires?

Based on relevance feedback, only 4 out of the top 20 retrieved documents were found to be relevant – a recall of 20% for this query. The reason behind the poor performance is that not all of the top 20 documents retrieved were relevant to the query. Some of the 20 documents were related to recycling but not to the recycling of tires. For example, recycling of aluminum, plastic, cans, and so on. Adding words like cans and glass changes the query concept and, as a result, irrelevant documents will be retrieved. Thus, the expansion is only as good as the top *n* documents. If the top *n* documents are relevant, good search terms words will be added to the query. Otherwise, unrelated words will be added, causing the query's performance to degrade. The goal of [Mitra, Singhal, and Buckley 1998] is to improve the relevance of the top *n* documents (i.e. improve precision of the top ranked documents). If these documents are relevant, then the expansion words are likely to be relevant and result in improved query effectiveness.

Two steps are used to improve the precision of the top rank documents. The first step is the use of Boolean constraints. In this step, all the non-stop words are AND-ed together to form the filter. To ensure the retrieval of relevant documents, all documents retrieved using this filter contain all the concepts in the query. For query 203, the Boolean query will look like this:

economic AND impact AND recycling AND tires

Any document in the collection that contains the four words will be part of the result set of this query. If straightforward query words are fed to a search engine, there is no guarantee that each of the top documents contains all four concepts. Documents might be related to recycling only, or economic impact issues that are not related to the recycling of tires. This will depend on the *idf* values of query terms. Therefore, conjunction of query terms will ensure that each of the retrieved documents contains all the concepts in the query.

The second step involves re-ranking the top ranked documents. Since the goal is to maximize the precision of the top n (say 20) ranked documents, a larger (greater than n) set k (say 50) of documents are retrieved. The k ranked documents need to be re-ranked more effectively and select the top n documents from k for the relevance feedback process. A new similarity measure is introduced for the re-ranking process. This similarity measure takes into account two aspects. The first aspect is the contribution of all query terms to the final score (rank) of a document:

$$Sim_{new}(D) = \sum_{t_i \in D} idf(t_i)$$

Where Sim_{new} is the new score for a document D and t_i is a term in the query. Sim_{new} is the *idf* summation of all terms in the query that exist in document D.

The second aspect in the similarity measure is term correlation. The authors believe that the co-occurrence of term B and another term A within the same documents makes term B less important. In other words, if two terms co-occur in documents the second term does not give much more information about the content of these documents. On the other hand, if two terms

are independent, then each term fully contributes (based on its *idf*) to the content of the document it represents. For example, consider query 248 of the TREC collection:

What are some developments in electronic technology being applied to and resulting in advances for the blind?

If the terms *electronic* and *technology* tend to co-occur together in a document, the term *technology* does not provide us with much more information about the content of that document. On the other hand, if the two terms *technology* and *blind* are considered, then the two terms contribute to the content identification because they are independent.

To implement this concept, the query terms are sorted by their document frequencies in an increasing order. The document frequency is the frequency of occurrence of a term in the k documents. The first (or most rare) term contributes with its full *idf*. The subsequent term is dependent on the previous match. If a term is highly correlated to a previous match, then its weight is reduced and, as a result, has less contribution to the retrieval effort. Thus, the Sim_{new} function can be modified to include term correlation:

$$Sim_{new}(D) = idf(t_1) + \sum_{i=2}^{m} idf(t_i) \times \min_{j=1}^{i-1}(1 - P(t_i \mid t_j))$$

where $P(t_i|t_j)$ is estimated based on word occurrences in D and is given by

$$\frac{\#\ documents\ in\ S\ containing\ words\ t_i\ and\ t_j}{\#\ documents\ in\ S\ containing\ word\ t_j}$$

Tables 2-2 and 2-3 show the experimental results using the TREC-4 collection.

Table 2-2 Feedback results using automatic filters (without term correlation information)

w → T ↓	50	100	200	Full doc
50	0.3118 2.6%	**0.3134** **3.2%**	0.3124 2.8%	0.3127 2.9%
100	.3088 1.7%	0.3124 2.9%	0.3133 3.1%	0.3123 2.8%
200	.3015 -0.7%	0.2919 -3.9%	0.2883 -5.1%	0.2912 -4.2%

Table 2-3 Feedback results using automatic filters (with term correlation information)

w → T ↓	50	100	200	Full doc
50	**0.3208** **5.6%**	0.3195 5.2%	0.3196 5.2%	0.3168 4.3%
100	0.3109 2.3%	0.3159 4.0%	0.3084 1.5%	0.3164 4.2%
200	0.2867 -5.6%	0.2867 -5.6%	0.2908 -4.3%	0.2937 -3.3%

The data in Table 2-2 is the result of applying Sim_{new} without correlation, while the data in Table 2-3 uses term correlation. Proximity constraints are used in these experiments. Documents are broken into blocks of fixed size (w) windows. The score of a document is equal to the score of its best matching block. As shown in the tables, different block sizes (or w) were used. Also, different values for the number of documents re-ranked (T) were experimented with.

As shown in the tables, Sim_{new} with no correlation performed its best at w = 100 and T = 50, while term correlation did best at w = 50 and T = 50.

2.5.5 Voorhees' Method

So far the work has concentrated on the co-occurrence of words. [Voorhees 1994] experimented with using the general-purpose lexical database WordNet for query expansion. Under this method, the candidate terms for expansion are chosen manually by humans. As a result, the outcome of this experiment is supposed to be considered the upper bound for automatic expansion of similar kind.

In this work, only nouns are considered for expansion. Synonyms are added manually to the original queries to broaden their coverage. The manually added synonyms do not have to be for a particular query term. A new word that is believed to improve performance based on an understanding of the query premise could be added. To expand a word, there are many potential words to chose from. The expansion is done using one or a combination of the following options:

1. Expansion words are the synonyms within the synset.
2. All descendants/ancestors in the is-a/part-of hierarchy of the synset.
3. All descendants and ancestors in all relations of the synset.

etc.

Since options 2 and 3 (and others like them) could add too many synonyms, a threshold can be applied to set the maximum distance between the original word and the potential synonyms. Specifically, only words that are t links or less apart from the query word are considered where t is the threshold distance. For example, consider that the synset *golf-stroke* was added to the

original query. If the expansion option is to visit all descendents with a threshold equal to 1, then words like *slice*, *hook*, *drive*, *putt*, and *approach* will be added as expansion words (see Figure 2-1). However, words like *chip* and *pitch* will be considered if the threshold used is 2.

The experimental results of this method did not show any performance improvement for long and relatively complete queries. However, some improvement in performance was achieved on less complete (or short) queries.

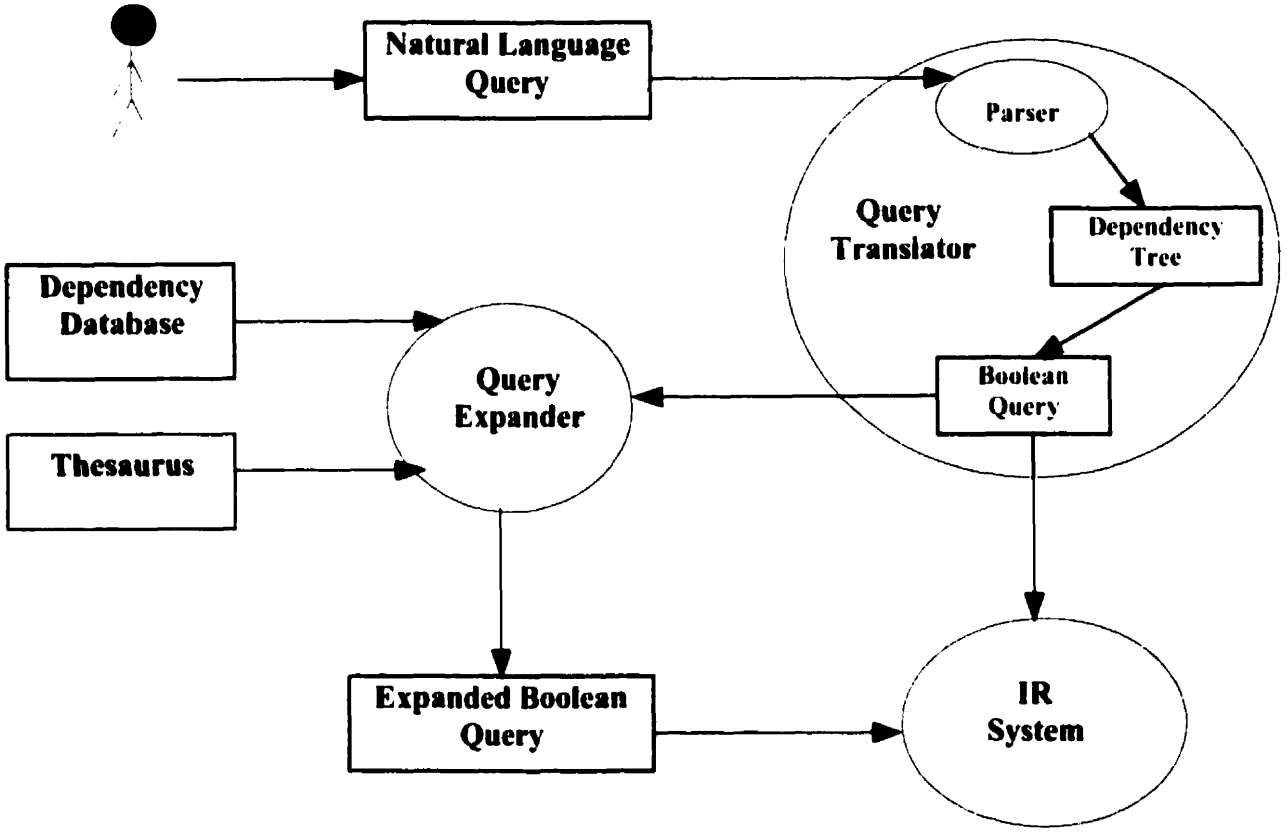
CHAPTER 3 - QUERY TRANSLATION AND EXPANSION

3.1 Introduction

To improve information retrieval performance, it is necessary to determine the relative importance of component terms and phrases. Once identified, more emphasis can be put on the important phrases (and terms in general) by increasing their weights. The goal of this weighting is improved retrieval effectiveness. Consider the query *what is the economic impact of recycling tires*? As humans, we can judge that for this query, the phrase *recycling tires* is much more important than the phrase *economic impact*. Identifying and boosting important terms within text can significantly improve the retrieval effectiveness. Furthermore, identifying important terms in queries makes term expansion more realistic and leads to better query expansion because expansion can be limited to only important terms. In the above example, adding the word *rubber* as an expansion to the word *tire* is much more effective than adding the word *effect* as an expansion to the word *impact*.

This thesis presents a solution that converts a natural language query into a Boolean query. To improve retrieval effectiveness of the automatically generated Boolean query, term expansion will be applied to the query. Only the important terms will be expanded. Important terms are those words in the user's query that, when expanded, result in a significant improvement in recall and/or precision. Natural language processing techniques (such as parsing) are used to break a query into a Boolean expression and to determine which terms are good candidates for expansion. This is the main contribution of this work since this approach has not been discussed in the literature. Figure 3-1 provides an overview of the system architecture.

Figure 3-1 System architecture overview.



A user enters a natural language query. The query is then converted into a Boolean query by the query translator component using natural language techniques. The resulting Boolean query can be fed into an IR system that supports the Boolean model. The resulting Boolean query can be further enhanced by adding new terms from a thesaurus using the query expander component. The query expander generates an expanded Boolean query that is ready to be fed into an IR system.

3.2 Query Translation

As previously described, queries are translated from natural language form to Boolean form by the query translator component. This component is broken down into five functional units. Descriptions of these functional units are presented in the next subsections.

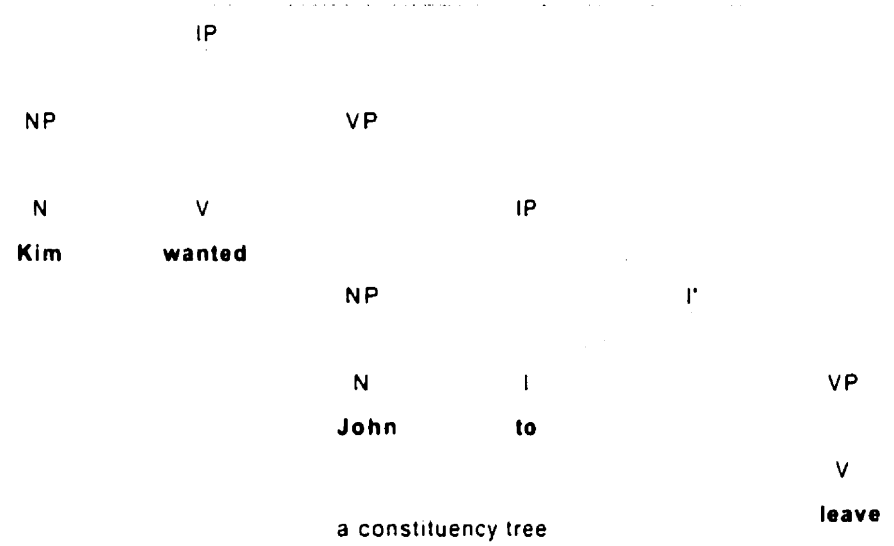
For the purposes of this work the query translator handles only single sentence natural language queries. If a query contains more than one sentence, only the first sentence will be processed. The set of queries used to evaluate this work contains mostly single sentence queries.

3.2.1 Parser and Dependency Tree

After a user enters a natural language query, the query is processed and broken down into its basic constituents by a *parser*. A parser is a program that takes a sentence as its input and breaks it down into its constituents based on a specified grammar [Allen, 1995]. The output describes the sentence's structure. This output can be presented using constituency or dependency trees.

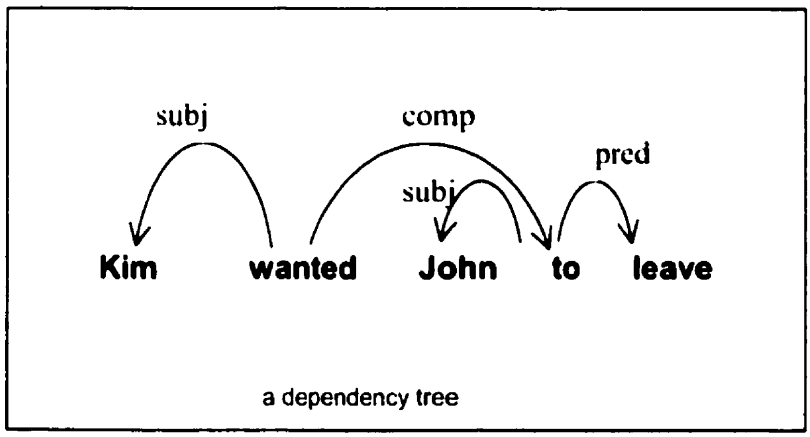
A constituency tree (or parse tree) shows how the sentence is broken down into sub-components such as noun phrases, verb phrases, etc. These sub-components are further broken down into their sub-components such as nouns, determiners, articles, verbs, etc. This breakdown continues until it reaches the individual constituents. To illustrate, the constituency tree for the sentence *Kim wanted John to leave* is presented in Figure 3-2.

Figure 3-2 Sample constituency tree.



A dependency tree shows the dependency relationships between words in a sentence. Each dependency relationship consists of a head, a type, and a modifier. For example, the dependency relationship type in *Kim wanted* is a subject where *wanted* is the head and *Kim* is the modifier. The root of the tree is the head of the sentence. The head of the sentence is the word that does not modify any other word [Lin, 1995]. Figure 3-3 illustrates the dependency tree for the sentence *Kim wanted John to leave.*

Figure 3-3 Sample dependency tree.



For the purposes of this project, the dependency tree is used. This is because the semantic structure of a sentence corresponds more closely to the dependency tree. Consider the following example for query 203. The following shows the dependency tree output for this query:

```
(
 (what          ~          N          < impact         subj)
 (is            be         Be         < impact         be)
 (the           ~          Det        < impact         det)
 (economic      ~          A          < impact         jnab)
 (impact        ~          N          *)
 (of            ~          P          > impact         of)
 (recycling     recycle    V_N        > of             pcomp-c)
 (tires         tire       N          > recycling      compl)
)
```

The dependency tree contains important information about the sentence. The first column contains all the words in a sentence. The second column is the root form of the words. The third column indicates the part-of-speech (noun, verb, adverb, adjective, etc.) of all terms in a sentence. In this example, *tire* is determined to be noun (N). The fifth column shows the term modifier. The dependency tree identifies the head of each modifier and describes the relationship between a modifier and its head. Each pair (modifier, head) that appears in the dependency tree output is syntactically dependent. In the above example, the modifier of the term *recycling* is *tire* and the syntactic dependency relationship between them is *compl*, as shown in column six. Table 3.1 shows some dependency relationships and their descriptions. Column four indicates whether the modifier appears before or after the head in the sentence. The symbol (<) means that the head appears before the modifier.

One final bit of information presented in the dependency tree is the identification of the head of the sentence. In the above example, the term *impact* is the head of the sentence. This is indicated by an asterisk.

Table 3-1 Dependency relationships.

Relationship	**Description**
compl	a word and its first complement
conj	two words connected by conjunction
det	a noun and its determiner
ji	a sentence modifier
jnab	a noun and its adjectival modifier
lex-mod	lexical modifier
nn	a noun and its nominal modifier
subj	a subject and a predicate

The parser of choice for this project was MINIPAR (previously called PRINCIPAR [Lin, 1993]). MINIPAR is a broad coverage, principal-based parser. It contains a lexicon with over 90,000 entries. MINIPAR produces both constituency and dependency trees. The results of an evaluation of the parser using the SUSANNE corpus show that dependency relationships can be correctly identified for approximately 85% of words [Lin, 1995].

3.2.2 Individual Term Weights

Typically, in information retrieval systems, terms are assigned weights based on their occurrence frequency in documents. We follow a similar approach but the frequency used in our method is more restricted. We use the occurrence frequency of a term based on its part-of-speech. For example, consider a term t in a query q. If the part-of-speech of t is a noun in the query q, then we use the occurrence frequency of t as a noun only. In other words, if t could be a noun, a verb, or an adjective, depending on the sentence that it is part of, we ignore the occurrence frequency

of t as a verb and an adjective because t occurs as a noun in the query q. This is made possible by the use of a parser.

After the parser assigns the part-of-speech to all query terms, weights are assigned to terms using the following formula:

$$w(t) = \log \frac{f_{ps}}{f_{t,ps}}$$

The weight of a term t with part-of-speech ps is calculated using the logarithm of the division of the overall occurrence frequency of the part-of-speech to which t belongs by the part-of-speech occurrence frequency of t. The occurrence frequency of terms is stored in a dependency database (will be discussed in Section 3.2.3). For a given term t and its part-of-speech ps, one can query the dependency database for the f_{ps} and $f_{t,ps}$. For example, consider query 203 of the TREC-4 collection:

What is the economic impact of recycling tires?

The word *impact* is analyzed by MINIPAR as a noun in this query. The system determines its weight by querying the dependency database for two values. The first value is the total number of nouns in the dependency database, f_{ps}. The second value is the number of occurrences of the term *impact* as a noun is the dependency database, $f_{t,ps}$. The weight of the term *impact* is calculated by dividing the two values and applying the logarithm function on the result. Using the above formula, terms having a lower occurrence frequency are given higher weights.

There are special terms that do not contribute using their fully calculated weight. The weights of these terms are reduced to lessen their influence on the overall query results. One situation where such a weight reduction is used occurs where terms are modified by wh-words such as *what*, *which*, and *how*. For query 203 presented above, the weight of the term *impact* is reduced because it is modified by *what*. In addition, the weights of terms that modify the term modified by wh-words should also be reduced. As a result, for this example, the weights of the terms *what*, *is*, and *the* would be reduced because they modify the word *impact*. In our experiment, term weights of this type were reduced by half. However, in practice, this operation does not need to be performed since these terms will already be ignored because they tend to be stop words.

That is, not all terms in a sentence will have their weights calculated. Terms that are analyzed as poor discriminators are ignored. There are three possible reasons for a term to be ignored by the query translator:

1. *Ignoring a term based on its part-of-speech.*

   For example the word *the* is considered a poor discriminator because of its part-of-speech (a determinant), and therefore, is *not* considered as part of the resulting query. A term that has one of the part-of-speech types listed in Table 3-2 is ignored.

Table 3-2 ignored part-of-speech types.

Part-of-speech	Description
Of	'of'
Be	a Be form such as 'is', 'are', and 'be'
Det	a determiner such as 'the', 'a', 'an', and 'such'

2. *Ignoring a term based on its grammatical relationship.*

The grammatical relationships among terms can lead us to identify poor discriminators. Intuitively we can ignore terms based on the following relationships.

The *ji* relationship. *ji* is the label for sentential modifiers. In this dependency relationship, a modifier can be a prepositional phrase, an adverbial phrase, or a clause that modifies the sentence. Terms with this kind of relationship tend to act as auxiliary terms that are usually used to enhance the flow of the sentence. An example of a term that would be ignored based on this type of relationship is *combat* in query 210 of the TREC-4 collection. *Combat* has a *ji* relationship to the term *done*.

How widespread is the illegal disposal of medical waste in the U.S. and what is being done to combat this dumping?

The *lex-mod* relationship. *lex-mod* is the label for lexical modifiers. This dependency relationship allows us to identify terms that are significant only in combination. The individual terms can be ignored. A good example of a lexical modifier is *power* in query 204 of the TREC-4 collection.

Where are the nuclear power plants in the U.S. and what has been their rate of production?

The dependency tree for *power plant* of the above query is:

```
(
 …
 (nuclear    - A               < plants    jnab)
 (power      - U               < plants    lex-mod)
 (plants     "power plant" N   < in        subj)
 …
)
```

The query translator ignores the term *power* because it is better used as part of a phrase than on its own. In this query the term *power plant* is treated as one term. The weight of such a term is calculated by dividing the total number of nouns in the dependency database over the occurrence frequency of the term *power plant* as a noun. The logarithm function is then applied on the result of the division.

3. *Ignoring a term based on stop word list.*

   The query translator uses a list of stop words (taken from the SMART IR system) to ignore poor discriminators (see Table 2-1).

3.2.3 Dependency Database

Before any further discussion about the query translator, it is necessary to discuss the dependency database and show how it is useful to our approach.

The dependency database was created by extracting syntactic dependencies between words from a corpus using MINIPAR [Lin 1998]. The dependency database consists of the following:

- A list of every word w found in the parsed corpus and its part-of-speech. Words with very low occurrence frequency (e.g. less than 5 times) through the entire corpus are not added to the database.
- The occurrence frequency of w for each of its part-of-speech types w_{ps} in the corpus.
- A list of every modifier m (in combination with its part-of-speech types m_{ps}) of the term w_{ps}, and the relationship between m_{ps} and w_{ps}. These combinations are referred to as "Dependency Triples".
- The occurrence frequency m_{ps} and w_{ps} for a specific syntactic dependency relationship.
- The occurrence frequency of each part-of-speech type.
- The likelihood ratio of m_{ps} and w_{ps} (will be used in the next section). This is a calculated value representing the likelihood that m_{ps} and w_{ps} will co-occur.

To better describe the dependency database, let us consider an example. Figure 3-4 presents the information available in the dependency database for the word *recycle*.

Figure 3-4 Dependency database output for the word *recycle*. The numbered items 1-6 are explained following the dependency database output.

```
recycle 1184(1)
  V     539(2)
      V:compl:N(3)          475(4)    184(5)
                …
                steel                   3       10.3969
                system                  4       3.125
                tin                     1       4.56525
                tin can                 4       28.6703
                tire                    4(6)    14.2485(7)
                trash                   8       40.8528
                trash can               1       5.18741
                truck                   3       6.80176
                …
      …
      V:conj:V          10      9
                handle                  1       4.44705
                produce                 1       3.1167
                redistribute            1       7.2306
                remove                  2       7.65552
                save                    1       3.55548
                staff                   1       5.70004
      …
      V:subj:N          129     63
                American                4       11.0605
                astronaut               1       3.69434
                bag                     1       3.94629
                burden                  1       4.44226
                business                3       6.57617
                buy                     1       4.40796
                Buying                  1       3.00513
                city                    4       9.68945
                community               3       10.5859
                council member          1       3.46118
                editor                  2       7.63647
                …
      …
…
```

(1) The number 1184 is the total number of dependency relationships that involves the word *recycle* (including both nouns and verbs).

(2) The number 539 is the frequency of the word *recycle* as a verb (part-of-speech type) in the parsed corpus.

(3) The form V:comp1:N means that all the words belonging to this form (e.g. *can* and *tire*) are nouns and that they modify the verb *recycle* with the *comp1* syntactic dependency. In other words, the verb *recycle* takes all the nouns listed under this form as objects.

(4) The number 475 represent the occurrence frequency of all nouns in the parsed corpus that modify the verb *recycle* with the *comp1* syntactic dependency.

(5) The number 184 represent unique words in V:comp1:N.

(6) The number 4 represent the occurrence frequency of the syntactic dependency *comp1* between the verb *recycle* and the noun *tire*. In other words, *tire* was used as the object of *recycle* 4 times.

(7) The number 14.2485 represents the log likelihood ratio [Dunning 1993] for recycle(V):comp1:tire(N). This ratio is determined by:

- The number of times the verb *recycle* takes *tire* as an object (k_1).
- The number of times the verb *recycle* takes any noun as an object (n_1).
- The number of dependency relationships where the head is *recycle* but the modifier is not *tire* or the relationship is not V:comp1:N (k_2).
- The number of dependency relationships that are not V:comp1:N. This is typically a large number (n_2).

The likelihood ratio is written as follows:

$$f(p,k,n) = k \log p + (n-k)\log(1-p)$$

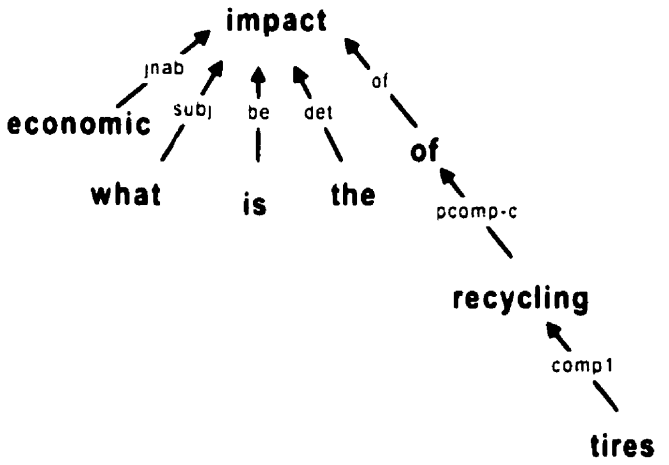
where

$$\log l = f(\frac{k_1}{n_1},k_1,n_1) + f(\frac{k_2}{n_2},k_2,n_2) - f(\frac{k_1+k_2}{n_1+n_2},k_1,n_1) - f(\frac{k_1+k_2}{n_1+n_2},k_2,n_2)$$

3.2.4 Associated Term Weights

Sentences (or queries) consist of a set of terms. These terms have grammatical/syntactic dependencies among each other. As previously mentioned, the dependency tree of a sentence represents all the terms of a sentence and their syntactic dependencies. The dependency tree of query 203 could be represented as illustrated in Figure 3-5.

Figure 3-5 Dependency tree for query 203.



The associated term weight in a query (or tree) is derived from the weight of the term (or node) itself and the links (or edges) that are associated with the term. More precisely, the associated term weight in a tree is calculated by adding the weights of its edges to the weight of the term. In Section 3.2.2, we presented the method of calculating individual term weights. Now we need to determine the weights of dependency links (i.e. the links between words).

A dependency relationship links two words in a sentence. For example, Figure 3-5 shows that the terms *economic* and *impact* are linked with the syntactic dependency relationship *jnab*. What

we are interested to determine here is the weight (or strength) of this relationship. The higher the weight (i.e., the greater the strength) of the relationship, the higher associated weight of the term.

The weight of the syntactic dependency relationship between any two terms in the dependency tree of a query is derived from the information stored in the dependency database. Given any two terms, their part-of-speech, and the syntactic dependency between them, we can calculate the weight of the relationship between them using the following formula:

$$w = \frac{r}{f}$$

where:

f represents the frequency occurrence of m_p, and w_p, for a specific syntactic dependency relationship (i.e., dependency triple)

r represent the likelihood ratio of the dependency triple

In the example presented in Figures 3-4 and 3-5, the syntactic dependency relationship between *recycling* and *tires* for the phrase *recycling tires* has a weight of 3.5621 (calculated: 14.2485 / 4).

The associated weight of terms in query 203 is computed using the following steps:

1. Compute the weight of the individual terms:

Term	Weight
impact	3.64555 (*log* 1.54614e+07/3497)
what	0
the	0
economic	2.50102 (*log* 3.82396e+06/12064)
of	0
recycling	3.92807 (*log* 4.56733e+06/539)
tires	4.23597 (*log* 1.54614e+07/898)

2. Compute the weight of the links in the tree:

   Link1: economic(N):jnab:impact(A) → w = 1.1839

   Link2: tires(N):comp1:recycling(V) → w = 3.5621

   Link3: impact(N):pcomp-c:recycling(N) → w = 0 (no entry available in the dependency database)

3. Compute the associated term weights:

   w(impact) = 1.1839 + 3.64555

   w(*economic*) = 1.1839 + 2.50102

   w(*recycling*) = 3.5621 + 3.92807

   w(*tires*) = 3.5621 + 4.23597

As shown above, the weights of links that have weak syntactic dependencies such as *of* and *be* were not computed and they are assumed to be zero. This rule was followed with one exception. The exception is related to the *of* syntactic dependency. All links of the form N - of - N were treated as N - N. As a result, in the above example, we compute the weight of the link between *recycle* and *impact* and bypass the *of* term. This makes our system treat terms like *retrieval of information* and *encyclopedia of childbirth* as *information retrieval* and *childbirth encyclopedia*, respectively.

3.2.5 Tree Decomposition and Fuzzy Boolean Query

There are several possible query forms that can be fed into an IR system. The simplest form requires feeding all the terms in a query to an IR system without giving any consideration to the contribution of individual terms to the query meaning. In this case, the IR system uses one (or more) of the techniques presented in Chapter 2 (say *tf* idf*) to retrieve and rank the documents that match a given query. This is very similar to using the AltaVista Internet search engine. Using this form, the input to an IR system for query 203 would be:

economic impact recycling tires

A better formulated query is achieved when weights are attached to the query terms. The weight assignment is based on the contribution of the individual terms to the meaning of the query. A possible method for deriving individual query term weights is shown in Section 3.2.2. Using this form, the input to an IR system for query 203 would be:

economic: 3.64555 impact: 2.50102 recycling: 3.92807 tires: 4.23597

In this approach, the explicit weight of a term is multiplied by the weight of that term as determined by the IR system. Therefore, the explicit weights result in the matching documents being ranked differently than without applying weights. For example, if there are two terms in a query that are assigned the same weight by the IR system, the term with the higher explicit

weight will be ranked higher and, therefore, the documents that contain that term will be displayed before the documents that contain the other term.

An even better query form is the weighted fuzzy Boolean query form. The Boolean model gives high performance in terms of precision and recall if the query is well formulated [Frakes and Baeza-Yates 1992]. The fuzzy Boolean model solves the limitation of the standard Boolean model. For a query "A and B and C and D and E" the standard Boolean model requires that the matching documents contain all the terms of the query. A document that contains all but one of the above terms will not be retrieved in response to this query. It is likely, however, that the user would be interested in such a document, and as a result it should be retrieved. Similarly, for a query A or B or C or D or E the standard Boolean model would consider a document that contains one of these terms to be just as important as a document containing more or all of them.

The fuzzy Boolean model recognizes that the most relevant documents to a query are the documents that contain all the terms of the query. At the same time, it also recognizes that documents that contain only a subset of the terms of the query, although less relevant, may still be relevant. This approach requires the determination of all the possible term subsets for a given query.

Determining all the possible subsets of terms can be achieved by decomposing the query dependency tree into smaller dependency trees. Each of the resulting dependency trees is considered a query on its own. More precisely, the query dependency tree is decomposed into all

possible sub-trees by breaking the links between nodes in a recursive manner. To eliminate undesired sub-trees, the breakage of links cannot occur if a link is of one of the following types:

- Lexical modifier: the link breakage is not performed if it is a lexical modifier. The two nodes that are associated with this relationship will be treated as one node, resulting in improved retrieval performance. An example of this is found in query 204. Treating the term *power plant* as one term is much more effective than separating the term into sub-trees that contain *plant* and others containing *power*.
- Proposition, determinant, and verb-to-be: examples of this type are *of*, *the*, and *is*, respectively. Having two identical sub-trees but one of them containing an extra node of type proposition, determinant, or verb-to-be is redundant because a node of this type does not contribute to retrieval performance.
- *ji*: as discussed earlier in Section 3.2.2, a node of this type is ignored and, therefore, creating extra sub-trees that contains this node is not necessary.

Each of the dependency trees in Figure 3-6 is treated as a separate query. The terms of these queries are AND-ed together to form a Boolean expression. Consequently, the resulting Boolean expressions are Or-ed together to form the final fuzzy Boolean query. More formally, a fuzzy Boolean query is generated by decomposing a natural language query into all possible dependency sub-trees. Terms in each sub-tree are treated conjuctively, and sub-trees are treated disjunctively. The term weights are calculated according to their frequency and their association with other links in the sub-tree.

Figure 3-6 Decomposition of dependency tree for query 203.

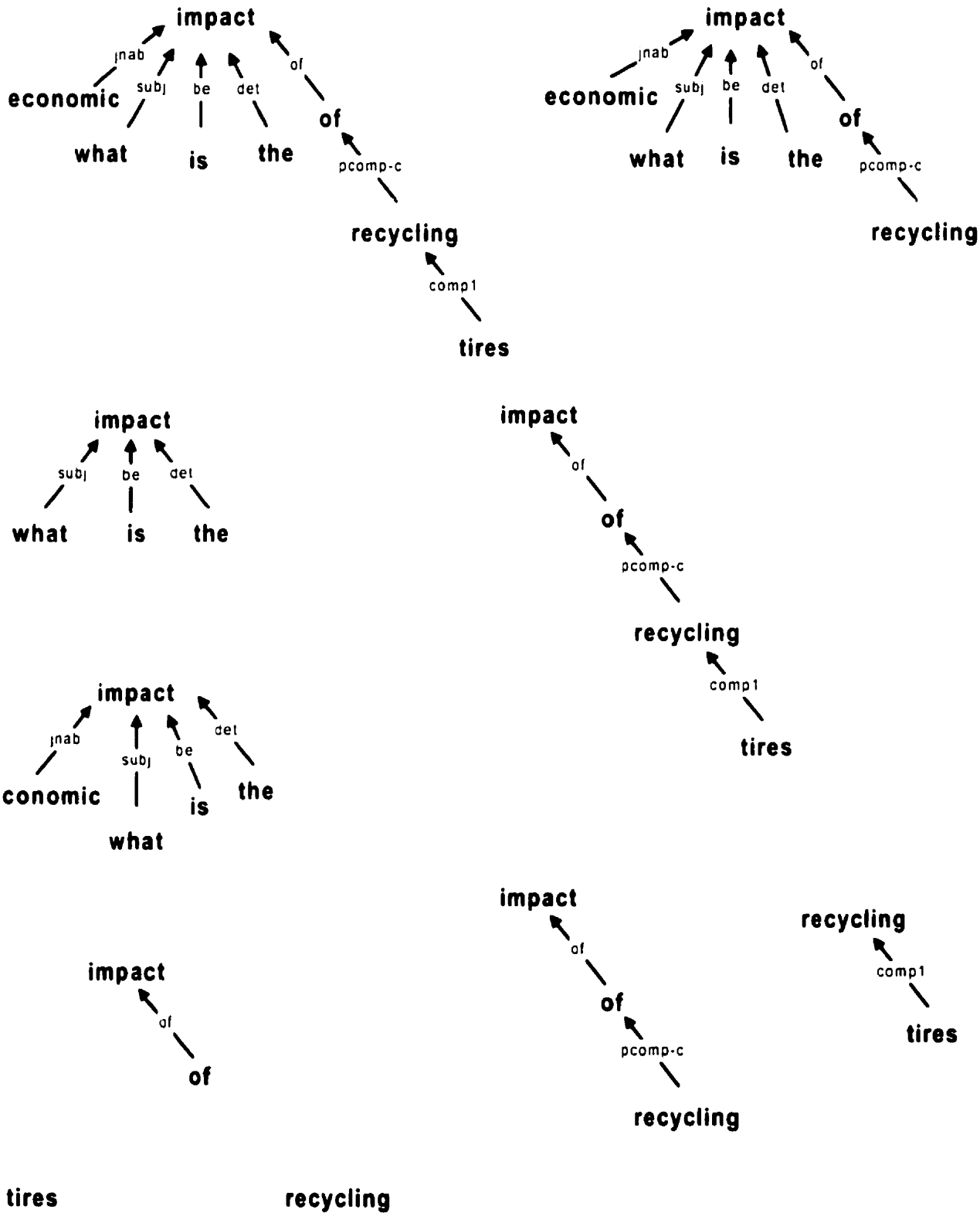


Figure 3-7 shows the weighted Boolean expressions of the dependency sub-trees for query 203 and Figure 3-8 shows the final weighted fuzzy Boolean expression for the same query. The final expression (Figure 3-8) can be fed directly to an IR system.

Figure 3-7 Weighted Boolean expressions of the dependency sub-trees for query 203.

```
Main Tree = economic:2.434439 AND impact:3.006705 AND (tire:7.798107 OR
            tires:7.798107) AND (recycle:7.490208 OR recycling:7.490208)
Sub-tree1 = impact:1.822777 AND (tire:7.798107 OR tires:7.798107) AND
            (recycle:7.490208 OR recycling:7.490208)
Sub-tree2 = (tire:7.798107 OR tires:7.798107) AND (recycle:7.490208 OR
            recycling:7.490208)
Sub-tree3 = economic:2.434439 AND impact:3.006705 AND (recycle:3.928074 OR
            recycling:3.928074)
Sub-tree4 = impact:1.822777 AND (recycle:3.928074 OR recycling:3.928074)
Sub-tree5 = economic:2.434439 AND impact:3.006705
Sub-tree6 = tire:4.235973 OR tires:4.235973
Sub-tree7 = recycle:3.928074 OR recycling:3.928074
Sub-tree8 = impact:1.822777
Sub-tree9 = economic:1.250511
```

As shown in Figure 3-7, a term and its root are used to broaden the coverage of queries to improve retrieval effectiveness.

Figure 3-8 Final weighted fuzzy Boolean expression for query 203.

```
(economic:2.434439 AND impact:3.006705 AND (tire:7.798107 OR tires:7.798107)
AND (recycle:7.490208 OR recycling:7.490208)
) OR
(impact:1.822777 AND (tire:7.798107 OR tires:7.798107) AND (recycle:7.490208
OR recycling:7.490208)
) OR
(
      (tire:7.798107 OR tires:7.798107) AND
      (recycle:7.490208 OR recycling:7.490208)
) OR
(economic:2.434439 AND impact:3.006705) AND
(recycle:3.928074 OR recycling:3.928074) OR
(impact:1.822777 AND (recycle:3.928074 OR recycling:3.928074)
) OR
(economic:2.434439 AND impact:3.006705) OR
(tire:4.235973 OR tires:4.235973) OR
(recycle:3.928074 OR recycling:3.928074) OR
impact:1.822777 OR
economic:1.250511
```

3.3. Query Expansion

So far we have been discussing query translation, the process of converting a natural language query into a Boolean query. The goal of the query translator is to improve retrieval performance. This is achieved by assigning high weights to good discriminators and generating fuzzy Boolean queries. Although the improvement in retrieval effectiveness using the query translator is significant, there is an upper bound limit to the degree of improvement. This is due to the well known information retrieval problem with word mismatching. The vocabulary used in user queries often differs from vocabulary used by authors of relevant documents. The goal of the query expander component is to further improve retrieval performance by introducing new terms to user queries that broaden the scope of query concepts.

Our query expander uses WordNet to expand query terms. Although there have been many attempts to use thesauri to improve retrieval performance, researchers have been faced with two main problems:

1. Word sense disambiguation: determining the correct sense or senses for a word is a very difficult task [Voorhees 1994] [Chakravarthy and Haase 1995]. For example, WordNet considers the words *slush*, *ice*, and *cocaine* to be synonyms for the word *snow*. *Cocaine* and *slush* have totally different meanings. Deciding which word to choose for expansion depends of their context in the query. Ignoring word context within queries and expanding all the possible senses (synonyms) of a word degrades retrieval performance significantly. [Chakravarthy and Haase 1995] implemented a discriminator that rejects undesirable word senses from WordNet. The discriminator uses 44 heuristics based on 12 kinds of relationships between words. As an example of a heuristic, consider the phrase *snow and slush*. The relationship in this phrase is conjunction. The discriminator rejects the *cocaine* sense of *snow* because it does not share any similarity with *slush*.
2. The expansion of all terms in a query (including those that are unimportant) also significantly degrades retrieval performance [Voorhees 1994].

Our approach overcomes these problems by expanding only the important terms in the query and disambiguating the undesired senses from WordNet.

3.3.1. Selective Term Expansion

The query expander is selective in what terms to choose for expansion. To consider a term for expansion, it must satisfy the following conditions:

1. The weight of the term must be higher than zero (i.e., be an important term).
2. The term must be involved in a syntactic dependency relationship with another term that has a weight higher than zero.
3. The part-of-speech of the term must be noun (other parts of speech such as adjectives, adverbs, and verbs could be used but nouns tend to be the least ambiguous).

Based on the above conditions, expansion happens to pairs only. Only the noun term(s) out of the pair will be considered for expansion. In addition, if a query contains an important term but is not associated with another important term, then this term will not be expanded. This approach is very restrictive in choosing potential terms for expansion. The algorithm used by the query expander to add terms to a query is as follows:

Input: Weighted Dependency Link
Output: SynHead, SynMod
Processing:

```
for every node A in the dependency tree {
        if A has a weight greater than zero {
                if A is a modifier to a node B that has weight greater than zero {
                        if A is a noun {
                                SynHead = All noun synonyms of A in WordNet
                        }
                        if B is a noun {
                                SynMod = All noun synonyms of B in WordNet
                        }
                }
        }
}
```

The output of the algorithm, SynHead and SynMod, contain the potential nouns that are considered for expansion. These lists of nouns are retrieved from WordNet as synonyms to the pair A and B found in the dependency tree.

Using query 203 as an example, note that there are two pairs that will be considered for expansion. The pair *economic* and *impact* and the pair *recycling* and *tire*. For the first pair, *economic* and *impact*, only the word *impact* is considered for expansion because it is a noun. The query expander retrieves all the noun synonyms from WordNet for the word *impact*. The list of synonyms is:

impact
contact
impinging
issue
striking
consequence
effect
outcome

For this project WordNet was chosen as the thesaurus, however, any other thesaurus could be used in a very similar manner.

3.3.2. Word Sense Disambiguation

Identifying the important terms for expansion and retrieving their synonyms is only one of the important functions of the query expander. Its other important function is to choose the relevant synonyms to the query and use only those synonyms that are believed to broaden the scope of the query without hurting performance. Therefore, the goal here is to select only the synonyms

relevant to the query from SynHead and SynMod. This is accomplished using the following algorithm:

Input: SynHead, SynMod, term A, term B, syntactic dependency S
Output: SynPair
Processing:

```
for every term X in SynMod {
    if there is an entry in the dependency database for A(Aps):S:X(Bps) {
        Add the pair X and A to SynPair
    }
}
for every term X in SynHead {
    if there is an entry in the dependency database for X(Aps):S:B(Bps) {
        Add the pair X and B to SynPair
    }
}
for every term X in SynHead {
    for every term Y in SynMod {
        if there is an entry in the dependency database for
            X(Aps):S:Y(Bps) {
        Add the pair X and Y to SynPair
    }
}
```

Synonyms are assigned weights just as was done for the original query terms. That is, independent term weights are calculated based on their frequency occurrence and link weights are added to the independent weights to come up with the associated term weights. Since terms modified by wh-questions have their weights reduced, their synonym weights are reduced as well.

The sample output (SynPair) for query 203 follows. From the list of eight possible synonyms for *impact* presented above, we have expanded the query to include only three – *consequence*, *effect*, and *issue*. These three terms are correct synonyms for the sense in which the original query used

impact. Our method of disambiguation successfully eliminated bad expansion choices such as *impinging* and *striking*.

SynPair for economic (adjective) : jnab : impact (noun)

economic (adjective) : jnab : consequence (noun)
economic (adjective) : jnab: effect (noun)
economic (adjective) : jnab: issue (noun)

Figure 3-9 shows the expanded weighted Boolean expressions of the dependency sub-trees for query 203 and Figure 3-10 shows the final expanded weighted fuzzy Boolean expression for the same query. The final expression (Figure 3-10) can be fed directly to an IR system.

Figure 3-9 Expanded weighted Boolean expressions of the dependency sub-trees for query 203.

```
Main Tree = ((economic:2.434439 AND impact:3.006705) OR (economic:2.717367 AND
            consequence:3.541575) OR (economic:1.624860 AND effect:2.051445)
            OR (economic:1.394261 AND issue:1.531122)) AND (tire:7.798107 OR
            tires:7.798107) AND (recycle:7.490208 OR recycling:7.490208)
Sub-tree1 = impact:1.822777 AND (tire:7.798107 OR tires:7.798107) AND
            (recycle:7.490208 OR recycling:7.490208)
Sub-tree2 = (tire:7.798107 OR tires:7.798107) AND (recycle:7.490208 OR
            recycling:7.490208)
Sub-tree3 = ((economic:2.434439 AND impact:3.006705) OR (economic:2.717367 AND
            consequence:3.541575) OR (economic:1.624860 AND effect:2.051445)
            OR (economic:1.394261 AND issue:1.531122)) AND (recycle:3.928074
            OR recycling:3.928074)
Sub-tree4 = impact:1.822777 AND (recycle:3.928074 OR recycling:3.928074)
Sub-tree5 = (economic:2.434439 AND impact:3.006705) OR (economic:2.717367 AND
            consequence:3.541575)  OR  (economic:1.624860 AND effect:2.051445)
            OR (economic:1.394261 AND issue:1.531122)
Sub-tree6 = tire:4.235973 OR tires:4.235973
Sub-tree7 = recycle:3.928074 OR recycling:3.928074
Sub-tree8 = impact:1.822777
Sub-tree9 = economic:1.250511
```

Figure 3-10 Final expanded weighted fuzzy Boolean expression for query 203.

```
(
      (
            (economic:2.434439 AND impact:3.006705) OR
            (economic:2.717367 AND consequence:3.541575) OR
            (economic:1.624860 AND effect:2.051445) OR
            (economic:1.394261 AND issue:1.531122)
      ) AND
      (tire:7.798107 OR tires:7.798107) AND
      (recycle:7.490208 OR recycling:7.490208)
) OR
(
      (impact:1.822777 AND (tire:7.798107 OR tires:7.798107) AND
      (recycle:7.490208 OR recycling:7.490208)
) OR
(
      (tire:7.798107 OR tires:7.798107) AND
      (recycle:7.490208 OR recycling:7.490208)
) OR
(
      (
            (economic:2.434439 AND impact:3.006705) OR
            (economic:2.717367 AND consequence:3.541575) OR
            (economic:1.624860 AND effect:2.051445) OR
            (economic:1.394261 AND issue:1.531122)
      ) AND
      (recycle:3.928074 OR recycling:3.928074)
) OR
(impact:1.822777 AND (recycle:3.928074 OR recycling:3.928074)
) OR
(
      (economic:2.434439 AND impact:3.006705) OR
      (economic:2.717367 AND consequence:3.541575) OR
      (economic:1.624860 AND effect:2.051445) OR
      (economic:1.394261 AND issue:1.531122)
) OR
(tire:4.235973OR tires:4.235973) OR
(recycle:3.928074 OR recycling:3.928074) OR
impact:1.822777 OR
economic:1.250511
```

CHAPTER 4 – EXPERIMENTS AND RESULTS

4.1. Introduction

In the previous chapter, we presented our approach for improving information retrieval performance. In this chapter we will present the experiments we have performed and the results of those experiments. We will begin by describing the experimental environment and then present the data gathered from the experiments.

4.2. The Environment

The environment is comprised of software components and a test collection hosted on a Sun Solaris workstation. A parser, a query translator, a query expander, a thesaurus, and a search engine constitute the software component portion of the environment. The test collection of choice was the TREC collection (as described in Chapter 3).

4.2.1. The Parser

The parser used in these experiments was MINIPAR [Lin, 1995]. It was developed by Dr. Dekang Lin at the University of Manitoba. MINIPAR was chosen as the parser for this work for the following reasons:

- It is a broad coverage parser. The test collection contains a very broad range of topics that span many domains.
- It produces a dependency tree structure in addition to a constituency tree structure.
- It achieves high accuracy. The dependency relationships can be correctly identified for approximately 85% of words [Lin, 1995].

- It is implemented entirely in the C++ programming language. This makes it easier to integrate with the other software components of this system.

4.2.2. The Search Engine

The search engine used in these experiments was Isearch [Nassar, 1997]. Isearch was chosen as the search engine for this work for the following reasons:

- It supports full text searching.
- It supports Boolean and weighted queries.
- It is implemented entirely in the C++ programming language. This makes it easier to integrate with the other software components of this system
- It is freely available (including source code).

4.2.3. The Thesaurus

The thesaurus used in these experiments was WordNet [Miller, 1990]. WordNet was chosen as a lexical database for this work for the following reasons:

- It is very rich in content.
- It is used by many Information Retrieval and Natural Language Processing researchers.
- It is implemented entirely in the C programming language. This makes it easier to integrate with the other software components of this system.
- It is freely available (including source code).

4.2.4. The Query Translator and Expander

The query translator and expander were developed as part of this work. They are the implementation of the concepts and algorithms presented in Chapter 3. They were written using the C++ programming language. This made it easier to integrate with MINIPAR, Isearch, and WordNet.

4.2.5. The Test Collection

The test collection used in these experiments was the TREC-4 collection [Harman 1996]. TREC-4 was chosen for its short-form queries. It was necessary to reduce the size of the collection because of limited disk space and processing power (the TREC collection is approximately 2 GB of text). As a result we used only the Associated Press documents of TREC-4 (82 MB).

4.3. Experiments

We ran the query translator and the query expander on the collection and calculated the recall and precision for each query. To fairly measure the effectiveness of our method, we wanted to choose a lower bound that we could compare our results against. We called this lower bound the *baseline*.

The *baseline* results were achieved by passing the TREC queries "as is" to the search engine and computing the recall and precision of each query. The query translator, on the other hand, first converts the TREC query into a Boolean expression and weights the terms before passing it to

the search engine. This conversion and weighting is also applied when the query expander is used.

Our goal is to improve recall and precision in general and in the top-ranked documents of result sets in particular. To facilitate measurement of how well we are meeting this goal, we calculated recall and precision at different document cut-off points. The cut-off determines the number of top-ranked documents to be considered. That is, if the cut-off value is set at n only the top n documents retrieved for each query will be used in calculating recall and precision for that query. Intuitively we expect that the recall will increase as the cut-off increases and that precision will decrease as the cut-off increases. Our results, as presented in the next section, are consistent with this expectation.

The cut-offs we used were 50, 100, and 1000 documents. We selected 50 as the lowest cut-off because we believe it is the maximum number of documents we could reasonably expect a user to review. We selected 1000 as the highest cut-off because we believe that it is a reasonable number of documents upon which to apply further processing in future expansions to this work. For example, we can analyse the top 1000 documents to determine new term weights based on their occurrence and syntactic dependencies in these documents. As a result we get a new set of top-ranked documents. We selected the cut-off of 100 arbitrarily.

4.4. Results

To measure retrieval performance, we computed precision and recall for each query at each cut-off under each of the three query methods (baseline, query translator, and query expander). We also calculated the average precision and recall for each cut-off / query method combination.

4.4.1. Recall

The results of our experiments show that overall performance of the query translator in terms of recall is better than the baseline. Also, the query expander achieved better overall results as compared to the query translator. Table 4-1 and Figure 4-1 present the recall results at each of the selected cut-offs. Appendix B shows the recall results for each query.

Table 4-1 Recall Results Summary.

Cut-off	Baseline	Query Translator (QT)	Query Expander (QE)	QT - Baseline	QE - Baseline	QE - QT
50	25.4653%	28.6726%	29.3531%	3.2073%	3.8878%	0.6805%
100	33.3590%	40.5374%	40.6504%	7.1785%	7.2914%	0.1129%
1000	69.6548%	73.8043%	74.3275%	4.1495%	4.6726%	0.5231%

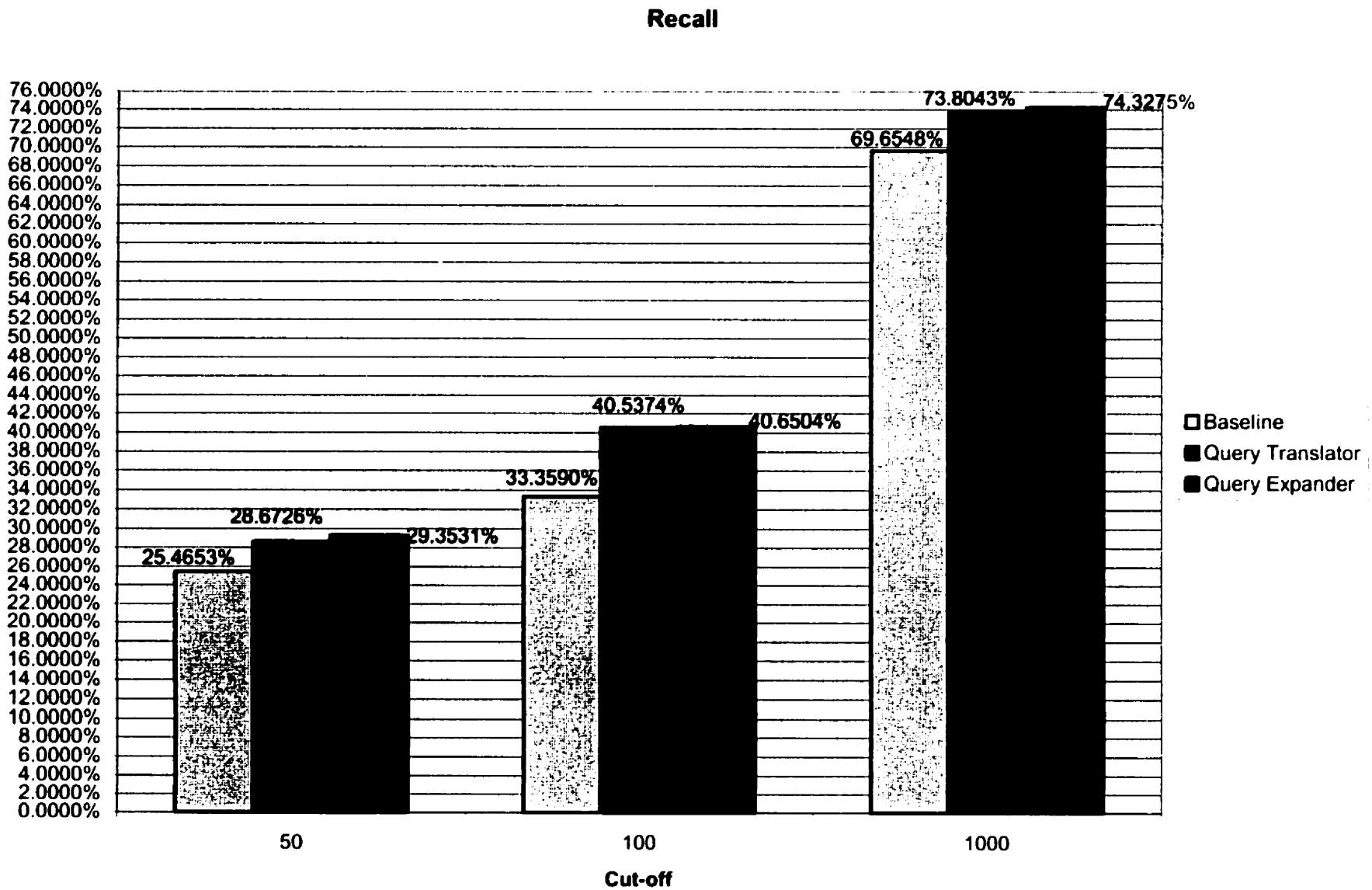


Figure 4-1 Recall Results Chart.

4.4.2. Precision

Table 4-2 and Figure 4-2 show the precision results. Appendix C shows the precision results for each query. Overall, in terms of precision, the query translator achieved better results than the straightforward baseline results at all cut-offs. Also, the query expander achieved better overall results as compared to the query translator

Table 4-2 Precision Results Summary.

Cut-off	Baseline	Query Translator (QT)	Query Expander (QE)	QT - Baseline	QE - Baseline	QE - QT
50	17.0000%	20.2500%	20.5417%	3.2500%	3.5417%	0.2917%
100	12.9375%	15.5625%	15.5833%	2.6250%	2.6458%	0.0208%
1000	3.8042%	4.1134%	4.1634%	0.3092%	0.3592%	0.0500%

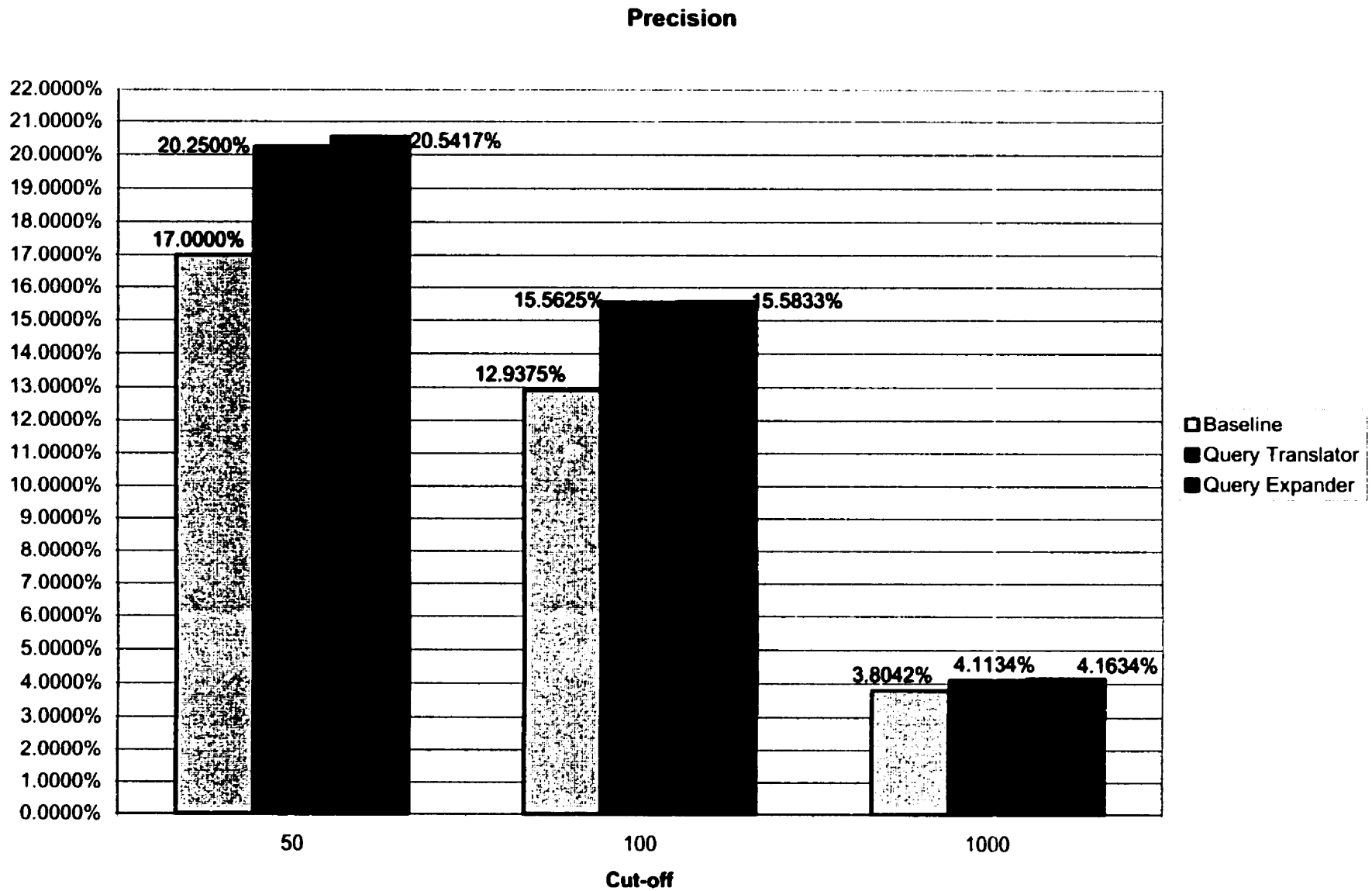


Figure 4-2 Precision Results Chart.

4.4.3. Statistical Analysis of the Results

Taking a query q as a random variable, it is possible to use statistical inference to draw conclusions about a population of queries based on a sample of the results of n queries. For our study we have a sample of 48 queries (i.e., n=48) – see Appendix A. As previously described, we have calculated the recall (Appendix B) and precision (Appendix C) for each of these 48 queries under baseline, query translator, and query expander query formulations for cut-offs of 50, 100, 1000. Descriptive statistics for each scenario are presented in tables 4-3a, 4-4a, 4-5a, 4-6a, 4-7a, and 4-8a. To simplify this analysis, we have assumed a standard normal distribution (possible due to the central limit theorem).

To test the hypothesis that our query translator and query expander improved results over the baseline, we conducted paired difference tests. The respective null hypothesis were:

1. There is no difference between the baseline mean and the QT mean.
2. There is no difference between the baseline mean and the QE mean.
3. There is no difference between the QT mean and the QE mean.

The alternative hypothesis were:

1. Mean QT > mean baseline.
2. Mean QE > mean baseline.
3. Mean QE > mean QT.

Test statistics for each scenario are presented in tables 4-3b, 4-4b, 4-5b, 4-6b, 4-7b, and 4-8b.

The calculated values of the test statistic t support the rejection of null hypotheses #1 and #2 at a 90% level of confidence. We were not able to reject the third null hypothesis at this level of confidence.

We have concluded that we can achieve even better results if we reduce the variability in the sample. Increasing the sample size (i.e., the number of queries) might help. Additionally, there are other factors contributing to the high variability that we will address in our future work. Specifically, the results were obtained by running the queries against a sub-set of the TREC-4 documents. With a full document set, we expect that we would have fewer extreme results (e.g., 100% recall for query 220 and 0% recall for query 233). This expectation is supported in [Vlez, Weiss, Sheldon, and Gifford 1997] where a significant improvement was seen going from an 8k document set to an 84k document set. We believe that we did not see a statistically significant improvement using QE over QT because it is highly restrictive (i.e. very few queries actually met the criteria for expansion).

To provide an indication of at what level of confidence we would be able to reject the null hypothesis (in those cases where we were unable to at the 90% level) we have also calculated P-values.

Table 4-3a Recall at cut-off 50 – descriptive statistics.

Desc. Statistics	Baseline	QT[1)]	QE[2)]
Mean	25.4653%	28.6726%	29.3531%
Standard Deviation	0.241012925	0.245677787	0.259229803
Variance	0.058087230	0.060357575	0.067200091

Table 4-3b Recall at cut-off 50 – test statistics.

Test statistic factors	QT v. Baseline	QE v. Baseline	QE v. QT
Observations	48	48	48
t stat	1.781009730	2.030010189	1.162543544
P-value	0.040687057	0.024018540	0.125441921
t Critical one-tail	1.299824817	1.299824817	1.299824817

Table 4-4a Recall at cut-off 100 – descriptive statistics.

Desc. Statistics	Baseline	QT[1)]	QE[2)]
Mean	33.3590%	40.5374%	40.6504%
Standard Deviation	0.260232660	0.281671507	0.286068892
Variance	0.067721037	0.079338838	0.081835411

Table 4-4b Recall at cut-off 100 – test statistics.

Test statistic factors	QT v. Baseline	QE v. Baseline	QE v. QT
Observations	48	48	48
t stat	4.265001882	4.145505232	0.352558074
P-value	0	0	0.362997964
t Critical one-tail	1.299824817	1.299824817	1.299824817

Table 4-5a Recall at cut-off 1000 – descriptive statistics.

Desc. Statistics	Baseline	QT[1)]	QE[2)]
Mean	69.6548%	73.8043%	74.3275%
Standard Deviation	0.245688674	0.249638477	0.251331809
Variance	0.060362924	0.062319369	0.063167678

Table 4-5b Recall at cut-off 1000 – test statistics.

Test statistic factors	QT v. Baseline	QE v. Baseline	QE v. QT
Observations	48	48	48
t stat	2.514063206	2.886855875	1.247272314
P-value	0.007708968	0.002931145	0.109238562
t Critical one-tail	1.299824817	1.299824817	1.299824817

Table 4-6a Precision at cut-off 50 – descriptive statistics.

Desc. Statistic	Baseline	QT[1)]	QE[2)]
Mean	17.0000%	20.2500%	20.5417%
Standard Deviation	0.138564065	0.171482894	0.177199951
Variance	0.019200000	0.029406383	0.031399823

Table 4-6b Precision at cut-off 50 – test statistics.

Test statistics factors	QT v. Baseline	QE v. Baseline	QE v. QT
Observations	48	48	48
t stat	2.424736817	2.454282650	0.942786299
P-value	0.009608662	0.008938021	0.175305685
t Critical one-tail	1.299824817	1.299824817	1.299824817

Table 4-7a Precision at cut-off 100 – descriptive statistics.

Desc. Statistic	Baseline	QT[1)]	QE[2)]
Mean	12.9375%	15.5625%	15.5833%
Standard Deviation	0.120837613	0.124574275	0.128192563
Variance	0.014601729	0.015518750	0.016433333

Table 4-7b Precision at cut-off 100 – test statistics.

Test statistics factors	QT v. Baseline	QE v. Baseline	QE v. QT
Observations	48	48	48
t stat	2.673278877	2.528118865	0.093930705
P-value	0.005147796	0.007443239	0.462781749
t Critical one-tail	1.299824817	1.299824817	1.299824817

Table 4-8a Precision at cut-off 1000 – descriptive statistics.

Desc. Statistic	Baseline	QT[1)]	QE[2)]
Mean	3.8042%	4.1134%	4.1634%
Standard Deviation	0.040692912	0.041283192	0.041753521
Variance	0.001655913	0.001704302	0.001743356

Table 4-8b Precision at cut-off 1000 – test statistics.

Test statistics factors	QT v. Baseline	QE v. Baseline	QE v. QT
Observations	48	48	48
t stat	2.725098570	3.283339684	1.273063099
P-value	0.004500275	0.000970448	0.104628365
t Critical one-tail	1.299824817	1.299824817	1.299824817

1) QT = Query Translator
2) QE = Query Expander

4.4.4. Data Analysis of the Results

The query translator was generally able to improve retrieval effectiveness. In some cases, it had no effect on the outcome. In other cases, query translator decreased the retrieval effectiveness. During analysis of the data, we made three observations worth mentioning:

1. *No change in results but underlying retrieval differences.*

   Although some queries seem to show no improvement using the query translator compared to the baseline, there are actually differences in the documents retrieved. For example, as shown in Appendices B and C, the performance of query 203 does not show any improvements between the query translator and the baseline for cut-off 50. However, the relevant documents retrieved by the query translator for query 203 are actually different than the relevant documents retrieved using the baseline.

   One of the documents retrieved using the baseline is a document about the hopes of the government of Malaysia to increase the amount of fish in its waters by purchasing millions of used tires from Japan and using them to form artificial reefs off its coast. This document is considered relevant but was retrieved by the baseline only because it contains the word *tire*. Using this method, the document could just as easily have been about tire safety.

   This document was not part of the result set at cut-off 50 using the query translator. The query translator retrieved another document on recycling tires that was not included in the baseline, making the overall results the same. All the documents retrieved by the query translator at cut-off 50 were about recycling of tires.

2. *The lex-mod relationship (described on section 3.2.2) sometimes decreases performance.*

For example, the terms *success story*, *automobile industry*, and *private corporation* appeared in queries 228, 230, and 243, respectively, caused the query translator performance to degrade at cut-off 50 compared to the baseline.

3. *Long and well-formed queries tend to perform slightly better using the baseline.*

   This is because of the fuzzy weighted Boolean queries in the query translator. The weight of some terms that are not very important semantically could have a higher combined weight than a single term that is more important to query's semantics. For example, query 240 generates expressions such as *assist* AND *combat*, *equipment* AND *advance*, and *assist* AND *combat* that are not individually important to the premise of the query.

When we compared the query expander to the query translator we made three observations based on our analysis of the data:

1. *Not all the generated expansion terms are necessarily good candidates for improving retrieval performance.*

   For example, terms such as *station* and *power station* were generated as expansion terms for the term *power plant* in query 204. The term *power station* is a good candidate but the term *station* is not.

2. *Some generated expansion terms appear useful but decrease performance anyway.*

   Query 219 is a good example. For the term *auto*, the generated expansion terms were *automobile*, *car*, and *motor vehicle*. Although all of these are good expansion terms they

worsen the retrieval performance. Another example is query 222. The term *death penalty* was generated as an expansion term for the term *capital punishment*. We would expect performance to be improved but it was not.

3. *A better interpretation of the dependency tree is needed to form more accurate dependencies between words.*

   For example, based on our interpretation of the dependency tree of query 208, terms such as *product* and *fertilizer* are engaged in a syntactic dependency. As a result, the term *product* and the term *fertilizer* were AND-ed together while they were suppose to be OR-ed in addition to the term *garbage*. Furthermore, the term *chemical* was generated as an expansion term for the syntactic dependency between *product* and *fertilizer* that caused the query performance to degrade.

CHAPTER 5 – CONCLUSION AND FUTURE WORK

Information filtering and retrieval continues to be a very important topic in our daily lives. Information retrieval systems enable us to filter relevant documents and ignore those that are irrelevant. As part of the filtering process, the relevant documents are sorted based on their closeness to users' information needs.

For more than three decades, researchers and practitioners have been working on defining effective and efficient algorithms and tools to tackle the challenges of this discipline. Most of the work done in this area is statistical in nature. It is more or less based on the occurrence frequency of words in documents.

In this thesis, we presented an overview of information retrieval system theory and provided a review of past and current literature related to query formulation and expansion. We then introduced a new method for query formulation and expansion. This method depends heavily on natural language processing. Natural language queries were converted to Boolean queries and assigned weights based on their syntactic dependency. We evaluated our work using the well-known TREC collection. The results support and validate the effectiveness of this method for improving retrieval performance.

Since the results are promising, future work seems warranted. First, we want to experiment with multi-sentence queries. As stated in Section 3. 2, the query translator developed for this thesis handles only single sentence natural language queries. As a result, if a query contains more than

one sentence, only the first sentence will be processed. In our future work we intend to expand the translator to handle multi-sentence queries.

Second, we plan to introduce the proximity operator, NEAR, to the syntax of the Boolean query. The NEAR operator will be more effective in some situations than the AND operator. The proximity operator enforces two terms not only to occur in the same document but also to be within a short distance from each other such as within the same sentence or within *n* words.

Third, the parsing techniques still need some improvement. One such improvement is the handling of conjunction relationships. For example, consider the phrase *A of B and C*. It is very difficult for the parser to identify the participants in the conjunction. The parser has to make a choice between the following possibilities:

- (A of B) and C
- A of (B and C)

The parser does not always come up with the right answer. This affects the structure of the query dependency structure and, as a result, the effectiveness of the Boolean query suffers.

Fourth, we intend to test or method on a larger collection. We are planning to use more of the TREC collections and their associated queries.

Finally, we would like to add additional processing to analyse the top *n* (perhaps 1000) documents to determine new term weights based on their occurrence and syntactic dependencies in these documents and re-weight the query to get a revised set of top-ranked documents.

APPENDIX A – TREC-4 QUERIES

List of the 202-250 TREC-4 topics (queries):

202: Status of nuclear proliferation treaties -- violations and monitoring
203: What is the economic impact of recycling tires?
204: Where are the nuclear power plants in the U.S. and what has been their rate of production?
205: What evidence is there of paramilitary activity in the U.S.?
206: Prognosis or viability of a political third party in U.S.
207: What are the prospects of the Quebec separatists achieving independence from the rest of Canada?
208: What are the late developments in bioconversion -- the conversion of biological waste, garbage, and plant material into energy, fertilizer, and other useful products?
209: Identify what is being done or what ideas are being proposed to ensure that Social Security will not go broke.
210: How widespread is the illegal disposal of medical waste in the U.S. and what is being done to combat this dumping?
211: How effective are the driving while intoxicated (DWI) regulations.
212: What countries have been accused of failing to adequately protect U.S. copyrights, patents and trademarks.
213: As a result of DNA testing, are more defendants being absolved or convicted of crimes?
214: What are the different techniques used to create self-induced hypnosis?
215: Why is the infant mortality rate in the United States higher than it is in most other industrialized nations?
216: What research is ongoing to reduce the effects of osteoporosis in existing patients as well as prevent the disease occurring in those unafflicted at this time?
217: Reporting on possibility of and search for extra-terrestrial life or intelligence.
218: How have mini steel mills changed the steel industry?
219: How has the volume of U.S. imports of Japanese autos compared with export of U.S. autos to Canada and Mexico?
220: How do crossword puzzle makers go about making their puzzles?
221: Steps taken by church, governments, community, civic organizations to halt carnage among youths engaged in drug or gang warfare.
222: Is there data available to suggest that capital punishment is a deterrent to crime?
223: What was responsible for the great emergence of "MICROSOFT" in the computer industry?
224: What can be done to lower blood pressure for people diagnosed with high blood pressure?
225: What is the main function of the Federal Emergency Management Agency (FEMA) and the funding level provided to meet emergencies?
226: Have large scale state allowed lotteries and gambling improved the state's financial conditions?
227: Identify instances and reasons of deaths in the U.S. military caused by other than enemy (e.g., friendly fire, training accidents).

228: What are some of the biggest success stories in recent years concerning environmental recovery from pollution?
229: What is being done to help people suffering from schizophrenia?
230: Is the automobile industry making an honest effort to develop and produce an electric-powered automobile?
231: Should the U.S. Government provide increased support to the National Endowment for the Arts?
232: Reports of and evaluation on the near-death experience.
233: What are the late developments of co generation for electrical power?
234: What progress has been made in fuel cell technology?
235: What support is there in the U.S. for legalizing drugs?
236: Are current laws of the sea uniform?
237: Identify alternative sources of energy for automobiles.
238: What is the function, annual budget, and cost involved with the management and upkeep of the U.S. National Parks?
239: Are there certain regions in the United States where specific cancers seem to be concentrated?
240: What controls, agreements, technological advances or equipment are now in use or planned to assist in combating terrorism?
241: Find examples of doctor and lawyer groups considering penalties against members of their professions for malfeasance and show the results of such investigations
242: How has affirmative action affected the construction industry?
243: Should the government restrict the use of fossil fuels as energy resources by private corporations and public utilities?
244: Status of trade balance with Japan -- deficit problem.
245: What are the trends and developments in retirement communities?
246: What is the extent of U.S. arms exports?
247: What are the predictions concerning the economic effects of the U.K.-Continent underwater tunnel (Chunnel).
248: What are some developments in electronic technology being applied to and resulting in advances for the blind?
249: How has the depletion or destruction of the rain forest effected the worlds weather?
250: Does available data show a positive correlation between the sales of firearms and ammunition in the U.S. and the commission of crimes involving firearms?

APPENDIX B – RECALL PER QUERY

Query	50			100			1000		
	Baseline	QT[1)]	QE[2)]	Baseline	QT[1)]	QE[2)]	Baseline	QT[1)]	QE[2)]
202	17.1717%	32.3232%	32.3232%	36.3636%	46.4646%	46.4646%	95.9596%	95.9596%	95.9596%
203	50.0000%	50.0000%	50.0000%	50.0000%	80.0000%	80.0000%	100.0000%	100.0000%	100.0000%
204	3.4965%	9.0909%	8.3916%	6.9930%	16.7832%	16.0839%	71.3287%	78.3217%	80.4196%
205	0.4098%	0.4098%	0.4098%	1.2295%	1.2295%	1.2295%	9.0164%	11.0656%	11.0656%
206	0.0000%	37.9310%	37.9310%	6.8966%	55.1724%	55.1724%	75.8621%	68.9655%	68.9655%
207	45.1613%	41.9355%	41.9355%	67.7419%	74.1936%	74.1936%	96.7742%	100.0000%	100.0000%
208	29.1667%	29.1667%	25.0000%	41.6667%	41.6667%	37.5000%	75.0000%	95.8333%	95.8333%
209	27.5862%	20.6897%	20.6897%	32.7586%	41.3793%	41.3793%	75.8621%	87.9310%	87.9310%
210	34.0000%	42.0000%	42.0000%	42.0000%	46.0000%	46.0000%	82.0000%	100.0000%	100.0000%
211	8.9474%	10.0000%	10.0000%	21.0526%	13.6842%	13.6842%	72.6316%	72.6316%	72.6316%
212	35.2941%	29.4118%	29.4118%	41.1765%	50.0000%	50.0000%	61.7647%	97.0588%	97.0588%
213	36.3636%	36.3636%	36.3636%	45.4545%	45.4545%	45.4545%	90.9091%	81.8182%	81.8182%
214	75.0000%	75.0000%	75.0000%	75.0000%	100.0000%	100.0000%	100.0000%	100.0000%	100.0000%
215	21.0526%	27.3684%	30.5263%	35.7895%	46.3158%	53.6842%	85.2632%	71.5789%	90.5263%
217	18.4211%	34.2105%	34.2105%	26.3158%	50.0000%	50.0000%	52.6316%	65.7895%	65.7895%
218	75.0000%	75.0000%	100.0000%	100.0000%	100.0000%	100.0000%	100.0000%	100.0000%	100.0000%
219	34.2105%	39.4737%	36.8421%	47.3684%	57.8947%	63.1579%	86.8421%	94.7368%	97.3684%
220	100.0000%	100.0000%	100.0000%	100.0000%	100.0000%	100.0000%	100.0000%	100.0000%	100.0000%
221	8.3333%	15.4762%	14.2857%	14.2857%	20.2381%	19.0476%	79.7619%	77.3810%	82.1429%
222	23.6364%	40.0000%	50.9091%	34.5455%	54.5455%	58.1818%	80.0000%	94.5455%	94.5455%
223	87.5000%	100.0000%	100.0000%	100.0000%	100.0000%	100.0000%	100.0000%	100.0000%	100.0000%
224	32.0000%	46.0000%	46.0000%	44.0000%	56.0000%	56.0000%	92.0000%	100.0000%	100.0000%
225	25.6757%	21.6216%	20.2703%	40.5405%	37.8378%	28.3784%	82.4324%	75.6757%	71.6216%
226	25.6410%	14.1026%	14.1026%	33.3333%	35.8974%	35.8974%	74.3590%	98.7179%	98.7179%
227	5.6277%	6.4935%	6.4935%	6.9264%	10.8225%	10.8225%	38.0952%	50.2164%	50.2164%
228	16.1290%	3.2258%	3.2258%	29.0323%	6.4516%	6.4516%	61.2903%	67.7419%	67.7419%
229	61.5385%	53.8462%	53.8462%	61.5385%	69.2308%	69.2308%	69.2308%	76.9231%	76.9231%

Query	50			100			1000		
	Baseline	QT[1)]	QE[2)]	Baseline	QT[1)]	QE[2)]	Baseline	QT[1)]	QE[2)]
230	28.5714%	14.2857%	14.2857%	28.5714%	35.7143%	35.7143%	35.7143%	42.8571%	42.8571%
231	63.1579%	36.8421%	36.8421%	78.9474%	78.9474%	84.2105%	100.0000%	100.0000%	100.0000%
232	20.0000%	20.0000%	20.0000%	20.0000%	20.0000%	20.0000%	60.0000%	60.0000%	60.0000%
233	0.0000%	0.0000%	0.0000%	0.0000%	0.0000%	0.0000%	50.0000%	25.0000%	25.0000%
234	20.0000%	40.0000%	40.0000%	20.0000%	40.0000%	40.0000%	70.0000%	50.0000%	50.0000%
235	15.7895%	25.5639%	25.5639%	23.3083%	38.3459%	38.3459%	56.3910%	72.9323%	72.9323%
236	0.0000%	0.0000%	0.0000%	0.0000%	0.0000%	0.0000%	26.1905%	28.5714%	28.5714%
237	18.7500%	15.0000%	15.0000%	26.2500%	31.2500%	31.2500%	56.2500%	82.5000%	82.5000%
238	7.2464%	8.6957%	8.6957%	11.5942%	11.5942%	11.5942%	36.9565%	31.1594%	31.8841%
239	2.2988%	6.8966%	6.8966%	4.5977%	11.4943%	11.4943%	27.5862%	43.6782%	43.6782%
240	2.3810%	1.1905%	1.1905%	2.9762%	2.9762%	2.3810%	25.0000%	27.9762%	27.9762%
241	0.0000%	3.5714%	3.5714%	0.0000%	3.5714%	3.5714%	17.8571%	25.0000%	25.0000%
242	42.1053%	63.1579%	63.1579%	47.3684%	68.4211%	68.4211%	94.7368%	89.4737%	89.4737%
243	17.6471%	11.7647%	11.7647%	32.3529%	32.3529%	32.3529%	88.2353%	82.3529%	82.3529%
244	12.5561%	12.5561%	12.5561%	25.5605%	14.7982%	14.7982%	96.8610%	91.0314%	91.0314%
245	0.0000%	45.4545%	45.4545%	36.3636%	54.5455%	54.5455%	72.7273%	90.9091%	90.9091%
246	3.6364%	2.4242%	2.4242%	5.4545%	7.2727%	7.2727%	50.9091%	52.7273%	52.7273%
247	35.0000%	40.0000%	40.0000%	40.0000%	40.0000%	40.0000%	60.0000%	60.0000%	60.0000%
248	18.1818%	0.0000%	0.0000%	27.2727%	27.2727%	27.2727%	72.7273%	72.7273%	72.7273%
249	12.1951%	26.8293%	26.8293%	19.5122%	46.3415%	46.3415%	85.3659%	85.3659%	85.3659%
250	5.4545%	10.9091%	14.5455%	9.0909%	23.6364%	23.6364%	50.9091%	65.4545%	65.4545%
	25.4653%	28.6726%	29.3531%	33.3590%	40.5374%	40.6504%	69.6548%	73.8043%	74.3275%

[1)] QT = Query Translator
[2)] QE = Query Expander

APPENDIX C – PRECISION PER QUERY

Query	50			100			1000		
	Baseline	QT[1)]	QE[2)]	Baseline	QT[1)]	QE[2)]	Baseline	QT[1)]	QE[2)]
202	34.0000%	64.0000%	64.0000%	36.0000%	46.0000%	46.0000%	9.5000%	9.5000%	9.5000%
203	10.0000%	10.0000%	10.0000%	5.0000%	8.0000%	8.0000%	1.0000%	1.0000%	1.0000%
204	10.0000%	26.0000%	24.0000%	10.0000%	24.0000%	23.0000%	10.2000%	11.2000%	11.5000%
205	2.0000%	2.0000%	2.0000%	3.0000%	3.0000%	3.0000%	2.2000%	2.7000%	2.7000%
206	0.0000%	22.0000%	22.0000%	2.0000%	16.0000%	16.0000%	2.2000%	2.0000%	2.0000%
207	28.0000%	26.0000%	26.0000%	21.0000%	23.0000%	23.0000%	3.0000%	3.1000%	3.1000%
208	14.0000%	14.0000%	12.0000%	10.0000%	10.0000%	9.0000%	1.8000%	2.3000%	2.3000%
209	32.0000%	24.0000%	24.0000%	19.0000%	24.0000%	24.0000%	4.4000%	5.1000%	5.1000%
210	34.0000%	42.0000%	42.0000%	21.0000%	23.0000%	23.0000%	4.1000%	5.0000%	5.0000%
211	34.0000%	38.0000%	38.0000%	40.0000%	26.0000%	26.0000%	13.8000%	13.8000%	13.8000%
212	24.0000%	20.0000%	20.0000%	14.0000%	17.0000%	17.0000%	2.1000%	3.3000%	3.3000%
213	8.0000%	8.0000%	8.0000%	5.0000%	5.0000%	5.0000%	1.0000%	0.9000%	0.9000%
214	6.0000%	6.0000%	6.0000%	3.0000%	4.0000%	4.0000%	0.4000%	0.4000%	0.4000%
215	40.0000%	52.0000%	58.0000%	34.0000%	44.0000%	51.0000%	8.1000%	6.8000%	8.6000%
217	14.0000%	26.0000%	26.0000%	10.0000%	19.0000%	19.0000%	2.0000%	2.5000%	2.5000%
218	6.0000%	6.0000%	8.0000%	4.0000%	4.0000%	4.0000%	0.4000%	0.4000%	0.4000%
219	26.0000%	30.0000%	28.0000%	18.0000%	22.0000%	24.0000%	3.3000%	3.6000%	3.7000%
220	6.0000%	6.0000%	6.0000%	3.0000%	3.0000%	3.0000%	0.3000%	0.3000%	0.3000%
221	14.0000%	26.0000%	24.0000%	12.0000%	17.0000%	16.0000%	6.7000%	6.5000%	6.9000%
222	26.0000%	44.0000%	56.0000%	19.0000%	30.0000%	32.0000%	4.4000%	5.2000%	5.2000%
223	14.0000%	16.0000%	16.0000%	8.0000%	8.0000%	8.0000%	0.8000%	0.8000%	0.8000%
224	32.0000%	46.0000%	46.0000%	22.0000%	28.0000%	28.0000%	4.6000%	5.0000%	5.0000%
225	38.0000%	32.0000%	30.0000%	30.0000%	28.0000%	21.0000%	6.1000%	5.6000%	5.3000%
226	40.0000%	22.0000%	22.0000%	26.0000%	28.0000%	28.0000%	5.8000%	7.7000%	7.7000%
227	26.0000%	30.0000%	30.0000%	16.0000%	25.0000%	25.0000%	8.8000%	11.6000%	11.6000%
228	10.0000%	2.0000%	2.0000%	9.0000%	2.0000%	2.0000%	1.9000%	2.1000%	2.1000%
229	16.0000%	14.0000%	14.0000%	8.0000%	9.0000%	9.0000%	0.9000%	1.0000%	1.0000%

Query	50			100			1000		
	Baseline	QT[1)]	QE[2)]	Baseline	QT[1)]	QE[2)]	Baseline	QT[1)]	QE[2)]
230	8.0000%	4.0000%	4.0000%	4.0000%	5.0000%	5.0000%	0.5000%	0.6000%	0.6000%
231	24.0000%	14.0000%	14.0000%	15.0000%	15.0000%	16.0000%	1.9000%	1.9000%	1.9000%
232	2.0000%	2.0000%	2.0000%	1.0000%	1.0000%	1.0000%	0.3000%	0.3436%	0.3436%
233	0.0000%	0.0000%	0.0000%	0.0000%	0.0000%	0.0000%	0.2000%	0.1000%	0.1000%
234	4.0000%	8.0000%	8.0000%	2.0000%	4.0000%	4.0000%	0.7000%	0.5000%	0.5000%
235	42.0000%	68.0000%	68.0000%	31.0000%	51.0000%	51.0000%	7.5000%	9.7000%	9.7000%
236	0.0000%	0.0000%	0.0000%	0.0000%	0.0000%	0.0000%	1.1000%	1.2000%	1.2000%
237	30.0000%	24.0000%	24.0000%	21.0000%	25.0000%	25.0000%	4.5000%	6.6000%	6.6000%
238	20.0000%	24.0000%	24.0000%	16.0000%	16.0000%	16.0000%	5.1000%	4.3000%	4.4000%
239	4.0000%	12.0000%	12.0000%	4.0000%	10.0000%	10.0000%	2.4000%	3.8000%	3.8000%
240	8.0000%	4.0000%	4.0000%	5.0000%	5.0000%	4.0000%	4.2000%	4.7000%	4.7000%
241	0.0000%	2.0000%	2.0000%	0.0000%	1.0000%	1.0000%	0.5000%	0.7000%	0.7000%
242	16.0000%	24.0000%	24.0000%	9.0000%	13.0000%	13.0000%	1.8000%	1.7000%	1.7000%
243	12.0000%	8.0000%	8.0000%	11.0000%	11.0000%	11.0000%	3.0000%	2.8000%	2.8000%
244	56.0000%	56.0000%	56.0000%	57.0000%	33.0000%	33.0000%	21.6000%	20.3000%	20.3000%
245	0.0000%	10.0000%	10.0000%	4.0000%	6.0000%	6.0000%	0.8000%	1.0000%	1.0000%
246	12.0000%	8.0000%	8.0000%	9.0000%	12.0000%	12.0000%	8.4000%	8.7000%	8.7000%
247	14.0000%	16.0000%	16.0000%	8.0000%	8.0000%	8.0000%	1.2000%	1.2000%	1.2000%
248	4.0000%	0.0000%	0.0000%	3.0000%	3.0000%	3.0000%	0.8000%	0.8000%	0.8000%
249	10.0000%	22.0000%	22.0000%	8.0000%	19.0000%	19.0000%	3.5000%	3.5000%	3.5000%
250	6.0000%	12.0000%	16.0000%	5.0000%	13.0000%	13.0000%	2.8000%	3.6000%	3.6000%
	17.0000%	20.2500%	20.5417%	12.9375%	15.5625%	15.5833%	3.8042%	4.1134%	4.1634%

[1)] QT = Query Translator
[2)] QE = Query Expander

BIBLIOGRAPHY

Allen J. 1995. *Natural Language Understanding.* Second Edition. The Benjamin/Cummings Publishing Company, Inc., 1995.

Anil S. Chakravarthy and Kenneth B. Haase. 1995. *NetSerf: Using Semantic Knowledge to Find Internet Information Archives.* Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR '95), July 9-13, 1995, Seattle, WA.

Buckley C., Salton G., Allan J., and Singhal A. 1994. *Automatic query expansion using SMART: TREC3.* In Donna Harman, editor, Proceedings of the Third Text Retrieval Conference (TREC-3), pages 69-80, Gaithersburg, MD. NIST and ARPA, 1994.

Frakes W. and Baeza-Yates R., editors. 1992. *Information Retrieval: Data Structures & Algorithms.* Prentice Hall, Englwood Cliffs, New Jersey, 1992.

Grefenstette G . 1992. *Use of Syntactic Context to Produce Term Association Lists for Text Retrieval.* In Proceedings of the 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Copenhagen, Denmark, 1992.

Harman D., editor. 1996. *Proceedings of the Fourth Text retrieval Conference TREC-4.* NIST Special Publication, 1996.

Kelledy F. 1997. *Query Space Reduction in Information Retrieval.* Ph.D. Thesis, February, 1997.

Lin D. 1993. *Principle-based parsing without overgeneration.* In Proceedings of ACL-- 93, pages 112--120, Columbus, Ohio, 1993.

Lin D. 1995. *University of Manitoba: Description of the PIE System as Used for MUC-6.* In Proceedings of the Sixth Conference on Message Understanding (MUC-6), Columbia, Maryland, 1995.

Lin D. 1998. *Extracting Collocations from Text Corpora.* First Workshop on Computational Terminology, pp.57--63. Montreal, Canada, August, 1998.

Miller G., Beckwith R., Fellbaum C., Gross D., and Miller K., 1990. *Introduction to WordNet: An on-line lexical database.* International Journal of Lexicography, volume 3, number 4, p. 235-244, 1990.

Mirta M., Singhal A., Buckley C. 1998. *Improving Automatic Query Expansion.* SIGIR '98, Australia, 1998.

Nassar N. 1997. *Searching with Isearch.* Web Techniques Magazine, Volume 2, Issue 5, May 1997.

Salton G. 1971. *The SMART Retrieval System.* Englewood Cliffs, N.J.: Prentice Hall, Inc, 1971.

Strzalkowski T., Lin F., Perez-Carballo J. 1997. *Natural Language Information Retrieval: TREC-6 Report.* NIST, 1997.

Vlez B., Weiss R., Sheldon M., and Gifford D. 1997. *Fast and Effective Query Refinement.* In Proceedings of the 20th ACM SIGIR Conference on Research and Development in Information Retrieval. Philadelphia, Pennsylvania, USA, 1997.

Voorhees E. 1994. *Query Expansion Using Lexical-Semantic Relations.* In Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Dublin, Ireland, 1994.

Voorhees E. and Harman D. 1996. *Overview of the Fourth Text REtrieval Conference (TREC-4).* National Institute of Standards and Technology, 1996.

Xu J. and Croft W. 1996. *Query Expansion Using local and Global document Analysis.* In Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Zurich, Switzerland, 1996.