Query Expansion Using WordNet

Submitted by:

Parantapa Goswami

Supervisor

Mandar Mitra
Associate Professor
Computer Vision and Pattern Recognition Unit
Indian Statistical Institute

A DISSERTATION SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF TECHNOLOGY IN COMPUTER SCIENCE

Indian Statistical Institute
203, Barrackpore Trunk Road
Kolkata 700108

CERTIFICATE

I certify that I have read the thesis prepared under my guidance by Parantapa Goswami, entitled **"Query Expansion Using WordNet"** and in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Master of Technology in Computer Science of Indian Statistical Institute.

Mandar Mitra
Associate Professor
Computer Vision and Pattern Recognition Unit
Indian Statistical Institute

Abstract

Query expansion is an effective technique to improve the performance of information retrieval systems. Intuitively, hand-crafted lexical resources, like WordNet, should provide reliable related terms for expanding queries. Most previous studies have shown that query expansion using only WordNet leads to very limited performance improvements. However, a recent study has shown the effectiveness of query expansion using WordNet within the recently proposed axiomatic framework. In this thesis, we re-examine the problem using the BM25 model. By defining new term weighting strategies, we are able to use the lexical information within WordNet to expand queries effectively. We obtained notable improvements while testing on the TREC7 and TREC8 collections. We observed that massive expansion leads to better performance. A tentative explanation of this observation is also explored.

Acknowledgement

First and foremost I offer my sincerest gratitude to my supervisor, Dr. Mandar Mitra, who has supported me throughout my dissertation with his patience and knowledge whilst allowing me the room to work in my own way. I attribute the level of my Masters degree to his encouragement and effort and without him this thesis, too, would not have been completed or written. One simply could not wish for a better or friendlier supervisor.

All the members of Information Retrieval Laboratory of CVPR Unit, ISI made it a convivial place to work. In particular, I would like to thank Dipasree Pal and Jiaul Hoque Paik for their help and admiration in the past one year. All other folks, including Sauparna Palchowdhury, Ayan Bandapadhyay, Kripabandhu Ghosh had inspired me in research and life through our interactions during the long hours in the lab.

My deepest gratitude goes to my father, mother and grandmother for their unflagging love and support throughout my life. I would also like to thank my friends for giving me a wonderful time to spend with.

Contents

CHAPTER 1

A Brief Overview on Information Retrieval

Archiving and finding information from archives efficiently has been practiced since 3000BC [10]. Back then, Sumerians designated special areas to store clay tablets with cuneiform inscriptions. They even designed efficient methodologies to find information from these archives.

Nowadays with the invention of computers, it has become possible to store large amount of information. Hence, to access and retrieve information from such large collection efficiently, has become a necessity. In 1945 Vannevar Bush published a article titled *As We May Think* which gave birth to the the concept of automated retrieval of information from large amount of stored data [10]. The field of information retrieval has advanced very fast over last sixty years. Currently various information retrieval based applications has been developed and commercialized. Many of them, such as a web search engine, has become an integrated part of day to day human life.

In this chapter we briefly discuss the basic concepts of information retrieval.

1.1 A Definition

The term *information retrieval* (IR) is used very broadly. Just searching a phone number from the phone book of a mobile is a form of information retrieval as well as searching a topic from web by some search engine. As an academic field of study *information retrieval* can be defined as [16]:

Definition 1.1.1. Information retrieval (IR) is finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections (usually stored on computers).

1.1.1 Document and Collection

A *document* is a file containing significant text content. It has some minimal structures e.g., title, author, date, subject etc. Examples of documents are web pages, email, books, news stories, scholarly papers, text messages, MSWord, MSPowerpoint, PDF, forum postings, blogs etc.

A set of similar documents is called *collection.* Generally all activities of an IR system is performed on a collection of documents with a pre-defined structure or format (e.g., normal text file, pdf, MSWord etc.).

1.1.2 Unstructured Text and Database Records

Databases are well structured data set. Database records are made up of well-defined fields or attributes. Each records has an unique key (either natural or imposed) by which it can be identified uniquely. Hence searching is very efficient. But content of an database must correspond to the nature of attributes. Whereas documents contain free, unstructured text or (only minimal structure like title, author etc.). Hence there is no restriction on a content of a document. But as there is no structure imposed, searching something in an document is very difficult. IR mainly handles searching mechanisms over documents.

Searching in a database is easy because, it is easy to compare fields of database with well-defined semantics of queries to find matches. Hence, queries must also maintain a proper structure. An example database query might be:

```
select name from Customer where balance > 50,000
```

Here `Customer` is a table of a bank database having `name` and `balance` attribute. On the other hand, just like documents, the queries in IR can also be free unstructured text, hence user friendly. An example IR query might be:

```
branches of SBI in south Kolkata
```

Information retrieval is fast becoming the dominating form of information access, overtaking traditional database style searching.

1.2 Information Retrieval Procedure

The retrieval process is discussed in this section. Such a process is interpreted in terms of two main component sub-processes - *Indexing* and *Retrieval.* Figure 1.1 explains the overview of the procedure.

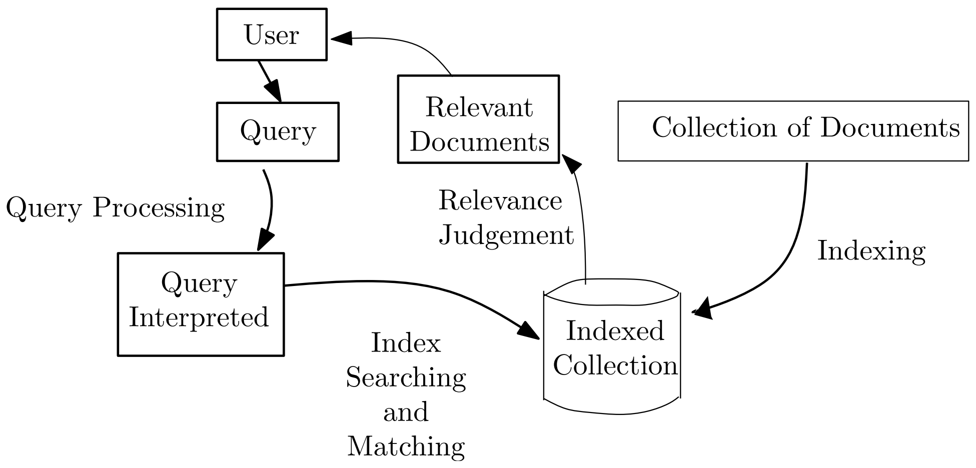


Figure 1.1: Outline of IR Procedure.

1.2.1 Indexing

First of all, before the retrieval process can even be initiated, it is necessary to define the text collection. This usually specifies the following:

1. the documents to be used
2. the operations to be performed on the text
3. the text model (i.e., the text structure and what elements can be retrieved)

Then documents within the collection is indexed. Indexing involves processing each document in a collection and to build a data structure of indexed documents. Following are the steps of indexing:

1. Reading and parsing a document.
2. Stopword removal and stemming of each term in the document.
3. Inserting each term in the data structure of indexed documents.

Efficiency of the IR system depends on the data structure to store indexed documents. Hence proper design of the data structure is of utmost importance. In the following subsections two alternative data structures have been discussed.

Term Document Incidence Matrix

Most obvious data structure is *Term Document Incidence Matrix*. Here we assume each document is a set of terms and each document is identified by a unique serial number, called document

ID. Now a matrix is formed where rows corresponds to terms and columns corresponds to documents (i.e. document IDs). Structure of a typical term document incidence matrix is shown in figure 1.2. Here t_is are the terms present in all the documents and $docID_j$s are document IDs.

$$\begin{array}{ccccc}
 & docID_1 & docID_2 & \cdots & docID_j \\
t_1 & 0 & 1 & & 0 \\
t_2 & 1 & 0 & \cdots & 1 \\
t_3 & 0 & 1 & & 0 \\
\vdots & & \vdots & & \vdots \\
t_i & 0 & 1 & \cdots & 1
\end{array}$$

Figure 1.2: Term Document Incidence Matrix.

But this data structure is space inefficient for a large collection. Then number of rows and columns will increase rapidly and most likely the matrix will be a sparse one.

Inverted Index

The alternative is *Inverted Index*. Here for term t, inverted index stores a list of IDs of all documents containing t. Then the term set is organized in a suitable data structure, e.g., array, hash table, binary search tree etc. Structure of a typical inverted index is shown in figure 1.3.

$$\begin{tikzcd}
\boxed{t_1} \arrow[r] & \boxed{docID_i} \arrow[r] & \boxed{docID_j} \arrow[r, dash] & \cdots \\
\boxed{t_2} \arrow[r] & \boxed{docID_m} \arrow[r] & \boxed{docID_i} \arrow[r, dash] & \cdots \\
\vdots & & &
\end{tikzcd}$$

Figure 1.3: Inverted Index.

1.2.2 Retrieval

Given that the document collection is indexed, the retrieval process can be initiated.

The user first specifies a information need via a query, which is then parsed and stemmed by the same parser and stemmer applied to the documents while indexing. This transformed query provides a system representation for the user's information need. The query is then processed to obtain the retrieved documents. Indexed terms are searched and matched with the query terms. If a matching is found, documents containing the matched term are considered as relevant documents. Fast term searching matching and is made possible by the index structure previously built.

Before been sent to the user, the retrieved documents are ranked according to some ranking function. The user then examines the set of ranked documents in the search for useful information. At this point, he might pinpoint a subset of the documents seen as definitely of interest and initiate a user feedback cycle. In such a cycle, the system uses the documents selected by the user to change the query formulation. Hopefully, this modified query is a better representation of the real user need. This procedure is known as *Relevance Judgement*.

1.3 Evaluation of an IR System

1.3.1 Standard Test Collection

To evaluate the performance of an ad hoc information retrieval system a test collection consisting of following three things is required:

1. A document collection.
2. A test suite of queries.
3. A set of relevance judgements, which is a binary assessment of either relevant or relevant for each query-document pair.

Document collection and query suite must be of reasonable size. As a rule of thumb, 50 queries has usually been found to be a sufficient minimum.

There exists a number of standard test collection and evaluation series. For ad hoc retrieval some most standard test collections are TREC [7], FIRE [2], CLEF [1].

The standard approach to information retrieval system evaluation revolves around the notion of relevant and non-relevant documents. With respect to a user information need, a document in the test collection is given a binary classification as either relevant or non-relevant. This decision is referred to as the *gold standard* or *ground truth judgment* of relevance.

1.3.2 Evaluation of Unranked Retrieval Sets

The two most basic parameters for performance measurement of an IR system are *precision* and *recall*. These are initially defined for the simple case where the IR system returns only a set of documents. These definitions can be extended for IR systems which returns a set of documents along with ranks.

Precision is the fraction of the documents retrieved that are relevant to the user's information need.

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

Recall is the fraction of the documents that are relevant to the query that are successfully retrieved.

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

1.3.3 Evaluation of Ranked Retrieval Results

The ranked retrieval results are now standard with search engines. In a ranked retrieval context, appropriate sets of retrieved documents are naturally given by the top k retrieved documents.

In recent years, *Mean Average Precision (MAP)*has become a standard parameter. It has been shown that MAP has especially good discrimination and stability among evaluation measures [16].

The concept of *Average Precision* is required to define MAP. For a single query, average precision is the average of the precision value obtained for the set of top k documents existing after each relevant document is retrieved, and this value is then averaged over information needs. Let the set of relevant documents for a query q_j is $D = \{d_1, \ldots d_{m_j}\}$. Let rel_{jk} is 1 if $d_k \in D$ is relevant to query q_j, 0 otherwise. Average precision for query q_j is defined as:

$$AP_{q_j} = \sum_{i=1}^{m_j} P(i) \times rel_{ji}$$

Here $P(i)$ is precision at i. It is defined as –

$$P(i) = \frac{\text{number of relevant documents in the rank list upto } i^{th} \text{ position}}{i}$$

When any relevant document is not retrieved, the average precision value in the above equation is taken to be 0.

Let the query set is Q. MAP of Q is the average of AP_{q_j} for all $q_j \in Q$. So:

$$MAP = \frac{1}{|Q|} \sum_{j=1}^{|Q|} AP_{q_j}$$

1.4 Information Retrieval Models

For the information retrieval to be efficient, the documents and queries are typically transformed into a suitable representation. There are several models such as vector space model, probabilistic model. Here *Okapi BM25* is discussed which is based on the probabilistic retrieval model.

1.4.1 Okapi BM25

The *Okapi BM25* ranking function was developed by Stephen E. Robertson, Karen Sparck Jones and others. Originally it is called BM25. It was first implemented by Okapi Information Retrieval System, and thus called Okapi BM25.

BM25 is a bag-of-words retrieval function that ranks a set of documents based on the query terms appearing in each document, regardless of the inter-relationship between the query terms within a document. It is not a single function, but actually a whole family of scoring functions, with slightly different components and parameters. One of the most prominent instantiations of the function is as follows [4][19].

Given a query q_j, containing keywords $t_{j1}, t_{j2}, \ldots, t_{jn}$, the BM25 score of a document d_i is:

$$score(d_i, q_j) = \sum_{k=1}^{n} IDF(t_{jk}) \frac{tf_{t_{jk},d_i}(k_1 + 1)}{tf_{t_{jk},d_i} + k_1(1 - b + b\frac{|d_i|}{avgdl})}$$

Here tf_{t_{jk},d_i} is the term frequency of t_{jk} in the document d_i, $|d_i|$ is the length of the document d_i in words, and *avgdl* is the average document length in the text collection from which documents are drawn. k_1 and b are free parameters. $IDF(t_{jk})$ is the inverse document frequency weight of the query term t_{jk}. It is usually computed as:

$$IDF(t_{jk}) = \log \frac{N - n(t_{jk}) + 0.5}{n(t_{jk}) + 0.5}$$

Where N is the total number of documents in the collection, and $n(t_{jk})$ is the number of documents containing t_{jk}.

CHAPTER 2

Query Expansion

A document may not explicitly contain the terms present in the query. Still the document may be relevant with respect to the idea of information need presented by the query. If a relevant document does not contain the terms that are in the query, then that document will not be retrieved. The aim of *query expansion* is to reduce this query-document mismatch by expanding the query using words or phrases with a similar meaning or some other statistical relation to the set of relevant documents.

Users often attempt to address this problem by manually refining a query. In this chapter we discuss methodologies in which a system can help with query refinement, either fully automatically or with the user in the loop.

2.1 Relevance Feedback

Relevance Feedback (RF) found to be one of the most powerful methods for improving IR performance. Improvements of 40 to 60 percent in precision noted.(Salton 1989, 322). RF is an iterative process, best modelled as a continuous loop and is a user-centered approach. It is mainly based on the two-fold idea that:

1. Relevant documents should be more strongly similar to each other than they are to non-relevant documents.
2. Users are the best judges of relevance.

The central idea of relevance feedback is to utilise the terms or expression from the documents which have been marked as relevant to reformulate the query. On the other hand information from irrelevant documents can also be used as a negative emphasis to the query reformulation. There are mainly three types of relevance feedback[12].

Explicit relevance feedback. Here an interactive user of the system explicitly marks a few top ranked document as relevant or irrelevant to their information need.

Implicit relevance feedback. In this case users do not explicitly mark the documents, but documents which are viewed acts as a feedback and the viewing information is used to reformulate the query.

Pseudo-relevance feedback. In this form no user interaction is involved. It is assumed that k top ranked documents are relevant and learn from these *pseudo-relevant* documents to improve the performance of the system.

2.1.1 Rocchio Algorithm for Relevance Feedback

Rocchio Classification algorithm is a implementation of relevance feedback. This approach is developed based on vector space model[5]. The main goal is to find a query vector which maximizes similarity with relevant documents and minimizes similarity with non-relevant documents. This is shown in figure 2.1.

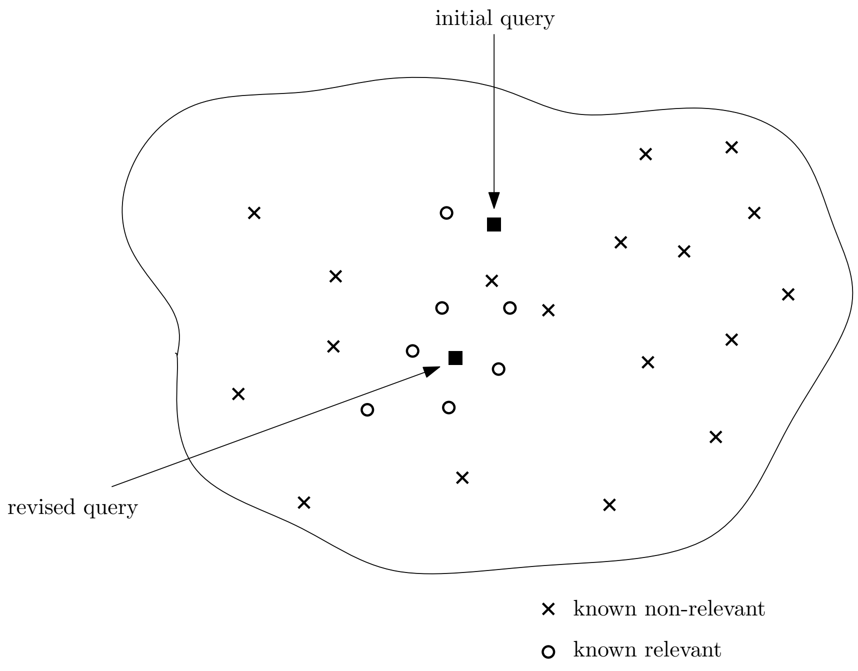


Figure 2.1: An application of Rocchio algorithm

In an IR query context we have an user query and partial knowledge of known relevant and non-relevant documents. The algorithm proposes following formula:

$$\vec{q_m} = \alpha \vec{q_0} + \beta \frac{1}{|D_r|} \sum_{\vec{d_j} \in D_r} \vec{d_j} - \gamma \frac{1}{|D_{nr}|} \sum_{\vec{d_j} \in D_{nr}} \vec{d_j}$$

Here $\vec{q_m}$ is the modified query vector, $\vec{q_0}$ is the original query vector, D_r is the set of relevant documents and D_{nr} is the set of non-relevant documents. d_j signifies an individual document vector.

α, β and γ are weights assigned to original query, set of relevant documents and set of non-relevant documents respectively. Values of these three parameters are responsible for shaping the modified query vector in a direction closer, or farther away, from the original query, related documents, and non-related documents. For example, if we have a lot of judged documents, we would like a higher β and γ. Starting from q_0, the new query moves some distance toward the centroid of the relevant documents and some distance away from the centroid of the non-relevant documents.

Relevance feedback can improve both recall and precision. But, in practice, it has been shown to be most useful for increasing recall in situations where recall is important. Positive feedback also turns out to be much more valuable than negative feedback, and so most IR systems set $\gamma < \beta$[16].

2.1.2 Drawbacks of Relevance Feedback

The success of relevance feedback depends on certain assumptions. The user has to have sufficient knowledge to be able to make an initial query which is at least somewhere close to the documents they desire. This is needed anyhow for successful information retrieval in the basic case. But there are certain kinds of problems that relevance feedback cannot solve alone. Few such cases are:

- If the user spells a term in a different way to the way it is spelled in any document in the collection, then relevance feedback is unlikely to be effective.
- Documents in another language are not nearby in a vector space based on term distribution. Rather, documents in the same language cluster more closely together. So relevance feedback is not very effective for cross-lingual information retrieval.
- If the user searches for 'car' but all the documents use the term 'automobile', then the query will fail, and relevance feedback is again most likely ineffective.

The relevance feedback approach requires relevant documents to be similar to each other. That is, term distribution in all relevant documents should be similar to that in the documents marked by the users, while the term distribution in all non-relevant documents should be different from those in relevant documents.

Relevance feedback can also have some practical problems. The long queries that are generated by straightforward application of relevance feedback techniques are inefficient for a typical IR system. This results in a high computing cost for the retrieval and potentially long response times for the user. Some experimental results have also suggested that using a limited number of terms like this may give better results (Harman, 1992) though other work has suggested that using more terms is better in terms of retrieved document quality (Buckley et al., 1994b).

Relevance feedback is not necessarily popular with users. Users are often reluctant to provide explicit feedback, or in general do not wish to prolong the search interaction. Furthermore, it is often harder to understand why a particular document was retrieved after relevance feedback is applied.

2.2 Automated Query Expansion

Unlike relevance feedback, in query expansion users give additional input on query words or phrases, possibly suggesting additional query terms. Query expansion may be automated also. In that approach system suggests some alternative expanded query. The challenge here is how to generate alternative or expanded queries most suitable for the user provided query. Here two such methods are discussed here.

2.2.1 Global Analysis Using Lexical Resources

The most common form of query expansion is global analysis, using some form of thesaurus or other lexical resources. For each term t in a query, the query can be automatically expanded with synonymous and related words of t from the thesaurus or other lexical resources. Use of a lexical resource can be combined with ideas of term re-weighting, for example, weights assigned to added terms may be less than the weights assigned to the original query terms.

2.2.2 Bo1 Term Weighting Model

The *Bo1* model uses DFR framework[1] and is based on Bose-Einstein statistics. It is very similar to Rocchio's relevance feedback method[15]. In Bo1, the informativeness $w(t)$ of a term is given by the following equation.

$$w(t) = tf_t . \log_2 \frac{1 + P_n}{P_n} + \log_2(1 + P_n)$$

Here, tf_t is the term frequency of the term t in the pseudo-relevant document set, and P_n is given by $\frac{F}{N}$. F is the term frequency of the term t in the whole collection and N is the number of documents in the collection.

[1]DFR term weighting models measure the informativeness of a term, $w(t)$, by considering the divergence of the term occurrence in the pseudo-relevant set from a random distribution[15].

CHAPTER 3

WordNet

To process natural language just like a human does, a system must have information about words and their meaning. Traditionally this information is recorded in the form of *dictionary*. Due to the cost efficiency of storage space, nowadays it is very convenient to create online dictionaries. But, the dictionary entries were evolved for convenience of human readers, not for machines[17]. WordNet is a large lexical database of English which provides traditional lexicographic information based on modern computing.

This chapter gives a brief overview of WordNet, how information is organized and how it can help in information retrieval.

3.1 Definitions

Some basic definitions are given in this section.

Definition 3.1.1. The *vocabulary* of a language is defined as a set of pairs (f, s), where a *form* f is a string over a finite alphabet and a *sense* s is an element from a given set of meanings.

Definition 3.1.2. Each form with a sense in a language is called a *word* in that language. A word having more than one sense is called *polysemous*. Two words having at least one sense in common are called *synonymous*.

Definition 3.1.3. A word's usage is the set of *linguistic contexts* denoted by $\mathcal{C}$ in which the word can be used. The syntax of the language partitions $\mathcal{C}$ into syntactic categories. Nouns are denoted by subset N, verbs are denoted by subset V, and so on. These are known as *syntactic context*.

Definition 3.1.4. Within each category of syntactic contexts, further categorization is provided by *semantic contexts*, the set of contexts in which a particular f can be used to express a particular s.

Definition 3.1.5. The *morphology* of the language is defined in terms of a set $\mathcal{M}$ of relations between word forms. For example, the morphology of English is partitioned into *inflectional*, *derivational*, and *compound morphological relations*.

Definition 3.1.6. The *lexical semantics* of the language is defined in terms of a set $\mathcal{S}$ of relations between word senses. The semantic relations into which a word enters determine the definition of that word.

3.2 Basics of WordNet

WordNet superficially resembles a thesaurus, in that it groups words together based on their meanings. However, there are some important distinctions. First, WordNet interlinks not just word forms, but specific senses of words. As a result, words that are found in close proximity to one another in the network are semantically disambiguated. Second, WordNet labels the semantic relations among words, whereas the groupings of words in a thesaurus does not follow any explicit pattern other than meaning similarity[9].

In WordNet, a sense is represented by the set of (one or more) synonyms that have that sense. Latest version WordNet (WordNet 3.0) contains more than $155,287$ different word forms and more than $117,659$ different word senses, or more than $206,941$ (f, s) pairs[8].

WordNet respects the syntactic categories *noun*, *verb*, *adjective*, and *adverb*[17]. Some word forms are interpreted as different syntactic categories in different linguistic contexts. For example, word forms like ‘back’, ‘right’, or ‘well’ are interpreted as nouns in some linguistic contexts, as verbs in other contexts, and as adjectives or adverbs in other contexts. For such word forms each is entered separately into WordNet.

3.3 Semantic Relations

The semantic relations in WordNet were chosen because they apply broadly throughout English. WordNet includes the following semantic relations:

Synonymy is WordNet’s basic relation, because WordNet uses sets of synonyms (synsets) to represent word senses. Synonymy is a symmetric relation between word forms. For example, rise and ascend are synonyms of each other.

Antonymy or opposing-name is also a symmetric semantic relation between word forms, especially important in organizing the meanings of adjectives and adverbs. For example, rapidly and slowly are antonyms of each another.

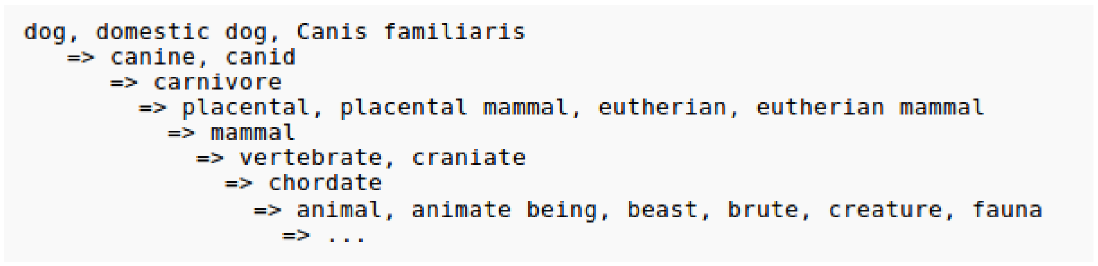
Hyponymy or sub-name and its inverse, **hypernymy** or super-name, are transitive relations between synsets. Because there is usually only one hypernym, this semantic relation organizes the meanings of nouns into a hierarchical structure. For example, palm is a kind of tree. WordNet distinguishes among Types (common nouns) and Instances (specific persons, countries and geographic entities)[9]. Thus, armchair is a type of chair, Barack Obama is an instance of a president.

Meronymy or part-name and its inverse, **holonymy** or whole-name, are complex semantic relations. This relation holds between synsets like chair and back, backrest, seat and leg. Parts are inherited from their superordinates: if a chair has legs, then an armchair has legs as well. Parts are not inherited upward as they may be characteristic only of specific kinds of things rather than the class as a whole: chairs and kinds of chairs have legs, but not all kinds of furniture have legs. WordNet distinguishes component parts, substantive parts, and member parts.

Troponymy or manner-name is for verbs what hyponymy is for nouns, although the resulting hierarchies are much shallower. For example whisper and speak.

3.4 Knowledge Structure

Both nouns and verbs are organized into hierarchies, defined by hypernymy. For instance, the hypernym hierarchy of first sense of the word ‘dog’ is given in figure 3.1. The words at the same



```
dog, domestic dog, Canis familiaris
   => canine, canid
      => carnivore
        => placental, placental mammal, eutherian, eutherian mammal
          => mammal
            => vertebrate, craniate
              => chordate
                => animal, animate being, beast, brute, creature, fauna
                  => ...
```

Figure 3.1: Hypernym hierarchy of first sense of the word ‘dog’

level are synonyms of each other: some sense of ‘dog’ is synonymous with some other senses of ‘domestic dog’ and ‘Canis lupus familiaris’, and so on. Each set of synonyms (synset), has a unique index and shares its properties, such as a gloss (or dictionary) definition.

At the top level, these hierarchies are organized into base types, 25 primitive groups for nouns, and 15 for verbs[8]. These groups form lexicographic files at a maintenance level. These primitive groups are connected to an abstract root node that has, for some time, been assumed by various applications that use WordNet.

In the case of adjectives, the organization is different. Two opposite ‘head’ senses work as binary poles, while ‘satellite’ synonyms connect to each of the heads via synonymy relations. Thus, the hierarchies, and the concept involved with lexicographic files, do not apply here the same way they do for nouns and verbs.

The network of nouns is far deeper than that for the other parts of speech. Verbs have a far bushier structure, and adjectives are organized into many distinct clusters. Adverbs are defined in terms of the adjectives they are derived from, and thus inherit their structure from that of the adjectives.

3.5 Applications of WordNet

WordNet has been used for a number of different purposes in information systems, including word sense disambiguation, information retrieval, automatic text classification, automatic text summarization, and even automatic crossword puzzle generation.

Another prominent example of the use of WordNet is to determine the similarity between words. Various algorithms have been proposed, and these include considering the distance between the conceptual categories of words, as well as considering the hierarchical structure of the WordNet ontology.

In this work, WordNet is used to determine the similarity between a term and a set of terms and the similarity information is used for automatic query expansion. More details have been discussed in the next chapter.

CHAPTER 4

Using WordNet for Automated Query Expansion

In 2.2.1 we discussed how to use lexical resources for automated query expansion. In 3.5 we described the use of WordNet to determine term similarity. In this chapter we discuss how to exploit the features of WordNet as a lexical resource for global query expansion.

4.1 Related Work

WordNet has been used by many researchers as a tool for automated query expansion, but in most of the cases the improvement in retrieval performance has not been satisfactory.

Voorhees showed that if the original queries are relatively complete descriptions of the information being sought then this technique of query expansion makes little difference in retrieval effectiveness even when the concepts to be expanded are selected by hand. But for less well developed queries, retrieval performance is significantly improved by expansion of hand-chosen concepts[20].

Stairmand used WordNet for query expansion and also reported that the improvement was restricted[18].

The first significant positive result was reported by Hui Fang[13]. The main contribution of this work is to show that automated query expansion using WordNet is effective in the recently proposed axiomatic framework[14]. Fang developed a methodology to determine the similarity of two terms using WordNet and then use this idea to expand queries[13].

Our work is based on the work of Fang. We modified the concept of term similarity and used *BM25* instead of the axiomatic framework.

4.2 Term Similarity by WordNet

In this section we discuss a term similarity function which uses lexical information from WordNet to measure similarity between a term and a set of terms.

4.2.1 Definition of a Term

WordNet is a hand crafted lexical system where words are organized in synsets [17]. Each node in Wordnet is a synset, i.e. a set of synonyms. For each synset, a definition is provided, commonly referred to as *gloss*. For a single term, all the synsets in which the term appears can be returned along with their corresponding definitions.

Definition 4.2.1. WordNet definition of a term t, denoted by $D(t)$, is the set of all words present in the definitions of all the synsets containing t.

4.2.2 Definition of a Query

A query can be considered as a set of terms. Based on the definition of each term present in the query, the definition of the query itself can be defined.

Definition 4.2.2. Let $q = \{t_1, t_2, \ldots, t_n\}$ be a query consisting of n terms; then the WordNet definition of q, denoted by $D(q)$, is defined as -

$$D(q) = \bigcup_{\forall t_i \in q} D(t_i)$$

4.2.3 Similarity of a Term with a Query

Now we can define the similarity between a term and a query based on synset definitions obtained from WordNet.

Definition 4.2.3. Similarity between a term t and a query $q = \{t_1, t_2, \ldots t_n\}$ consisting of n terms, denoted by $sim(t, q)$, is defined as -

$$sim(t, q) = \frac{|D(t) \cap D(q)|}{|D(t) \cup D(q)|}$$

4.3 Candidate Terms for Query Expansion

Let q_0 be the original query and let us call terms of q_0 as *original query terms*. Initially each term of q_0 is assigned a weight of 1.0. Documents from the collection are retrieved using the BM25 model for q_0, and ranked in decreasing order of scores. Now, the top d documents are considered as the pool of candidate terms, i.e. all the terms present in top d documents are considered as eligible terms for query expansion.

The value of the parameter d plays an important role in the performance of the system. So its value along with other parameters needs to be properly tuned to obtain optimal performance.

4.4 Obtaining Expansion Terms

For each term t in the pool of candidate terms, $sim(t, q_0)$ is computed and terms are sorted in descending order according to similarity values. Now, the top e terms are taken as expansion terms and are included in the query for the final round of retrieval.

Just like the value of d, the value of the parameter e is another deciding factor in the performance of the system and hence tuning of e is also required.

4.5 Assigning Weights to the Expansion Terms

Let the expanded query be q_e. Note that the original query terms are also included in q_e. Each term of q_e has a similarity value (i.e. $sim(t, q_0)$ for any $t \in q_e$). This value is used to calculate term weights for the expanded query.

For a term t in q_e, let $w(t)$ be the weight of t and this weight is assigned based on similarity value of t. Various normalization schemes can be applied to redistribute the term weights. In the following sections four such schemes are discussed which are used. Their impact on the performance is also shown.

4.5.1 Normalization Scheme 1

The most simple standard method is to keep the weights unchanged, i.e. the weight of a term is equal to the similarity of the term with q_0.

$$w(t) = sim(t, q_0)$$

Original query terms are also weighted using this scheme.

4.5.2 Normalization Scheme 2

Let $sim_{max} = max_{\forall t \in q_e}(sim(t, q_0))$. For a term t in q_e -

$$w(t) = \frac{sim(t, q_0)}{sim_{max}}$$

For a particular query, term weights assigned by scheme 1 may be obtained by multiplying sim_{max} with the term weights assigned by scheme 2. Since the BM25 model uses a non-linear function of term weight to score the documents (see 1.4.1) [19], the performance of these two schemes differ.

4.5.3 Normalization Scheme 3

In this scheme, some extra weightage has been given to the original query terms. Original query terms are assigned a weight 1.0 irrespective of their similarity values, while weights of expansion terms are their similarity values.

$$w(t) = \begin{cases} sim(t, q_0) & \text{if } t \in q_e \setminus q_0 \\ 1 & t \in q_0 \end{cases}$$

4.5.4 Normalization Scheme 4

In this scheme, original query terms are assigned a weight 1 irrespective of their similarity value, while, weights of expansion terms are as assigned by scheme 2.

$$w(t) = \begin{cases} \frac{sim(t,q_0)}{sim_{max}} & \text{if } t \in q_e \setminus q_0 \\ 1 & t \in q_0 \end{cases}$$

Here, $sim_{max} = max_{\forall t \in q_e}(sim(t, q_0))$.

4.5.5 Normalization Scheme 5

Here more weightage is given to the original query terms. The weight of each term is assigned as per the following equation.

$$w(t) = \begin{cases} sim(t, q_0) & \text{if } t \in q_e \setminus q_0 \\ 2 & t \in q_0 \end{cases}$$

4.6 Experiments

In this section, we experimentally evaluate the effectiveness of query expansion with the term similarity function (discussed in 4.2.3) and various normalization schemes (discussed in 4.5).

4.6.1 Experiment Design

We have used the Terrier IR Platform[1][6] to conduct all experiments. We integrated WordNet[2] with Terrier using the JAVA API for WordNet Searching[3]. All experiments are conducted over the TREC7 and TREC8 collections. Table 4.1 shows some statistics related to these collections.

We did several sets of experiments on different values of d (for $d = 2, 3, 5, 10, 15, 25, 50$). But we obtained optimal performance when $d = 3$. The value of b in the equation of BM25 (discussed in 1.4.1) is taken to be 0.75. The preprocessing stemps include stopword removal and stemming with Porter's stemmer. Both these modules are included in Terrier.

[1]version 2.2.1

[2]version 3.0

Table 4.1: Statistics of Test Collections

Collection	Description	Size	Number of Documents	Number of Queries
TREC7	TREC disks 4, 5	2GB	528K	50
TREC8	TREC disks 4, 5	2GB	528K	50

4.6.2 Results

We compare the retrieval performance of query expansion with different normalization schemes using short keyword (i.e., title-only) queries. The results for TREC7 and TREC8 are given in Figures 4.1 and 4.2 respectively. In both figures, *baseline* indicates the performance without any query expansion. All results are evaluated using MAP (mean average precision).

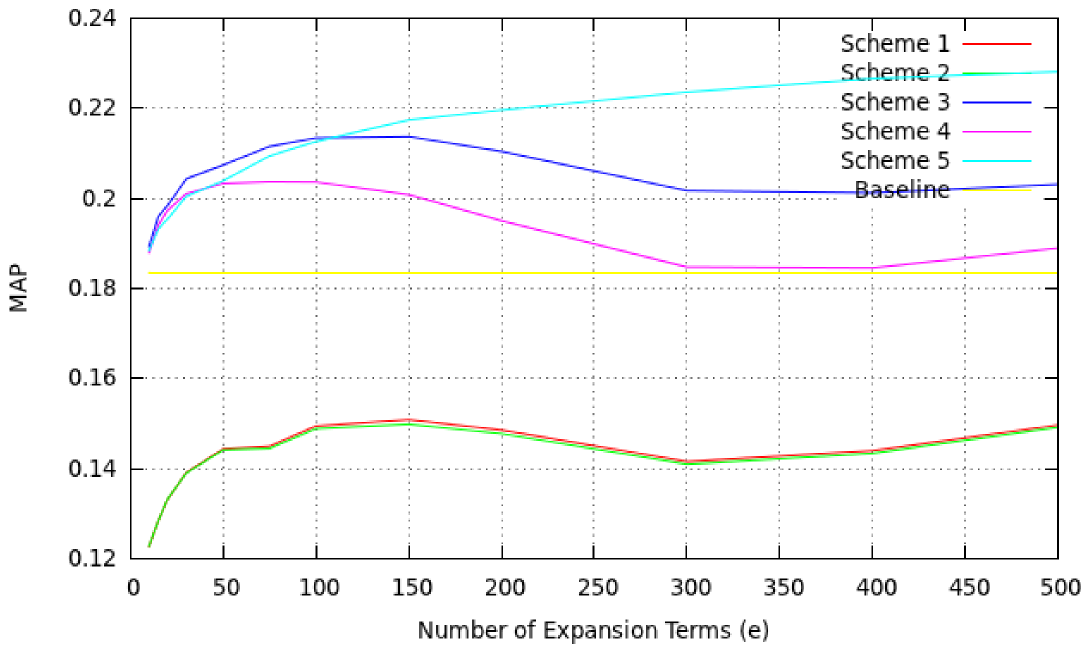


Figure 4.1: Different normalization schemes with $d = 3$ on TREC7

The results show that normalization schemes 3, 4 and 5 perform much better than scheme 1 and 2. In fact, scheme 5 performs best. Thus, when original terms are assigned greater weights than the expansion terms, performance improves.

From the figures, it is evident that notable performance improvements are obtained using a large number of expansion terms. In the following section we try to find a suitable explanation for this phenomenon.

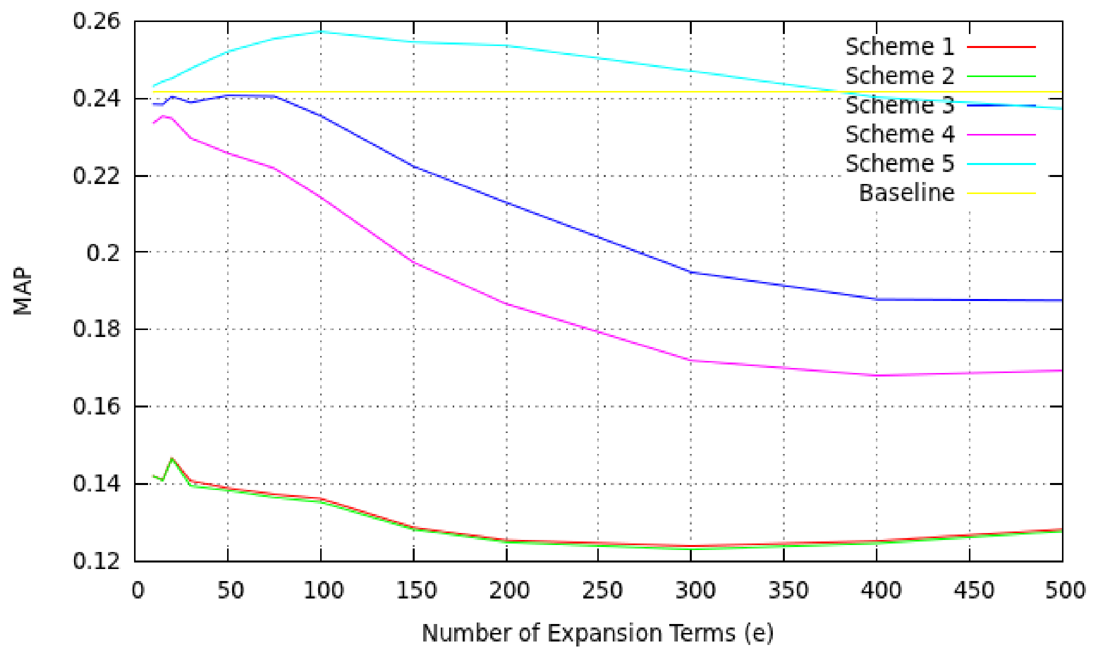


Figure 4.2: Different normalization schemes with $d = 3$ on TREC8

4.7 Effect of Number of Expansion Terms

the best MAPs obtained and percentage improvements for both the collections are shown in Table 4.2. For TREC7 this MAP is obtained at $e = 500$ and for TREC8 at $e = 100$ with normalization scheme 5.

Collection	Baseline MAP	Best MAP Obtained	Improvement
TREC7	0.1835	0.2281	24.3%
TREC8	0.2416	0.2573	6.5%

Table 4.2: Best Performances Obtained

In both the cases best performances are obtained using massive expansion, i.e. the number of expansion terms is very high.

4.7.1 Discussion

In an earlier study, Buckley and Salton showed that massive expansion performs reasonably well[11]. However, later it was found that large number of expansion terms were compensating for an improper term weight normalization method. When proper normalization is applied on term weights, massive expansion does not perform well. Given these observations and given that we have used BM25 retrieval model (which is known to use a appropriate normalization scheme), the results presented above are counter intuitive. So we did further analysis to properly explain this phenomenon.

4.7.2 Analysis

We have done the analysis mainly on the results of TREC7. These are the steps that have been followed:

1. All expansion terms are obtained and their Rocchio weights are calculated as per the equation given in 2.1.1.
2. Queries having less than 20 relevant documents are filtered out as it is usually difficult to detect consistent patterns in the results for such queries.[3]
3. For each query and for each value of e used in our experiments average precision values (AP) are tabulated. Then for each query large positive and negative changes in AP are identified and these intervals are used for further analysis.
4. For each interval identified in the previous step, the number of good terms and bad terms that are inserted in the query in the corresponding interval are enumerated. Goodness and badness of a term is determined based on the Rocchio weights calculated in step 1. Here we assumed (based on a preliminary manual inspection of the data) that a term having Rocchio weight more than 0.2 is considered a good term and a term having Rocchio weight less than -0.1 is considered as a bad term. The sum of Rocchio weights of these good and bad terms are also calculated.

When all the above information is analysed, we observed the following patterns:

1. In intervals where the AP has increased significantly, the sum of Rocchio weights is also a high positive value. Some examples of such intervals are given in Table 4.3.

Query ID	Start AP (#expansion terms)	End AP (#expansion terms)	Rocchio Sum
358	0.2585(20)	0.3322(50)	3.40513
365	0.8014(200)	0.8520(300)	2.80288
368	0.4259(0)	0.5174(20)	3.42857

Table 4.3: Large Increasing Intervals

2. In intervals where the AP has increased moderately, the sum of Rocchio weights is positive but of a more moderate magnitude. Some examples of such intervals are given in Table 4.4.
3. In intervals where the AP has decreased, the sum of Rocchio weights is either a small positive value or negative. Some examples of such intervals are given in Table 4.5.

[3] We used *qrel* file of TREC7 to find the number of relevant and non-relevant(judged) documents for the queries in step 1 and 2.

Query ID	Start AP (#expansion terms)	End AP (#expansion terms)	Rocchio Sum
365	0.7378(100)	0.7666(150)	1.23206
374	0.2291(0)	0.2541(20)	1.12849
377	0.3116(0)	0.3423(20)	1.45076

Table 4.4: Moderately Increasing Intervals

Query ID	Start AP (#expansion terms)	End AP (#expansion terms)	Rocchio Sum
360	0.2278(150)	0.2047(200)	0.059219
366	0.4015(100)	0.3708(150)	-0.046965
397	0.4220(100)	0.3855(150)	-1.98978

Table 4.5: Decreasing Intervals

Though almost all intervals under consideration follow one of the above three patterns, some notable exceptions are also observed which are shown in Table 4.6. In the first two cases, AP has increased but the Rocchio sum is negative, and in the third case AP has decreased but the Rocchio sum is a large positive value. We need to perform a more careful analysis on these exceptional cases to explain their behaviour properly.

Query ID	Start AP (#expansion terms)	End AP (#expansion terms)	Rocchio Sum
357	0.2500(300)	0.3105(400)	-1.10965
357	0.3105(400)	0.3461(500)	-0.21368
382	0.6289(150)	0.5785(200)	3.36355

Table 4.6: Exceptions

4.7.3 Conclusion

The above observations clearly show a pattern. For the intervals where AP has increased significantly, the terms inserted in the query are mostly good terms (in the Rocchio sense); whereas, in the intervals with a negative change in AP, the terms inserted in the query are mostly bad terms (in the Rocchio sense).

The overall improvement in performance as e is increased suggests that good Rocchio terms are to be found low down in the ranking of expansion terms obtained using the WordNet-derived term similarity values. More generally, the ranking of terms based on the WordNet-derived similarity and that based on Rocchio weights are expected to be different. We have computed Spearman's ρ for each query to find the correlation between these two rankings. The mean and standard deviation of Spearman's ρ over all the queries are 0.156199 and 0.122249 respectively. These values confirm our hypothesis that the rankings are more more or less independent of each other.

This is, however, a very preliminary observation. Further analysis is required to provide a more concrete explanation.

CHAPTER 5

Future Work

Following are the issues that are yet to be addressed:

1. From the results it is evident that, when larger weights are assigned to original query terms, performance improves. We have assigned a weight of 2.0 to original query terms. Further experiments are necessary to find a proper balance between weights assigned to original query terms and weights assigned to the expansion terms.

2. The analysis to find a suitable explanation for why massive expansion works is to be done in a systematic way so that clear patterns are visible in the data set. This analysis is to be performed on the TREC8 results also.

3. A careful comparison between WordNet-based query expansion and Blind Feedback-based query expansion needs to be done. This will, hopefully, suggest ways in which the best of these two methods may be combined to obtain performance superior to that achieved by the individual methods.

Bibliography

[1] Cross-Language Evaluation Forum (CLEF). `http://www.clef-campaign.org/`.

[2] Forum for Information Retrieval Evaluation (FIRE). `http://www.isical.ac.in/~clia/`.

[3] Java API for WordNet Searching. `http://lyle.smu.edu/~tspell/jaws/index.html`.

[4] Okapi BM25. `http://en.wikipedia.org/wiki/Okapi_BM25`.

[5] Rocchio Classification. `http://en.wikipedia.org/wiki/Rocchio_Classification`.

[6] Terrier IR Platform. `http://terrier.org/`.

[7] Text REtrieval Conference (TREC). `http://trec.nist.gov/`.

[8] Wordnet. `http://en.wikipedia.org/wiki/WordNet`.

[9] Wordnet - A lexical database for English. `http://wordnet.princeton.edu/`.

[10] Amit Singhal. Modern Information Retrieval: A Brief Overview. *IEEE Data Eng. Bull.*, 24(4):35–43, 2001.

[11] Chris Buckley and Gerard Salton. Optimization of Relevance Feedback Weights. In *SIGIR*, pages 351–357, 1995.

[12] Craig Macdonald. *The Voting Model for People Search.* PhD thesis, Department of Computing Science, Faculty of Information and Mathematical Sciences, University of Glasgow, 2009.

[13] Hui Fang. A Re-examination of Query Expansion Using Lexical Resources. In *ACL*, pages 139–147, 2008.

[14] Hui Fang and ChengXiang Zhai. An exploration of axiomatic approaches to information retrieval. In *SIGIR*, pages 480–487, 2005.

[15] Gianni Amati. *Probability Models for Information Retrieval based on Divergence from Randomness*. PhD thesis, Department of Computing Science, University of Glasgow, 2003.

[16] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schutze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[17] G. A. Miller, R. Beckwith, C. Fellbaum, D. Gross, and K. J. Miller. Introduction to Wordnet: An on-line lexical database. *International Journal of Lexicography*, 3(4):235–244, 1990.

[18] Mark A. Stairmand. Textual Context Analysis for Information Retrieval. In *SIGIR*, pages 140–147, 1997.

[19] Stephen E. Robertson and Hugo Zaragoza. The Probabilistic Relevance Framework: BM25 and Beyond. *Foundations and Trends in Information Retrieval*, 3(4):333–389, 2009.

[20] Ellen M. Voorhees. Query Expansion Using Lexical-Semantic Relations. In *SIGIR*, pages 61–69, 1994.