Improving relevance in search through Ontology and Query Expansion

Author: Anirudh Vemula
Supervisor: Prof. Pushpak Bhattacharyya

A report submitted in partial fulfilment of the requirements
for the degree of Bachelor of Technology

in

CFILT group
Computer Science and Engineering department

November 2014
Abstract

Prof. Pushpak Bhattacharyya  
Computer Science and Engineering department

Bachelor of Technology

Improving relevance in search through Ontology and Query Expansion

by Anirudh Vemula

From the inception of Semantic Web in the late 20th century, ontology has been a major focus to achieve the idea of semantic search. In this work, we will review different approaches that have been employed over the years to bring this idea into realization. Ontology brings in domain-specific knowledge which the keyword-based approaches lack, thereby lacking the capability of semantic search. We will see how exploiting ontology improves performance of an information retrieval system, more so when the data is very domain-specific such as Medical databases and E-commerce data. The performance improvements through ontology have been exhibited through a prototype implementation which uses a simplistic approach and still achieves significant improvement, thus showing the capabilities of ontology-based IR. We will also look into query expansion which rose as a solution to the vocabulary problem but has now become an essential part of state-of-the-art information retrieval systems. An extensive survey of the literature in the area of query expansion paves the way for our future work to introduce ontology into the query expansion framework.
Acknowledgements

I would like to take this opportunity to thank my advisor and guide, Prof. Pushpak Bhattacharyya, for his endless persistence in guiding me to my goal, suggestions and motivation. This work would not have been possible without his esteemed guidance. I would also like to thank my co-researchers Rahul Singhal and Bhargava Reddy for their useful suggestions and discussions. Last but not the least, I would like to thank Yahoo! for their initial prodding, which led me to take up ontology in IR as my dissertation topic.
# Contents

Abstract 2

Acknowledgements 3

Contents 4

List of Figures 6

Symbols 7

## 1 Basics of Information Retrieval 8

1.1 Introduction 8

1.2 Extraction of Index Terms 9

1.2.1 Pre-Processing Steps 9

1.2.2 Tokenization of the text 9

1.2.3 Elimination of Stopwords 10

1.2.4 Stemming of words 10

1.3 Traditional retrieval models 11

1.3.1 Boolean Model 11

1.3.2 Vector Space Model 12

1.3.3 Probabilistic Model 14

1.4 Retrieval evaluation 16

1.4.1 Recall and Precision 17

## 2 Motivation and Problem Definition 19

## 3 Query Expansion : A literature survey 21

3.1 Introduction 21

3.2 Utility of Query Expansion 22

3.3 Related Techniques 23

3.3.1 Interactive Query refinement 23

3.3.2 Relevance Feedback 23

3.3.3 Word Sense Disambiguation in IR 23

3.4 Stages in Query Expansion 24

3.4.1 Data Pre-processing 24
List of Figures

1.1 An example Precision versus Recall curve . . . . . . . . . . . . . . . . . . 17

3.1 Stages in Query Expansion . . . . . . . . . . . . . . . . . . . . . . . . . . 24
3.2 Classification of QE approaches . . . . . . . . . . . . . . . . . . . . . . . 29

4.1 Example of Relations in WordNet . . . . . . . . . . . . . . . . . . . . . . 32
4.2 The property of generalization implies that $\text{sim}(D, B) < \text{sim}(B, D)$ . . . 34

6.1 Precision vs Recall curve . . . . . . . . . . . . . . . . . . . . . . . . . . . 43
6.2 Precision curve . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 43
6.3 Recall curve . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 43
Symbols

$t_i$  index term

$d$  document

$w^d_{t_i}$  weight associated with $t_i$ in $d$

$m$  total number of index terms

$T$  set of all index terms

$u^d$  index term vector of the document

$g_i$  function which returns the weight of term $t_i$ in a document

$N$  total number of documents in the collection

$df(t_i)$  number of documents in which the term $t_i$ appears
Chapter 1

Basics of Information Retrieval

In this chapter, we first describe the framework of information retrieval and then describe basic techniques such as extraction of index terms, retrieval models and retrieval evaluation, briefly. Finally, we describe related techniques of information retrieval.

1.1 Introduction

Information retrieval is classically defined as, finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections (usually stored on computers), [1]. More generally, the goal of an information retrieval system is to obtain information from a knowledge resource to satisfy a user’s need. The information need is usually represented as a query, a natural language statement that must be understood by the system to process the need. Due to the inherent ambiguity within natural language, the query in an information retrieval system is always regarded as approximate and imperfect.

The knowledge resources used in the information retrieval system are usually a collection of documents of an unstructured nature. The term “unstructured” refers to data that does not have clear, semantically overt, easy-for-a-computer structure. It is the opposite of structured data, the canonical example of which is a relational database. The size of the collection also varies across different information retrieval tasks and usually distinguishes one approach from the other. Before we process the information need of the user i.e. the query, we conduct a series of pre-processing steps that are described in the following sections.
1.2 Extraction of Index Terms

Not all words are equally significant for representing semantics of the document. In written language, some words are more significant and carry more meaning than others. Usually, noun words or group of noun words are the ones that are most representative of the content of the document. Therefore, it is worthwhile to pre-process the documents in the collection to determine terms that can be used as index terms. There are several steps in this pre-processing phase which are described in this section briefly.

1.2.1 Pre-Processing Steps

The pre-processing phase can be divided into three operations, [2]

1. Tokenization of the text
2. Elimination of stopwords
3. Stemming of words

In the following subsections, each of these steps are described briefly.

1.2.2 Tokenization of the text

Tokenization is the process of converting a stream of text of the documents into a stream of words. The entire goal of tokenization is to identify words from the text. A very intuitive approach would be to split the text at whitespaces and extract words. However, there is more to it than this. Symbols such as digits, punctuation marks, hyphens and case of the letters play a role.

Numbers are not good index terms because they are inherently vague without a surrounding context. Hence, it is a good idea to disregard numbers as index terms. But this might not always be true. There are many cases where the number might prove to be a very useful index term. For example, the term ”1994” in a news document, which definitely serves a purpose in the document (it might be the year of publication) and is lost when we ignore numbers. An intelligent approach to this would involve understanding the data in the collection and using appropriate regular expressions to extract such important terms as candidates for index terms.

Hyphens pose another difficult decision for the tokenizer. The words ”state-of-the-art” and ”state of the art” mean the same and if we split the text on the hyphens, we would
essentially treat them as the same. But there are exceptions to this rule too. There are
words which include hyphens as an indispensable part such as "C-10" where splitting
the text at hyphens would result in the loss of this term. Again, the suitable approach
would be to understand the data and deal with such exceptions.

Punctuation also poses a similar difficulty as hyphens when they are an indispensable
part of the term, such as in the term "98B.C.". The case of letters is not usually
important for the identification of index terms. As a result, the tokenizer converts all
the term into either lower or upper case before moving onto further steps. However, in
some instances this might lead to loss of semantic meaning that is brought about by
the case of the letters. For example, the words Bank and bank have different meanings.
This fact is common to many other pair of words.

All these text operations must be given careful thought because they might have pro-
found impact on the performance of the retrieval system.

1.2.3 Elimination of Stopwords

Words which occur very frequently in the collection of documents are not good dis-
criminators. In fact, a word which occurs in more than 80% of the documents in the
collection is not useful as an index term. Such words are referred to as stopwords and
are filtered out as potential index terms. In addition, stopwords are words that do not
carry meaning by themselves in natural language. Generally, nouns have meanings by
themselves (independently). Therefore, articles, prepositions and conjunctions are nat-
ural candidates as stopwords. Elimination of stopwords results in a drastic decrease in
the size of indexing structure, which is among the primary advantages of eliminating
stopwords.

One of the problems in elimination of stopwords is reduction of recall in some cases. For
example, consider the query "to be or not to be", due to the removal of stopwords, this
query matches very few documents although there might be many relevant documents.
This is a reason why many web search engines adopt full text index.

1.2.4 Stemming of words

A stem is the portion of a word which is left after the removal of its affixes (i.e. prefixes
and suffixes). A typical example of a stem is the word connect which is the stem of
the words connection, connecting, connected and connections. Stems are thought to
be useful for improving retrieval performance because they reduce variants of the same
root word to a common concept. Furthermore, stemming has the secondary advantage of reducing the size of the indexing structure.

In affix removal, the most important part is suffix removal because most variants of a word are generated by the introduction of suffixes. The most popular suffix removal algorithm is the Porter’s stemmer because of its simplicity and elegance.

1.3 Traditional retrieval models

In this section, we will describe the following three classic information retrieval models:

1. Boolean model
2. Vector space model
3. Probabilistic model

The notation used in the following discussions is already detailed in the notation page at the start of the report. The weight of an index term in a document i.e. \( w_d^t \) quantifies the importance of the index term for describing the document semantic contents. We assume that index term weights are mutually independent. This is a simplification because occurrences of index terms in a document are not uncorrelated.

1.3.1 Boolean Model

The boolean model is a simple retrieval model based on set theory and boolean algebra, and is quite easy to grasp by a common user of an information retrieval system. Furthermore, the queries are specified as boolean expressions that have precise semantics.

The boolean model considers that index terms are present or absent in a document. As a result, the index term weights are assumed to be all binary i.e. \( w_d^t \in \{0, 1\} \). A query \( Q \) is composed of index terms linked by three connectives: and, or and not. Thus, a query is essentially a boolean expression that can be represented as a disjunction of conjunctive vectors.
Chapter 1. Basics of Information Retrieval

1.2 Definition Boolean Model

For the boolean model, the index term weight variables are binary i.e. \( w_{ti} \in \{0, 1\} \). A query \( Q \) is a conventional boolean expression. Let \( Q_{dnf} \) be the disjunctive normal form of \( Q \). In addition, let \( Q_{cc} \) be any of the conjunctive components of \( Q_{dnf} \). The similarity \( \text{sim}(d, Q) \) of a document \( d \) to the query \( Q \) is defined as:

\[
\text{sim}(d, Q) = \begin{cases} 
1; & \text{if } \exists Q_{cc} \mid (Q_{cc} \in Q_{dnf}) \lor (\forall t_i, g_i(w^d) = g_i(Q_{cc})) \\
0; & \text{otherwise}
\end{cases}
\]  

(1.1)

If \( \text{sim}(d, Q) = 1 \) then the boolean model predicts that the document \( d \) is relevant to the query \( Q \). Otherwise, the prediction is that the document is not relevant.

The boolean model predicts that each document is either relevant or non-relevant. There is no notion of a partial match to the query conditions. The main advantages of the boolean model are the clean formalism behind the model and its simplicity. Among the major drawbacks of the boolean model, primary drawback is that its retrieval strategy is based on a binary decision criterion without any notion of ranking, which prevents good retrieval performance. Secondly, while boolean expressions have precise semantics, it is not often simple to translate user need into a boolean expression. These drawbacks are addressed by index term weighting, which leads to notable improvement in retrieval performance.

1.3.2 Vector Space Model

The Vector space model, [3] recognises that the use of binary weights like the Boolean model is too limiting and proposes where partial matching is possible. This is accomplished by assigning non-binary weights to index terms in queries and in documents. These term weights are used to compute a similarity measure between each document and the user query. By sorting the documents in decreasing order of this similarity measure, the model returns a ranking order of the documents in relevance to the user query.

The document \( d \) and query \( q \) are represented as \( m \)-dimensional vectors of index term weights. The vector model proposes to evaluate similarity of documents \( d \) with regard to the query \( q \) as the correlation between the vectors \( w^d \) and \( Q \). This correlation can be quantified by the cosine of the angle \( \theta \) between these two vectors in \( m \)-dimensional space i.e.

\[
\text{sim}(w^d, Q) = \frac{w^d \cdot Q}{|w^d| \ast |Q|}
\]  

(1.2)
Since $w^d_i \geq 0$ and $q_t \geq 0$, $sim(w^d, Q)$ varies from 0 to 1. Thus, instead of attempting to predict whether the document is relevant or not, the vector model ranks the documents according to their similarity to the query. More generally, a document might be retrieved even if it only partially matches the query. For instance, one can establish a threshold on $sim(w^d, Q)$ and retrieve the documents with similarity measure above that threshold. But in order to compute rankings, we first need to specify how we obtain the index term weights.

In the vector space model, the raw frequency of term $t_i$ inside a document $d$ is quantified. Such a term frequency is usually referred to as the $TF$ factor and provides a measure of how well that term describes the document contents. Furthermore, the inverse of the frequency of a term $t_i$ among the documents of the collection is also quantified. This factor is usually referred to as Inverse document frequency or the $IDF$ factor. The motivation for the usage of an IDF factor is that terms which appear in many documents are not very useful in distinguishing a relevant document from a non-relevant one.

**Definition Vector Space Model**

For the vector space model, the weight $w^d_i$ is positive and not binary, unlike the boolean model. Furthermore, the index terms are also weighted. Let $q_{t_i}$ be the weight of the term $t_i$ in query vector $Q$. Thus, $Q = (q_{t_1}, q_{t_2}, \ldots, q_{t_m})$, where $m$ is the total number of index terms in the system. As before, the vector for document $d$ is represented by $w^d = (w^d_{t_1}, w^d_{t_2}, \ldots, w^d_{t_m})$.

Let $N$ be the total number of documents in the system and $df(t_i)$ be the number of documents in which the index term $t_i$ appears. Let $tf(t_i, d)$ be the raw frequency of the term $t_i$ in the document $d$. Then the normalized frequency $tf_{norm}(t_i, d)$ of term $t_i$ in document $d$ is given by:

$$tf_{norm}(t_i, d) = \frac{tf(t_i, d)}{\sum_{j=1}^{m} tf(t_j, d)}$$

If the term $t_i$ doesn’t appear in the document $d$ then $tf_{norm}(t_i, d) = 0$. Furthermore, if $idf(t_i)$, the inverse document frequency of term $t_i$ is given by:

$$idf(t_i) = \log \frac{N}{df(t_i)}$$

We combine both of these to assign weights to index terms in the following way:

$$w^d_i = tf_{norm}(t_i, d) * idf(t_i) \quad (1.3)$$

The term weighting strategy described above is popularly known as the TF- IDF scheme.
For the query term weights,

\[ q(t_i) = \left( 0.5 + \frac{0.5 \times Qf(t_i)}{\sum_{j=1}^{m} Qf(t_j)} \right) \times \log \frac{N}{df(t_i)} \]  

(1.4)

where \( Qf(t_i) \) is the raw frequency of the term \( t_i \) in the query \( q \).

The main advantages of vector space model are:

1. its term-weighting scheme improves retrieval performance
2. its partial matching strategy allows retrieval of documents that approximate query conditions
3. its cosine ranking formula sorts the documents according to their similarity to the query

Theoretically, the major disadvantage of vector space model is that it assumes index terms to be mutually independent.

Despite its simplicity, the vector space model is an effective ranking strategy with general collections. It yields ranked answers sets which are difficult to improve on without query expansion or relevance feedback within the framework of the vector space model. Although there are a wide variety of alternative ranking methods, vector space model is either superior or almost as good as the known alternatives. Furthermore, it is simple and fast. For these reasons, it is one of the most popular retrieval models.

### 1.3.3 Probabilistic Model

Robertson et. al., [4] introduced the classic probabilistic model. The model later came to be known as the Binary independence retrieval model (BIR).

The probabilistic model is based on the following fundamental assumption: Given a user query \( q \), the probabilistic model assigns to each document \( d \), as a measure of its similarity with query \( q \), the ratio, \( \frac{P(d \text{ relevant to } q)}{P(d \text{ not relevant to } q)} \) which computes the likelihood of the document \( d \) being relevant to the query \( q \). Taking the likelihood of relevance as the rank minimizes the probability of an erroneous judgement, [5] [6].
Definition: Probabilistic model

In the probabilistic model, the index term weights are all binary i.e. \( w_{d}^{t_{i}} \in \{0, 1\} \), \( w_{q} \in \{0, 1\} \). A query \( q \) is considered as a subset of index terms. Let \( R \) be the set of documents to be known (or initially guessed) to be relevant. Let \( \bar{R} \) be the complement of \( R \) i.e. set of documents to be known as non-relevant. Let \( P(R|d) \) be the probability that the document \( d \) is relevant to the query \( q \) and \( P(\bar{R}|d) \) be the probability that the document \( d \) is non-relevant to the query \( q \). The similarity \( sim(d, q) \) of the document \( d \) w.r.t. query \( q \) is given by:

\[
sim(d, q) = \frac{P(R|d)}{P(\bar{R}|d)}
\]  

(1.5)

Using Bayes rule we have,

\[
sim(d, q) = \frac{P(d|R) \times P(R)}{P(d|R) \times P(R) + P(d|\bar{R}) \times P(\bar{R})}
\]

where \( P(d|R) \) stands for the probability of randomly selecting the document \( d \) from the set \( R \). In addition, \( P(R) \) stands for the probability that a document randomly selected from the entire collection is relevant to the query \( q \). \( P(R) \) and \( P(\bar{R}) \) are the same for all documents in the collection. In other words, they are constants. Therefore,

\[
sim(d, q) \sim \frac{P(d|R)}{P(d|\bar{R})}
\]

Assuming independence of index terms,

\[
sim(d, q) \sim \prod_{g_{i}(d)=1} P(t_{i}|R) \times \prod_{g_{i}(d)=0} P(\bar{t}_{i}|R)
\]

\[
= \prod_{g_{i}(d)=1} P(t_{i}|R) \times \prod_{g_{i}(d)=0} P(\bar{t}_{i}|R)
\]

where \( P(t_{i}|R) \) stands for the probability that the index term \( t_{i} \) is present in a document randomly selected from \( R \). We can define \( P(t_{i}|R) \), \( P(\bar{t}_{i}|R) \) and \( P(t_{i}|R) \) similarly. Now, taking logarithms and noting that \( P(t_{i}|R) + P(\bar{t}_{i}|R) = 1 \) and ignoring factors which are constant for all documents in the context of the same query, we can finally write,

\[
sim(d, q) \sim \sum_{i=1}^{m} q_{t_{i}} \times \left( \log \frac{P(t_{i}|R)}{1 - P(t_{i}|R)} + \log \frac{1 - P(t_{i}|R)}{P(t_{i}|R)} \right)
\]

Since we do not know the set \( R \) in the beginning, we need to devise a method to obtain \( P(t_{i}|R) \) and \( P(t_{i}|\bar{R}) \).
In the very beginning, there are no retrieved documents. Therefore, one has to make
simplifying assumptions such as: (a) assume that \( P(t_i|R) \) is constant for all index terms
\( t \) (typically, equal to 0.5) and (b) assume that the distribution of index terms among the
non-relevant documents can be approximated by the distribution of index terms among
all documents in the collection. These two assumptions yield

\[
P(t_i|R) = 0.5,
\]

\[
P(t_i|\bar{R}) = \frac{df(t_i)}{N}
\]

Given this initial guess, we can then retrieve the documents which contain the query
terms and provide an initial probabilistic ranking for them. After that, the initial ranking
is improved.

Let \( V \) be a subset of documents initially retrieved and ranked by the probabilistic model.
For instance, such a subset can be defined as the top \( r \) ranked documents, where \( r \) is a
pre-determined threshold. Furthermore, let \( V_i \) be the subset of \( V \) containing the
documents which have the index term \( t_i \) in them. For improving the probabilistic ranking,
we need to improve our guesses for \( P(t_i|R) \) and \( P(t_i|\bar{R}) \). This can be accomplished
using the following assumptions: (a) we can approximate \( P(t_i|R) \) by the distribution
of index term \( t_i \) among the documents retrieved so far, and (b) we can approximate
\( P(t_i|\bar{R}) \) by considering that all the non-retrieved documents are non-relevant. With
these assumptions, we can write,

\[
P(t_i|R) = \frac{|V_i|}{|V|}
\]

\[
P(t_i|\bar{R}) = \frac{df(t_i) - |V_i|}{N - |V|}
\]

This process can be repeated iteratively. By doing so, we improve our guesses without
any human intervention.

### 1.4 Retrieval evaluation

In an information retrieval system, since the user query is inherently vague, the retrieved
documents are not exact answers. However, they have to be ranked in according to
the relevance w.r.t. the query. Therefore, information retrieval systems require the
evaluation of how precise the answer set is. In this section, we discuss different ways
in which we can evaluate the retrieval performance of an information retrieval system.
Such an evaluation is usually based on a test reference collection and an evaluation
measure. The test reference collection consists of a collection of documents, queries and the documents relevant to these queries as tagged by specialists.

In the following discussion, we explain two of the most popular retrieval evaluation measures: recall and precision.

### 1.4.1 Recall and Precision

Let us consider an example information request $I$ and its set $R$ of relevant documents. Let $A$ be the set of documents retrieved by the system that is evaluated for the request $I$.

**Recall**

This measure is the fraction of relevant documents which have been retrieved i.e.

$$Recall = \frac{|A \cap R|}{|R|} \quad (1.6)$$

**Precision**

This measure is the fraction of retrieved documents which are relevant i.e.

$$Precision = \frac{|A \cap R|}{|A|} \quad (1.7)$$
Recall and Precision as defined above do not take the ranking given by the retrieval system into consideration. To evaluate the ranking of the retrieved documents, we need to examine the precision versus recall curve. This curve plots the precision vs recall seen in the ranking considering different portions of the ranking itself.

For example, if the relevant documents to a query $q$ are

$$R_q = \{d_2, d_4, d_6, d_8\}$$

and the ranking retrieved by the system is: $d_2, d_3, d_9, d_4, d_5, d_6, d_7, d_1, d_8$.

If we examine this ranking, starting from the top document, we can observe the following points. First, the document $d_2$ which is ranked first is relevant to the query $q$. Furthermore, this document corresponds to 25% of the relevant document set. Thus, we can say that we have 100% precision and 25% recall. Second, the document ranked 4th is the next relevant document. At this point we can say that, we have 50% precision (two out of four retrieved documents relevant) and 50% recall (two out of relevant documents retrieved). Similarly continuing in this way, we get a set of precision-recall values that can be plotted on a curve and used as a measure of the performance of the system.
Chapter 2

Motivation and Problem
Definition

The use of ontologies to overcome the limitations of keyword-based search, such as the vector space model and the boolean model approach, has been put forward as one of the motivations of the Semantic web since its inception. While there have been several contributions in this direction, most of them partially exploit the full expressive power of an ontology-based knowledge representation, or are based on boolean retrieval models, and therefore lack an appropriate ranking model needed for scaling up to massive information sources.

In the former approach, ontologies provide a shallow representation of the information space, equivalent to taxonomies and thesauri used. The use of an ontology enables to define concepts and relations representing knowledge about a particular document in domain specific terms. In order to express the documents explicitly, it is necessary to create links (associations) between the document and relevant parts of the domain model (to which the content of the document belongs to) i.e. links to those elements of the domain model, which are relevant to the contents of the document.

This approach has brought improvements over classic keyword-based approaches through e.g. query expansion based on class hierarchies and rules on relationships, or multifaceted searching and browsing. However, these approaches have several limitations. First, because of the huge amount of information currently available to information systems in the form of unstructured text and media documents, acquiring ontology for this volume of information is very expensive. Second, documents hold a value of their own, and are not equivalent to the sum of their pieces, no matter how well formalized and interlinked. The replacement of a document by a bag of information atoms inevitably implies a loss of information value. Third, wherever ontology values carry free text, boolean semantic
search systems do a full-text search within the string values. In fact, if the string values hold long pieces of free text, a form of keyword-based approach is taking in practice beneath the ontology-based query model. While this may be manageable for small-scale systems, the boolean model doesn’t scale properly for large systems.

In this work we focus on improving the performance of ontology-based similarity in information retrieval. Instead, of retrieving documents solely on the basis of the occurrence of query terms, the documents that have terms that are semantically related to the query terms could be taken into consideration using the underlying ontology.

The indexing process maps information found in documents into the ontology, identifying concepts and their positions in the ontology. Information in queries can similarly be mapped into the ontology, and thus in addition to retrieving the exact match, the structure of the ontology can be used to retrieve semantically related documents. Our focus is on devising novel ontological concept similarity measures and, then combine ontological indexing and ontological similarity to promote semantics in the document retrieval process.

In addition to observing how ontology alone improves the performance of a retrieval system, we also observe how we can improve the performance by employing ontology within the query expansion framework. Ontology has the potential to provide the much-needed context in query expansion and can lead to better selection and generation of expansion terms for a given query. We focus on how to bring in ontology within the query expansion pipeline and devise a novel approach to find more relevant expansion terms.
Chapter 3

Query Expansion : A literature survey

3.1 Introduction

The most critical language issue for retrieval effectiveness is the term mismatch problem: the indexers and the users do often not use the same words. This is known as the vocabulary problem, [7]. Synonymy (same word with different meanings) may result in a failure to retrieve relevant documents, with a decrease in recall (the ability of the system to retrieve all relevant documents). Polysemy (different words with same/similar meaning) may cause retrieval of erroneous or irrelevant documents, thus implying a decrease in precision (the ability of the system to retrieve only relevant documents) and recall (the ability of the system to retrieve relevant documents).

To deal with this vocabulary problem, several approaches have been proposed including query refinement, relevance feedback, word sense disambiguation and search results clustering. One of the most natural and successful techniques is to expand the original query with other words that best capture the actual user intent, or that simply produce a more useful query i.e. a query that is more likely to retrieve relevant documents. This approach is called Query Expansion.

The need for approaches like query expansion is ever increasing, if we observe the trend that the information in the world is always increasing, but the number of terms used in the query by a user has more or less remained the same. Thus, given that the query is usually short and that the natural language is inherently ambiguous, simple information retrieval models are prone to errors and omissions. According to Hitwise, in 2009 the average query length was 2.30 words, the same as that reported ten years before by Lau
and Horvitz, [8]. While there has been slight increase in the number of long queries, the most prevalent queries are still those of one, two and three words. In this situation, the vocabulary problem has become serious because the lack of sufficient number of query terms to describe the user need, reduces the possibility of handling synonymy while the heterogeneity and size of the data makes the effect of polysemy more severe.

### 3.2 Utility of Query Expansion

In most information retrieval systems, the query terms are connected by an implicit OR. Under this assumption, one advantage of query expansion is that there is more chance for a relevant document that does not contain the original query terms to be retrieved, with an obvious increase in recall, [9]. For instance, if the query *McDonalds* is expanded to *McDonalds McD Big mac Ronald McDonald*, the new query will not only retrieve documents containing the term *McDonald*, but also the documents that do not use its full form and are in some way related to it (through its famous products or its mascot). Notice that query expansion brings in an improvement even when the query terms are connected by an AND (as is the case in some web search engines).

The additional query terms may however cause query drift i.e. the deviation of the focus of a search query by improper expansion, [10], thus hurting the precision. When an expansion term is correlated to a single term of the query rather than the entire query, it may easily match unrelated concepts. A further reason for a decrease in precision is that the relevant documents that match just the original query terms may move lower down in the ranking after query expansion, even if the additional terms are relevant to the query concept. For example, if the query *Gallardo* is expanded with the terms *car, Lamborghini* and *Murcielago*, a document about a different car produced by Lamborghini may be assigned a higher score than a document about Lamborghini Gallardo that does not contain the additional terms.

But it also has been seen that, [11], query expansion may also improve precision by implicitly disambiguating query terms. When the query terms are inherently ambiguous, exhibit synonymy and polysemy, the expansion terms help in disambiguating the term by providing some context in which the search can take place. Sometimes, query expansion achieves better precision in the sense that it has the effect of moving the results toward the most popular or representative meaning of the query in the collection at hand and away from other meanings, [12]. Query expansion is also seen improving precision when it is required that several aspects of the query must be present at once in the relevant document, when the user doesn’t articulate his need properly.
3.3 Related Techniques

3.3.1 Interactive Query refinement

Its main difference from automatic methods is that the system provides several suggestions for query reformulation, but the decision is made by the user. The interactive query refinement shares some of its starting stages with query expansion, in that it needs to find expansion terms to supplement the original query. But it differs in the stage where the final expanded query is to be chosen, where IQE requires human intervention but query expansion needs to intelligently chose the right one.

One of the best known systems of this kind is the suggestion that Google Search provides real-time to a user as he types his query. IQE has the potential for producing better results than query expansion, [13], but this generally requires expertise and effort on the part of the user.

3.3.2 Relevance Feedback

Relevance feedback takes the results that are initially returned from a given query and uses information provided by the user about whether or not those results are relevant to perform a new query. The content of the assessed documents is used to adjust the weights of terms in the original query and/or to add words to the query.

Popular relevance feedback techniques include variants of the Rocchio algorithm, [14]. Relevance feedback essentially reinforces the system’s original decision, by making the expanded query more similar to retrieved relevant documents, whereas query expansion tries to form a better match with the user’s underlying intentions.

3.3.3 Word Sense Disambiguation in IR

Word sense disambiguation (WSD) is the ability to identify the meaning of words in context in a computational manner, [15]. WSD is a natural and well known approach to deal with the vocabulary problem in IR. Significant improvements have been seen in retrieval performance by embedding WSD in the IR process, [16]. On the whole however, the application of WSD to IR faces limitations since a typical query context may be too short for sense disambiguation.
3.4 Stages in Query Expansion

3.4.1 Data Pre-processing

This step transforms the raw data source used for expanding the user query into a format that will be more effectively processed by subsequent steps. Pre-processing of a data source is usually independent of the particular user query that is to be expanded but it is specific to the type of data source and expansion method being considered. The steps involved are:

1. Tokenization
2. Stop word removal
3. Word Stemming
4. Word Weighting

The first three of the above list have already been explained in great detail in chapter 1. The last step simply suggests assigning a score to each word in the document that signifies the importance of the word in that document (We have already seen one form of this in the TF-IDF approach).

3.4.2 Generation and Ranking of Expansion terms

In this stage, the system generates candidate expansion terms and ranks them. The reason that feature ranking is important is that most query expansion methods will only choose a small proportion of the candidate expansion features to add to the query. The input to this stage is the original query and the transformed data source; the output is a set of expansion features, usually with associated scores. The query is also subjected to the same pre-processing steps as the document before the expansion features are generated.
There are several approaches in this stage depending upon the relationship between expansion terms generated and the query terms. We will discuss some of this approaches in the following sections.

3.4.2.1 One-to-One Associations

In this approach, each expansion term is related to a single query term. In practice, one or more expansion term is generated and scored for each query term, using a variety of techniques. One of the most natural approaches is to rely on linguistic associations.

Foremost of these linguistic approaches, is to find synonyms and related words of a query word from a thesaurus, usually from Wordnet, [17]. Expansion term generation from the Wordnet requires selecting one synset for a given query term, thus solving the ambiguity problem, and then traversing the hierarchy by following its links.

A radical departure from the linguistic approach consists of generating associations automatically by computing term-term similarities in a collection of documents ,[18]. The general idea is that two terms are semantically related if they appear in the same document, very much in the same way we assume two documents to be semantically related if they contain the same terms. More specifically, these approaches compute such term-term similarity by computing term-term correlation matrix from the term-document matrices. Consider a term-document matrix $A$, where each cell corresponds to the weight $w_{d_i}^{d}$ of the term $t_i$ in document $d$. If we calculate $C = AA^T$, then $C$ is a term-term correlation matrix, where each element can be expressed as:

$$c_{u,v} = \sum_{d_j} w_{u}^{d_j} \cdot w_{v}^{d_j}$$  \hspace{1cm} (3.1)

Using the above matrix $C$ we find correlated terms among the collection of documents. Computing co-occurrence of terms in the whole document is simple but it has the drawback that terms which are very far within the same document are less semantically related than terms which appear within the same line/paragraph (in the same document). This aspect is usually considered by factoring term proximity (considering fixed windows and find co-occurrences only within them). But this approach also suffers the major drawback that simple co-occurence doesn’t usually mean semantic similarity.

Expansion features can also be generated by mining user query logs, with the goal of associating the terms of the original query with terms in past related queries. As the texts extracted from such data are usually short (queries are usually short to-the-point texts), the standard correlation techniques cannot be applied. Thus, similarity between queries is usually done by identifying useful associations such as considering queries that
occurred in the same session, [19] or queries that yielded same set of relevant documents, [20].

### 3.4.2.2 One-to-Many Associations

One-to-One associations add an expansion term when it is strongly related to a query term. Although, in many cases this term may not be as strongly related to the query itself. A simple workaround to this is to consider one-to-many associations for a candidate expansion term. The idea is that if an expansion term is correlated to several query terms, then it is correlated to the query as a whole, [21]. If we use term-to-term correlations, we might compute the correlation factors of a given candidate expansion term with every query term and then combine the found scores to find the correlation to the query $q$:

$$c_{q,v} = \frac{1}{|q|} \sum_{u \in q} c_{u,v}$$  \hspace{1cm} (3.2)

An interesting work is [21], where concepts rather than single terms are generated as expansion terms. A concept is defined as a group of adjacent nouns in the top retrieved documents. The above equation is used to compute a query-concept correlation rather than the query-term correlation. This method is called local context analysis.

### 3.4.2.3 Term distribution in top-ranked documents

The idea is to use the initially retrieved documents to the original query as a more detailed description of the underlying query topic, from which to extract the most important terms to be used as expansion features. In a sense, the expansion terms are related to the full meaning of the query because the extracted terms are those that characterize the ”pseudo-relevant” documents as a whole, but their associations with query terms is not analyzed explicitly.

A simple approach, inspired by Rocchio’s method for relevance feedback [14], is to assign a score to each term in the top retrieved documents by a weighting function applied to the whole collection. The weights collected by each term are then summed up and the resulting score is used to sort the set of terms. This approach, termed pseudo-relevance feedback, is simple and computationally efficient, but it has the disadvantage that each term weight may reflect more the usefulness of that term with respect to the entire collection rather than its importance with respect to the user query. There are several term-ranking functions used in this approach in previous works and the ordered sets of
expansion terms suggested for each query vary depending on the term-ranking function used.

### 3.4.2.4 Query language modeling

Another commonly approach is to build a statistical language model for the query, specifying a probability distribution over terms. The best terms for the query expansion are the terms with the highest probabilities. The two main models using this approach are mixture model, [22], and relevance model, both making use of the top retrieved documents.

**Mixture Model**

In this method, we try to build a topic model for the query by exploiting the top ranked documents. We achieve this by extracting the part that is most distinct from the rest of the document collection. As there could be noise in the top retrieved documents, they can be represented as a mixture generative model that combines the query topic model $\theta_T$ (to be estimated) and the collection language model. The log likelihood of top ranked documents is as follows, where $R$ is the top retrieved document set, $c(t, d)$ is the number of occurrences of term $t$ in document $d$ and $\lambda$ is the interpolation weight.

$$
\log P(R|\theta_T) = \sum_{d \in R} \sum_t c(t, d) \log((1 - \lambda)P(t|\theta_T) + \lambda P(t|C)) \quad (3.3)
$$

The EM algorithm, [23] is then used to extract the topic model so as to maximize the log likelihood of the top ranked documents.

**Relevance Model**

In this approach, it is assumed that both the query and the top-ranked documents are samples from an unknown relevance model $\theta_{REL}$. To approximate such a model, the probability of a term $t$ is related to the conditional probability of observing that term given that we just observed the original query terms. By assuming that the $k$ query terms $q_i$ and the document terms are sampled identically and independently, the following estimate can be derived:

$$
P(t|\theta_{REL}) = \sum_{d \in R} P(d)P(t|d) \prod_{i=1}^{k} P(q_i|d) \quad (3.4)$$
3.4.3 Selection of expansion terms

After ranking the candidate features, the top elements are selected for query expansion. The selection is made on an individual basis, without considering the mutual dependencies between the expansion features. Usually only a limited number of features is selected for expansion, partly because the resulting query can be processed more rapidly, partly because the retrieval effectiveness of a small set of good terms is not necessarily less successful than adding all candidate expansion terms, due to noise reduction, [24].

3.4.4 Query Reformulation

The last stage of query expansion is query reformulation, namely how to describe the expanded query that will be submitted to the IR system. This usually amounts to assigning a weight to each feature describing the expanded query - termed query reweighting. The most popular reweighting technique is modeled after rocchio’s formula [14] for relevance feedback. A general formulation is the following, $q'$ is the expanded query, $q$ is the original query, $\lambda$ is a parameter to weight the relative contribution of query terms and expansion terms, and $score_t$ is a weight assigned to expansion term $t$:

$$ w'_{t,q'} = (1 - \lambda) \cdot w_{t,q} + \lambda \cdot score_t \quad (3.5) $$

When the expansion terms are extracted from pseudo-relevant documents and their score is computed using the documents, or Rocchio’s, weights, it is easy to show that the expanded query vector computed by the above equation moves towards the centroid of pseudo-relevant documents. The value of $\lambda$ can be adjusted so as to optimize performance if there is training data available. A typical default choice is to give more weight to original query terms when compared to expansion terms.

3.5 Approaches in Query Expansion

The approaches taken in query expansion can be expressed very broadly as shown in the figure, [25].
3.5.1 Linguistic Analysis

These techniques leverage language properties such as morphology, lexical and syntactic and semantic word relationships to expand the query. They are typically based on knowledge resources, mainly Wordnet. As the expansion terms are generated independently of the full query they are more sensitive to word sense ambiguity.

An interesting approach in this area is Ontology browsing, [26]. Knowledge models such as ontologies and thesauri provide a means for paraphrasing the user’s query in context. Most of the work in this area has focused on using Wordnet as the knowledge model, but it has several drawbacks such as the lack of proper nouns, no exact match between query and concepts, one-to-many mapping from query terms to synsets.

Another main approach is syntactic analysis, where the objective is to extract relations between the query terms, which can be used to identify expansion features that appear in related relations. The syntactic approach is most useful for natural language queries, such as in the general web search setting, [27]

3.5.2 Corpus-specific global techniques

In these techniques, we analyze the contents of the database to identify features used in similar ways. Most approaches exploit the term-term correlation matrix we described before, to find appropriate expansion terms. Other approaches include context vectors, latent semantic indexing and interlinked Wikipedia articles. Note that these approaches are data driven and do not have a linguistic explanation.
3.5.3 Query-specific local techniques

These techniques exploit the local context provided by the query. These typically make use of top-ranked documents. The most commonly used methods are analysis of term distribution and model-based query expansion. Other approaches are based on pre-processing top retrieved documents for filtering out irrelevant features prior to the utilization of a term-ranking function.

3.5.4 Search Log analysis

The idea in this class of approaches is to mine the query associations that have been implicitly out by the users, thus requiring no need on the system’s part to find out such associations through content analysis. The greatest advantage of search logs is that they encode implicit relevance feedback.

There are two main approaches in this class, first is to treat the individual queries as documents and extract features from those related to the original user query, with or without making use of their associated retrieval results, [19]. The second and more widely used technique consists of exploiting the relation of queries and retrieval results to provide additional or greater context in finding expansion features, [20]

3.5.5 Web Data

This class of approaches mainly includes exploiting a web-related resource to get better expansion terms. Popular approaches include using anchor texts and Wikipedia. Anchor texts and real user search queries are very similar because both of them represent the document in very few words. However, in absence of human intervention it is very hard to find similarity between a query and an anchor text because of their short sizes (the similarity equations described until now, don’t work well on short texts). The other approach involves using Wikipedia documents and hyperlinks to restrict the set of candidate expansion terms to only the relevant ones.
Chapter 4

Ontology-based IR : A literature survey

Up until the last decade of 20th century, ontology had primarily been a discipline in philosophy. In recent years, however, ontology has also emerged as a research area related to computer science. In computer science, we define ontology to be *a theory concerning the kinds of entities and specifically the kinds of abstract entities that are to be admitted to a language system*. A more concise and to-the-point interpretation of ontology is given by Gruber, [28], *An ontology is an explicit specification of a conceptualization*.

4.1 Ontology in Wordnet

WordNet is a large lexical database for English, [29]. The unit of WordNet is words, as the name indicates, even though it contains compounds, phrasal verbs, collections, and idiomatic phrases. The fundamental idea behind WordNet was to organize lexical information in terms of meanings, rather than word form, thus moving from a traditional dictionary towards a thesaurus by including semantic relations between words. The power of WordNet lies in the semantic relations that are established between words and the diversity in them.

In WordNet, word forms and word meanings are mapped similar to a dictionary. If one word form maps to more than one meaning, the form is polysemous, and if more than one form maps to the same meaning, these forms are synonymous. This can be shown as a lexical matrix where we have a mapping from word forms to word meanings and two forms having(mapped to) the same meaning are said to be synonyms and a single word having(mapped to) two meanings is said to be polysemous.
Traditional dictionaries are solely ordered alphabetically by word form. WordNet is also ordered in a taxonomical hierarchy of meanings, thus providing a significant contribution to an ontology. The concepts are specified by lexicalization, where synonymous word forms are grouped into sets defining the meaning they share. These synonym sets are called synsets and constitute the unit of meaning in WordNet.

WordNet includes the following semantic relations, [29]

- **Synonymy** is the basic relation in WordNet, because it is used to form sets of synonyms to represent word meanings. Synonymy is a symmetrical relation between word forms.

- **Antonymy** is also a symmetrical relation between word forms, especially important in organizing the meanings of adjectives and adverbs.

- **Hyponymy and Hypernymy** are the subordinate or specification relation and superordinate or generalization relation respectively.

- **Meronymy and Holonymy** are the part-whole relations

- **Entailment** is between verbs, e.g. snoring entails sleeping.
4.2 Ontology-based indexing

Ontologies can be considered as a kind of extended taxonomies that also are comprised of complex concepts. To utilize ontology, we need to somehow introduce ontology into the indexing process. The two main challenges in utilizing ontologies in information retrieval are 1) to map the information in documents and queries into the ontologies and 2) to improve retrieval by using knowledge about relations between concepts in the ontologies.

We need some kind of semantic analysis to somehow recognize concepts in the documents and map them into an ontology, and by so doing reveal the precise meaning, which is called word sense disambiguation.

In our work, we consider that the ontology is already included in the collection in the form of a taxonomy/concepts (only documents, not queries). Use-cases of such a scenario include search in E-Commerce, where there is already an inherent taxonomy within the inventory, and search in Medical Database, where documents are classified according to the topic(s) they belong/are relevant to. Note that we assume this presence of ontology only for the collection of documents and not for the query. We focus on information retrieval systems which support web-oriented search for the user to enable relevant document retrieval. So, it is our job to map the query to its place in the ontology and we explore methods to achieve this.

4.3 Ontological Similarity

In this section, we focus on ontological concept similarity, the similarity between concepts in ontologies. The foundation for establishing similarity is ontologies formed by a set of concepts interrelated by a set of semantic relations. We explain the concept of ontological similarity with the help of a popular relation, the ISA relation, which is used to describe generalization and specialization. The ISA relation is normally defined as a transitive relation, but for the rest of our report, we consider it to be non-transitive and use it to define distance between two concepts. The following sections describe some of the simple similarity measures previously explored in literature.

4.3.1 Path length Approaches

In this section, we present two simple approaches which make use of the graphical nature of an ontology. Before we go onto the approaches, we need to clarify the non-symmetric nature of ontological similarity. Common similarity measures are symmetric
Chapter 4. Ontology-based IR

Figure 4.2: The property of generalization implies that \( \text{sim}(D, B) < \text{sim}(B, D) \)

i.e. \( \text{sim}(A, B) = \text{sim}(B, A) \). If we just consider the similarity between concepts, then the order of these is not taken into account, e.g. the similarity between a "cat" and a "dog" is equal to similarity between a "dog" and a "cat". However, in a context where the order between concepts can be determined by a taxonomy or an ontology, the symmetry property becomes deceptive, e.g. the ontology in the figure implies that \( \text{sim}(D, B) < \text{sim}(B, D) \). For example, consider an ontology where plankton is a specialization of organism, then the intuition that plankton satisfies the intention of the query on organism whereas organism (which could be of any kind) does not necessarily satisfy the intention of a query on plankton, [30].

This indicates that in the ontology-based information retrieval context, the similarity measure cannot be symmetrical, and should somehow capture that the “cost” of similarity in the direction of the inclusion (generalization) should be significantly higher than the similarity in the opposite direction of the inclusion (specialization).

**Shortest Path Length** One obvious way to measure similarity in a taxonomy, given its graphical representation, is to evaluate the distance between the nodes corresponding to the items being compared, where a shorter distance implies higher similarity. The principal assumption is that the number of edges between terms in a taxonomy is a measure of conceptual distance between concepts:

\[
dist(c_i, c_j) = \text{minimal number of edges in a path from } c_i \text{ to } c_j
\]

(4.1)

Also, note that the shortest path length measure doesn’t comply with the generalization property because the measure is symmetric.

**Weighted shortest path** It is argued that concept inclusion (ISA) intuitively implies strong similarity in the opposite direction of inclusion (specialization). In addition, the direction of the inclusion (generalization) must contribute some degree of similarity, as, for example, in the small excerpt of an ontology in the figure. With reference to this
Ontology, the atomic concept dog has high similarity to the concepts poodle and alsatian. Similarity reflecting “distance” can then be measured from the path length in the graph corresponding to the ISA relation. In this approach, the similarity measure is a product of all weights along the paths between concepts $i$ and $j$ where the weights are dependent on the number of specializations and generalizations along the path.

### 4.3.2 Depth relative approaches

Despite the apparent simplicity, the edge counting approaches have a widely acknowledged problem of typically relying on edges in the taxonomy to represent uniform distances. Consider the two pairs of concepts taken from WordNet 1) “Siberian Tiger” and “Bengal Tiger” and 2) “Car” and “Scooter”. Using our intuition, we would judge the similarity for the first pair to be higher than the similarity for the second, since the first pair of concepts is much more specific.

All the approaches in this class are based on this property: *The distance represented by an edge is influenced by the depth of the location of the edge in the ontology* i.e. edges which are very deep in the ontology must have less "distance” compared to the ones which are shallow in the ontology.

### 4.4 Ontology-based IR

The most obvious solution for introducing ontological similarity in the query evaluation is to modify the descriptions of the query to include the revealed knowledge, more specifically, to expand the descriptions with similar concepts found using ontological similarity. The purpose of the expansion of descriptions is to reduce the gap between user intention and system interpretations of queries and documents. An expansion that adds relevant aspects to a description will lead to increased recall and the assumption is that this, in turn, will lead to better answers because the resulting reduced precision
can be dealt with through inherent modification mechanisms of the evaluation principle, for instance, the ranking of results.

In an ontology-based system, the idea is to map information found in documents or queries into an ontology, and in so doing, draw closer to the meaning of the information. To achieve this, we can either expand the documents, denoted *object expansion* or *query expansion*. The ”cost” of query expansion is very less (compared to *object expansion*) due to the small size of the query and therefore the processing of the query can be done easily at query time. In our work, we only deal with *query expansion*.

Past work has dealt with a new line of thought called *concept expansion*, which essentially means expansion in the concept space.

### 4.4.1 Concept Expansion

The goal of a concept expansion is to expand the interpretation of a given concept with closely related concepts in order to achieve a match on “conceptual content” rather than on specific words or concepts, as well as to compensate for the uncertainty due to ambiguous senses in natural language. Concepts are closely related when they have a high degree of similarity, which for the similarity measures in use here means concepts that are positioned closely together in the ontology with respect to distance. In the information retrieval system, this would equal the ability of the system to reveal documents judged relevant by the users.
Chapter 5

Prototype Implementation

In this chapter, we describe a prototype implementation of a very simplistic ontology-based information retrieval. The main objective of this, is to show the effectiveness of exploiting ontology within simpler models to achieve better performance. The implementation involves a very simple and easy-to-understand ontological similarity measure embedded within the popular vector space model, 1. To be specific, the system implemented is a concept-based IR system rather than an ontology-based system as we do not consider a full-fledged ontology in the collection. Instead, if each document in the collection is mapped to a set of concepts $C_d$, which is a subset of the overall set of concepts $C$, then that is sufficient for our implementation. Any semantic relations, such as generalization and specialization, within these concepts are not exploited in this simplistic system.

5.1 Description of the system

In this system, each resource in the collection (documents) are linked to concepts from an ontology (or taxonomy). Contrastingly, the query is given by the user without any relevant concepts and it is the job of the system to assign the required relevant concepts to the query before running it on the ontology-based system. There are several approaches that can be explored in this area such as using an external knowledge source, such as WordNet, to extract concepts which are synonyms to the query terms. But these approaches have a significant drawback as we are withdrawing from domain-specific knowledge that ontology brings to the table. The effect is even more compounded when query terms exhibit polysemy and term concept disambiguation would be needed.
Chapter 5. Prototype Implementation

This ontology-based approach is built on-top of the vector space model approach. To reiterate the similarity measure used in the vector space model approach:

\[
\text{sim}_{VS}(d, q) = \frac{w^d \cdot Q}{|w^d| \cdot |Q|} \tag{5.1}
\]

where \(w^d\) is the index term weight vector of the document \(d\) and \(Q\) is the index term weight vector of the query \(q\).

### 5.2 Ontological Similarity

In this system, as stated before, we assume that the concepts related to each document in the collection is already given. We extract concepts from the query using a "pseudo-relevance feedback"-like approach, which is described in the section. For now let’s assume that we also have the concepts related to the given query. Let \(Q_{con}\) be the set of concepts related to the query \(q\) and \(D_{con}\) be the set of concepts related to the document \(d\). Given this data, we calculate the similarity of the document \(d\) and the query \(q\), in terms of its concepts in the following way, [31]

\[
\text{sim}_{ONTO}(d, q) = \begin{cases} 
|Q_{con} \cap D_{con}| & \text{if } |Q_{con} \cap D_{con}| \neq 0 \\
 k & \text{otherwise}
\end{cases} \tag{5.2}
\]

where \(k\) is an empirically determined small constant (like 0.1) that is used as the similarity measure between a query and a document which do not share any concepts.

### 5.3 Combined Similarity Measure

In this ontology-based system, we combine the above two similarity measures to get the best of both worlds i.e. the domain-level knowledge (or semantic) brought by the ontology and the keyword similarity brought by the vector space model approach. The final similarity measure can be expressed as:

\[
\text{sim}(d, q) = \text{sim}_{VS}(d, q) \ast \text{sim}_{ONTO}(d, q) \tag{5.3}
\]

The above similarity measure is used to rank the documents in the final retrieval. Observe that the ontological similarity enhances those documents which share concepts with the query in addition to keywords (expressed by the vector space model approach).
The more concepts they share, the more their similarity is enhanced/magnified. Contrastingly, if a document and the query share keywords but don’t share any concepts, then the factor $k$ makes sure that such a document is repressed in the ranking. In other words, the similarity of such a document with the query is decreased to represent its dissimilarity with the query in the concept space.

5.4 Extraction of concepts from query

In the above approach, an essential stage is the extraction of relevant concepts for the given query. It is quite essential as the performance of the system is greatly dependent on it. There are several approaches that can be taken, including but not limited to, using an external knowledge source such as WordNet to find concepts which are synonyms to the query terms. The major drawback with this approach is the fact that such one-to-one associations (query term to concept) will lead to concepts which may not be relevant to the whole query (just relevant to a single term). To avoid this, we need to consider the entire query and extract relevant concepts.

One of the approaches that deal with the query in its entirety, is pseudo-relevance feedback. Although, it is mainly used in query expansion, we can employ it here in our scenario to find relevant concepts for a given query. Since, each document in the collection already has concepts associated to it, we can exploit the relevance relation between query and documents to extract concepts for the query itself.

As is usually the case in PRF methods, we first run the query on the base retrieval model, i.e. the Vector space model, in our case. Among the retrieved documents we consider the top $l$ documents ($l$ is empirically determined) and extract $m$ most frequently associated concepts with these $l$ documents. We then associate these $m$ concepts to the query itself. The justification is that, since the base method has claimed that the top retrieved documents are relevant to the query, we can make use of that fact and assume that concepts of a query and a document which have high similarity are also very similar.

Upon obtaining the $m$ concepts for the query, we now run the query on our ontology-based system and retrieve documents. The advantages of this method include a) no manual concept assignment needed for the query (which is obviously, very expensive for large number of queries) b) we have an automated way of assigning topics which is essentially relying on the base approach. For example, for a query $q$, the VS model retrieves a document $d$ which contains similar terms as $q$. Now, when you assign concepts of $d$ to $q$ and run it on the ontology-based system, then we may obtain a new document $d'$ which may share very less terms with $q$ but is still relevant to it (in the concept space).
The major drawback of this approach is that, like most PRF-based approaches, the performance of the system is heavily dependent on the initial retrieval. If the initial retrieval has poor performance, then the PRF-based approach would also have a poor performance, albeit a slightly better one.
Chapter 6

Experimentation and Results

In this chapter, we discuss the experiments done on the prototype implementation described in the previous chapter. We present some initial results, which serve to indicate that ontology is a great avenue to explore for improving retrieval performance, as it contains domain-level knowledge that is lacking in most retrieval models.

6.1 Description of the data used

In this experiment, we used the Cystic Fibrosis collection from Medline data, that is publicly available. It contains a collection of 1239 files, where the minimum size of the file is 0.12Kb, maximum is 3.8Kb and the average size is 1.05Kb.

The collection also contains a set of 100 queries with a set of relevant documents assigned to each of these queries, as judged by specialists in the field. Each document in the collection already has concepts associated with it. The total number of concepts in the collection is 821 and the average number of concepts assigned to a document is 2.8.

6.2 Implementation details

Since we are dealing with a small sized data collection, we built a small-scale IR system from scratch. The implementation consists of functions and classes involved in constructing TF-IDF weights, and term-by-document matrices which are essential for the vector space model approach. Although, if we tackle very large data collections, we might be forced to use publicly available scalable libraries such as Lucene to index the collection.
Several data pre-processing steps such as tokenization, stop word removal, stemmer (Porter stemmer) are conducted on the terms of each document in the collection before they are added into the index.

6.3 Evaluation

6.3.1 Groundtruth Approach

In the evaluation of the system we also consider a groundtruth approach, which is similar to manual assignment of concepts to the query. In this approach, since we already know the documents which are relevant to the query, we extract the top $m$ most frequent topics among the relevant documents and assign it to the query. Although, this is not feasible in a real setting, the performance of this approach shows the maximum improvement that can be obtained in the performance, if we exploit ontology properly.

6.3.2 Parameters

In our evaluation, we use a value of 5 for $l$ where $l$ is the number of top retrieved documents from which concepts are extracted. We set $m$ to 5, where $m$ is the number of concepts assigned to a query.

6.3.3 Results

The results obtained can be seen in the precision versus recall curve given for a single query comparing different approaches. We also draw precision and recall versus number of documents retrieved curves, averaged over all 100 queries, which show the performance of different approaches in precision and recall respectively.

6.4 Observations

If you observe the precision versus recall curve, you observe that the groundtruth approach beats both the other approaches by a huge margin. This just shows how much potential ontology has to improve the performance of retrieval systems and provides justification for our line of work. Another thing to observe is that the PRF approach is only slightly better than the vector space model approach and that is because, as mentioned before, the performance of PRF-based approaches depend heavily on the initial
Chapter 6. Experimentation and Results

Figure 6.1: Precision vs Recall curve

Figure 6.2: Precision curve

Figure 6.3: Recall curve
retrieval. If the initial retrieval, in our case vector space model retrieval, is poor then the performance of PRF-based approach will also be poor (but slightly better).

Observing the precision curve, we see that both the groundtruth and PRF-based approach are better than the vector space model approach. Interesting thing to note is that, it looks like the precision of the groundtruth based approach is decreasing with increasing number of retrieved documents. This does not indicate that ontology doesn’t scale well to large number of documents but rather that in the current data, the number of relevant documents for a query are quite less hence as the number of retrieved documents increase, the precision decreases.

The recall curve clearly indicates the trend that both the groundtruth and the PRF-based approach are better in recall when compared to the base vector space model approach.
Chapter 7

Conclusion and Future Work

7.1 Summary and Conclusion

We started out with a basic introduction to Information retrieval and the important steps in the IR pipeline. We discussed different ways of pre-processing the data to make it more "indexable". In this, we saw stages such as tokenization, stopword removal and stemming of words. Further, we saw some traditional retrieval models which have stood through the test of time and are still relevant today. In this section, we mainly looked at three models, namely, Boolean Model, Vector Space Model and the Probabilistic Model. We then looked at metrics that are used to evaluate a retrieval system’s performance.

We then set out to lay the motivation for our work and defined the problem rather informally. We described our goal of embedding ontology into the information retrieval framework to bolster the performance of an IR system. In addition to our goal of a standalone ontology-based information retrieval system, we also expressed our goal of exploring different ways ontology can be introduced to the query expansion framework.

In the following chapter, we discussed different query expansion methods that have been proposed over the years. We then proceeded to explain the query expansion framework and the stages involved in the pipeline. We generalized several approaches that have been used in the past for each of these stages. Starting from pre-processing, we traced the pipeline to generation and ranking of candidate expansion terms. After obtaining the candidate expansion terms, we need to select only a few of them. To achieve this, we discussed several approaches that were established by past literature. Finally, we reformulate the query using the expansion terms so that the new query has a better performance than the original query. Following this, we classified different query expansion approaches into broad categories based on the techniques that they use.
We followed the literature survey of query expansion with a brief literature survey of Ontology and its use in IR. We explained how WordNet can be seen as an ontology and described its underlying structure. We then proceeded to Ontology-based indexing or more specifically, on how to introduce ontology into the indexing process. We followed it up by different ontological similarity measures that have been used in the past. We ended our discussion on Concept expansion, a method to use ontology within an IR framework.

A prototype implementation, to see first-hand the improvement ontology can bring into an IR system, was described in the next chapter. We described the system a bit briefly followed by the ontological similarity measure that is used. We explained how we will introduce this similarity measure to the already existing similarity measure from the keyword-based retrieval system. We ended the chapter with an approach to extract concepts from a query automatically without human intervention.

The next chapter dealt with the results obtained from the prototype implementation. We showed how ontology can result in drastic improvement (through the groundtruth case) in the performance of an IR system. We also showed how our approach based on PRF, performs slightly better than the original keyword-based retrieval. We listed out several observations that we can glean from the results at the end of this chapter.

### 7.2 Future Work

This report details the work that has been done in the stage I of the BTech Project. Our plans for the stage II include, but not limited to, devising a novel ontological similarity measure based on the concepts and the semantic relations between them (something that the prototype implementation did not exploit). As we have described before, we will also set out to explore different ways in which we can use ontology to improve the performance of query expansion methods, by somehow exploiting the context and domain-level knowledge that ontology has.

On a parallel track, we will also build a large-scale IR system on which we can conduct evaluations for our proposed model. An important question that we seek to answer is, can ontology be used to improve performance even in large scale systems? This question can only be answered by devising a scalable ontology-based IR system and then evaluating it against traditional retrieval models.
Bibliography


