IMPROVEMENT OF CORPUS-BASED SEMANTIC WORD SIMILARITY USING VECTOR SPACE MODEL

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCE
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

YUNUS EMRE ESİN

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

JULY 2009

Approval of the thesis:

IMPROVEMENT OF CORPUS-BASED SEMANTIC WORD SIMILARITY USING VECTOR SPACE MODEL

submitted by **YUNUS EMRE ESİN** in partial fulfillment of the requirements for the degree of **Master of Science in Computer Engineering Department, Middle East Technical University** by,

Prof. Dr. Canan Özgen
Dean, Graduate School of **Natural and Applied Sciences** _____

Prof. Dr. Müslim Bozyiğit
Head of Department, **Computer Engineering** _____

Assoc. Prof. Ferda Nur Alpaslan
Supervisor, **Computer Engineering Department** _____

Examining Committee Members:

Assoc. Prof. Dr. Nihan Kesim Çiçekli
Computer Engineering Dept., METU _____

Assoc. Prof. Dr. Ferda Nur Alpaslan
Computer Engineering Dept., METU _____

Assoc. Prof. Dr. Ali Doğru
Computer Engineering Dept., METU _____

Asst. Prof. Dr. Pınar Şenkul
Computer Engineering Dept., METU _____

Özgür Alan, M.Sc.
Computer Engineer, Orbim Ltd. Şti. _____

Date: _____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name: YUNUS EMRE ESİN

Signature :

ABSTRACT

IMPROVEMENT OF CORPUS-BASED SEMANTIC WORD SIMILARITY USING VECTOR SPACE MODEL

Esin, Yunus Emre
M.S., Department of Computer Engineering
Supervisor : Assoc. Prof. Ferda Nur Alpaslan

July 2009, 68 pages

This study presents a new approach for finding semantically similar words from corpora using window based context methods. Previous studies mainly concentrate on either finding new combination of distance-weight measurement methods or proposing new context methods. The main difference of this new approach is that this study reprocesses the outputs of the existing methods to update the representation of related word vectors used for measuring semantic distance between words, to improve the results further. Moreover, this novel technique provides a solution to the ***data sparseness of vectors*** which is a common problem in methods which uses vector space model.

The main advantage of this new approach is that it is applicable to many of the existing word similarity methods using the vector space model. The other and the most important advantage of this approach is that it improves the performance of some of these existing word similarity measuring methods.

Keywords: semantic word similarity, vector space model, WordNet, text corpus, NLP

ÖZ

VEKTÖR UZAYI MODELİNİ KULLANARAK METİN YIĞINI KÜMESİ TABANLI MANTIKSAL KELİME BENZERLİĞİNDE İYİLEŞTİRME

Esin, Yunus Emre
Yüksek Lisans, Bilgisayar Mühendisliği Bölümü
Tez Yöneticisi : Doç. Dr. Ferda Nur Alpaslan

Temmuz 2009, 68 sayfa

Bu çalışma büyük kelime yığınlarından mantıksal olarak birbirine benzer kelimeleri bulmada yeni bir yaklaşım sunmaktadır. Bu konudaki daha önceki çalışmalar ya yeni kelimeler arası uzaklık ölçümü metodlarının geliştirilmesine ya da yeni çerçeve yöntemlerinin keşfedilmesine odaklandılar. Bu çalışmadaki yaklaşımın önceki araştırmalardan ana farkı varolan metodlardan elde edilen sonuçların geri besleme ile tekrar işleme sokulmasıdır. Ayrıca bu yöntem vector uzayı modellerinde genel bir sorun olan ***vektörlerdeki veri seyrekliği*** problemine de bir çözüm oluşturmaktadır.

Önerilen metod şu aşamalardan oluşmaktadır: Birinci aşama kelimelerin bilinen yöntemlerle belirli sayıdaki benzerlerinin bulunması, ikinci aşama bulunan benzer kelimelerin belli oranlarda sisteme geri besleme olarak verilmesi, üçüncü ve son aşama ise güncellenmiş sistemin tekrar işleme sokulmasıdır. Böylece kelimeleri gösteren vektörler, bir önceki bulunan sonuçlarla güncellenerek sonuçlar iyileştirilir. Bu çalışmanın önemli bir avantajı, vektör-uzayı modelini kullanarak mantıksal olarak benzer kelimeleri bulan metodlara uygulanabilir olması, diğer bir avantajı da bilinen kelime benzerliği alanında kullanılan metodların başarısını artırmasıdır.

Anahtar Kelimeler: anlamsal kelime benzerliği, vektör uzayı modeli, WordNet, kelime yığını kümesi, doğal dil işleme

To my family

ACKNOWLEDGMENTS

I would like to express my sincere gratitude and appreciation to Assoc. Prof. Ferda Nur Alpaslan and Özgür Alan for their encouragement and support throughout this study.

I am deeply grateful to my family for their love and support. Without them, this work could not have been completed.

I am also grateful to my dear friends Ali Yaşar Yiğit and Süleyman Eren Doğan for their support and motivation.

I would thank the Scientific and Technological Research Council of Turkey (TÜBİTAK) for providing the financial support for this study.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

CHAPTER 1

INTRODUCTION

1.1 MOTIVATION

Starting from the late twentieth century *science* and *communication* are becoming increasingly popular. Actually they are the two most important actors that shape the life of the human beings nowadays. When concentrated on these two actors *word* and *computing* terms can be found in the intersection area. ***Words*** are the main building blocks of *communication* between the human beings and widely used in *science*. ***Computing*** is the main activity which is involved in both *science* and *communication* applications. When the actual meaning of the words can be automatically discovered by computers then new applications might be developed which really address the needs of these eras. In order to achieve this one of the mile stone is the discovery of semantically similar words automatically using facilities of computing.

Determining the actual meaning of a word in a sentence is sometimes confusing for human beings because of the ambiguity of the word itself or the usage of the word in a sentence. In such a case, a well-known similar word may help to understand the meaning easily. Moreover, one of the most important steps of learning a new language is learning the synonyms of the most common words. For these purposes, in the history, people tried to create synonym lists of such words manually and in schools students try to memorize them. One of the oldest and most popular of such a list is ”Thesaurus of English Words and Phrases” published in 1852. After then people created several such kind of resources especially for English language.

”Language is a living organism” which means that it is not only as a set of words that is used to communicate among people but also it is born, started growing with time, in some cases gives birth to different dialects or experiences death [1]. Therefore at anytime a new

word can join or leave the language. Moreover as time passes or as the conditions change, meaning of words may also change. Because of these reasons not only the creation but also the maintenance of such lists is very difficult. One solution to this problem might be unsupervised and automated creation of such lists by using the abilities of computers.

As the technology evolves, researchers try to answer the question *"How this task can be handled on computers"*. Actually some of the subfields of Computer Science mainly aim at answering this question. Especially for the Natural Language Processing (NLP) finding semantic similarities between words is a high priority research area. Besides the conducted several studies in this area, there are many on going researches exist in many of universities from all over the world. Moreover, there are also much related studies, which use the benefits of the studies on semantic word similarity, exist such as Word Sense Disambiguation (WSD). Therefore ***semantic word similarity*** is a hot topic and it has many application areas. In addition to these, studies on semantic word similarity are conducted or supported by some of the well known Information Technology (IT) companies. In some developed countries government also supports such kind of studies.

Considering the importance of the problem we first start working on exploring related studies in the literature. After a deep analysis, we implemented several promising approaches and thought on possible enhancements. After brain storming on this issue we came up with a novel idea. After implementation we conducted large number of experiments and analyzed the results in detail. According to the results our approach has promising results.

1.2 SEMANTIC WORD SIMILARITY

With this study we try to answer the question *"How semantically similar words of a headword can be discovered on computers automatically?"*. Our study is based on existing unsupervised approaches to this problem. However, we also propose a novel unsupervised approach on semantic word similarity which has promising results. In this study, we made used of word statistics and word patters of large text corpora. We propose a feed-forward mechanisms, which is similar to the working principles of neurons in Artificial Neural Networks (ANN), to update the headword vectors in the vector space. In this new technique, the headword vectors are updated using the found semantically similar words of the related headwords vectors.

According to the results of large number of experiments we improved the performance of existing methods in significant amounts. Moreover, this new approach also provides a solution to vector data sparseness problem, which is considered as a common problem for Statistical Corpus based approaches.

According to the detailed analysis of the experiments, the most curial enhancement of this novel reprocess approach obtains better results in terms of semantic word similarity, even with small corpora and rare headwords. Moreover, this new approach increases the probability of finding similar words at first positions in the found semantically similar words list.

In brief, the are three main advantages of the proposed novel approach. They are:

- It is applicable to many of the existing word similarity methods using the vector space model
- This approach improves the performance of some of these existing word similarity measuring methods
- It is flexible and open for future enhancements

1.3 ORGANIZATION OF THE THESIS

Chapter 2 gives a brief information on existing studies in the literature related with semantic word similarity. Chapter 3 explains the details of the methods (*window context method, vector space model, distance measure functions, weight functions*) which are used for finding semantically similar words of a headword from large collection of texts. Chapter 4 gives the details of the proposed novel approach and the conducted experiments. Chapter 5 discusses the experimental results of the proposed method in general and in detailed basis. Finally, chapter 6 includes a summary of the study and mentions about possible future improvements.

CHAPTER 2

LITERATURE SURVEY

This chapter contains information about the existing related works in the literature conducted by researchers so far. These studies are categorized in three related domains. They are listed as below:

- Word similarity studies in the literature
- Application areas of word similarity
- Testing similarities of the words

The studies found in literature related with these domains is discussed in the following separate sections.

2.1 WORD SIMILARITY STUDIES IN THE LITERATURE

There exist several studies on computer-based word similarity in the literature. These studies are grouped into four categories ([2]). They are:

1. Electronic collection of the human lexical background about words
2. Using existing electronic collections to find semantic relations between words
3. Using statistical information of large text corpus to find semantic relations between words

4. Hybrid methods which use both statistical information of corpus and electronic collections

Existing studies in the literature of each four item is given in the following subsections.

2.1.1 Electronic Collection of the Human Lexical Background about Words

The most important studies on the first category use large lexical databases such as WordNet [3] which contains combinations of the collected human made dictionaries, thesauri, and some other kind of similar data sources. Controlled vocabularies, which are more organized way of thesauri and taxonomies of the word and describe every concept in a domain, can also be classified in this category. Library of Congress Subject Headings (LCSH) can be given as an example ([4]). In addition to these, current dictionaries and thesaurus, which are prepared in many languages, can also be counted as in this category.

2.1.2 Using Existing Electronic Collections to Find Semantic Relations Between Words

Second category involves the studies such as [5] which uses WordNet's conceptual hierarchy in order to measure the similarities of the words and the work in [6] proposing a system of measuring text similarity by integrating the meaning of texts into the similarity measure. In the study [7] semantic similarity is measured between words using existing web search engines. Actually this study can be categorized in the third category. However, since which sources are used from search engines cannot be defined easily and in the study search engines used as if existing electronic collections to find semantic relations between words, this category is more suitable for this article. The study in [8] is similar to the previous one, the main idea in both of them is the same. In [8] authors present two novel web-based similarity metrics for computation of semantic similarity between any words. One of which considers the number of pages returned from a web search query. The other one considers top ranked documents and, by using ”wide-context” and ”narrow-context” metrics, it provides a promising word semantic similarity measurement.

2.1.3 Using Statistical Information of Large Text Corpora to Find Semantic Relations Between Words

For the third category, one of the earliest studies is by Lin ([9]). He is one of the first researchers working on computer-based word similarity and his studies regarded as the starting point of most of the studies in the literature in this category. His work determines a new similarity measure method and presents new evaluation methodology for automatically created thesaurus. The work in [10] not only evaluates existing approaches but also proposes new promising methods. It is one of the most detailed study done in the area of word similarity so far. In this study, the success of 5 different context methods, 13 different semantic difference measure methods, and 13 different weight methods are evaluated with several text corpora (such as [11] and [12]). One recent study ([13]) works on very huge collection (terabyte-sized) of web data and compares the existing two similarity measure methods. In [14] a new method is introduced which combines word-to-word similarity metrics into text-to-text metrics and authors of the article claim that this new method outperforms the traditional text similarity metrics based on lexical matching. The same authors and Carlo Strapparava conducted another recent work ([15]). This study concentrates on the semantic similarity of texts, instead of individual words. Moreover, this work gives importance to pre-processing of large text corpora. Their findings for improving the quality of results are considered in this thesis.

A very recent study ([16]) also concentrates on similarity of texts using a corpus-based measure of semantic word similarity and uses modified version of the Longest Common Subsequence (LCS) string matching algorithm to further improve the results. Another study ([17]) evaluates 14 different existing text similarity measures and compares the success of each. [18] is an article on the area of Data Mining. By using large text collections (text corpora) authors have developed a model based on nonparametric Bayesian modeling for automatic discovery of semantic relationships between words. This article explains a new approach on knowledge discovery of semantic relationships between words. Contrary to the other works in this third category (which use statistical information gained from the text corpus), the approach in the article is based on probabilistic generative model. This is why Bayesian Graph Model has been used. An old study ([19]) can be considered related with the work done in [18] because both make the use of probabilities. Distributional similarity measures are studied for the purpose of improving probability estimation for unseen co-occurrences. In addition to this, well

known similarity distance measurement functions are evaluated. [20] not only compares the *Cosine Similarity* and *Jaccard Coefficient*, which are the two famous vector based similarity measures commonly used in the field of Information Retrieval (IR), but also proposes a new measurement technique. In this study, domain-specific ontologies are compared and mapped by using these corpus-based semantic similarity measures. In this thesis, we first used this new measurement approach. However, we could not get promising results from the evaluations, therefore the results of this new measure method is ignored. [21] is a related work in which a novel word co-occurrence model based on an ontology representation of word sense, is presented. This paper can also be categorized in the application areas of word similarity. The difference of the study [22] from the others is that semantic relatedness is computed with similar methods but using Wikipedia-based ESA (Explicit Semantic Analysis). This is a new method which represents the meaning of texts in high dimensional space of concepts. Finally, [23] and [24] are the two related study. The former one is a study in the domain of Chinese in which study syntactic related co-occurrences are used as context vectors. The latter one investigates the influence of the type of context features on the kind of semantic information to be extracted. Findings of the latter one is important and very related with our study. It discusses window size and how it affects the word relatedness. Moreover, a distinction between semantic similarity and semantic relatedness has been made and this is useful when evaluating the results in terms of these two factors.

Some crucial studies has been conducted on Latent Semantic Analysis (LSA). This is a theory and method used for extracting and representing the contextual-usage meaning of words by using statistical information of corpora [25]. The underlying idea in LSA is similar to the idea in most of the studies in the third category. Moreover, there are lots of articles and useful information about LSA available on the web [26].

In some of the studies on this area the factors that affect the quality of the results are considered. For example in [27] a systematic exploration of the principal computational possibilities for formulating and validating representations of word meanings using statistical information from corpus word co-occurrence statistic is presented. Moreover, in this study previous work is also evaluated and (a common evaluation method) ratio of correct answers to the TOEFL questions is used for defining the success of approaches. Another example is [28] where mainly which factors affect the success of context-methods are discussed. A simple and efficient system for computing word statistics is proposed. Again TOEFL tests are used for the

evaluation of the study.

2.1.4 Hybrid Methods Which Use Both Statistical Information of Corpus and Electronic Collections

An instance of this (the fourth) category [29] concentrates on measuring semantic similarity distance between words and concepts. They use both corpus statistics and WordNet hierarchical structure (such as path length between two nodes and distance to the first common parent). One result of this study is that if the occurrence of a word increases in a corpus, then its informativeness decreases. In our work, we use this information as a parameter. Depth, average density, and the link/relation type factor are considered in [29]. An old study ([30]) introduces a system called SEXTANT (Semantic EXtraction from Text via Analyzed Networks of Terms). In this system dictionaries and corpus statistical methods are used. SEXTANT uses fine-grained syntactically derived contexts and it produces results to judge the word similarities. The work in [31] employs the third category studies to find word similarities and use them to improve search quality for search engines in general and specific domains. A recent study ([32]) introduces a supervised corpus-based machine learning algorithm for classifying analogous word pairs. In this study, subsumption of synonyms, antonyms, and associations is analyzed. The success of the approach is evaluated in SAT analogy questions, TOEFL synonym questions, ESL synonym-antonym questions, and similar-associated-both questions from cognitive psychology. Actually testing the success of the approaches on tests prepared for human beings is very common in literature. [33] is a study on the comparison of WordNet and distributional similarity-based approaches, and it is useful for considering the parameters that affects the success of hybrid methods.

2.1.5 Our Approach on Semantic Word Similarity

The work proposed in this thesis can be classified as an instance of the third category. In this approach (details are explained in chapter 4), we use statistical word similarity measures on very large text corpora. Window method is used for constructing contexts of headwords and vector space model for representing these contexts. The starting point is similar to the studies done in the third category. We use existing methods and find similar words of the same predefined headwords. Then using these found words, we update the vectors of the related

headwords in vector space and after reprocessing the updated vector space we find new similar word-lists. As the extensive experimental evaluation on RCV1 ([11]) results shows, this new approach produces better word-lists in terms of semantic word similarity compared to the first found lists [2].

2.2 APPLICATION AREAS OF WORD SIMILARITY

Words are the main building blocks of the communication among human beings. Therefore they are very important in a human life. As the technology evolves, words are being used in many areas especially in computers. Nowadays, words are used in electronic documents, dictionaries, thesaurus and especially in World Wide Web (WWW). [34] is a study about application areas of word similarity. In this study, a method for computation of the two short texts based on the similarities of their words is proposed. The application areas listed in the study are designing exercises for second language-learning, acquisition of domain-specific text corpus, information rerieval, and text categorization.

In this section, possible application areas which are both very common and hot-research topics in Computer Science are given. They are grouped in the following categories.

- Word Sense Disambiguation
- Search Engines
- Similar Studies in Different Domains

2.2.1 Word Sense Disambiguation

WSD (Word Sense Disambiguation) is a hot topic in the field of Computer Science and it is very related with word similarity. Since some words have many senses, sometimes defining which sense of the word is used in a sentence automatically is difficult. WSD studies try to solve this problem.

There are many articles available which propose solutions to the WSD problem with word similarity. In many of these studies, the main idea is finding the most similar word to the ambiguous word in order to discover which sense of the word is used in the related sentence.

One of the oldest studies in WSD related with word similarity is [35] and it presents an unsupervised learning algorithm for sense disambiguation. [36] is one of the first milestone of the studies in this area. In this study WSD problem is solved with word similarities. One important study, which is also another milestone, is the Ph. D. thesis in [37]. This study contains detailed analysis of existing approaches on corpus-based methods. In [38] word to be disambiguated is assigned the sense that it is mostly related to the senses of its neighboring words. In the study [39] using WordNet, a probabilistic approach is presented for discovering the actual meaning of the target word. Besides the WordNet, text corpus is also used in this process. [40] is an important study which is much related with WSD and word similarity. It describes an unsupervised graph-based method for WSD, and presents comparative evaluations using several measures of word semantic similarity. The study in [41] is very similar to the [40]. It presents an unsupervised WordNet-based WSD system using the relatedness of the words. A very recent study ([42]) presents a fully unsupervised word sense disambiguation method that requires only a dictionary and text corpora.

[43], [44], [45], [46], [47] and [48] are the other related works found in the literature which are very related with semantic word similarity and WSD. Moreover, they provide solutions to the problems in different application areas.

From all these studies, WSD and semantic word similarity are very related and inter-connected problems. A solution to one of them can be used for the other one.

2.2.2 Search Engines

As the internet becomes available to many people, World Wide Web (WWW) is used very frequently. According to [49], there are about 8 billions of web pages exist and this number increases very rapidly. Since there are too many pages in the web, memorizing the whole web page addresses that are important is unrealistic. In order to solve this problem web search engines are prepared, which frequently index the whole URLs of pages in WWW and in case of a query try to find the best matches and return the found URLs to the user. Most of the popular search engines try to find the exact words or their stemmed versions when a query is entered by a user.

One problem is that as the web is growing in an incredible speed, this kind of searches will

not be adequate, because, in case of an ordinary search, too many pages will be returned and many of which will not be the one that will be looked for. A solution seems to be the semantic search engines which do not try to find the exact words that is queried but the semantics of the words in the query. In these type of search engines if a meaning of word is ambiguous in the query, first engine try to solve the meaning and as stated in section 2.2.1, semantic word similarity is widely used on WSD problem.

Most of the new generation web search engines measure the semantic distance between the words in the query and the words in the possible relevant pages, then put the most relevant one on the top of the returned lists. An important study in this area is [50]. In this study, search engine can be configured both as an exact-match and semantic-similarity enabled searches. An XXL search engine is produced and it aims to exploit similarity conditions on element names and contents (which can be evaluated much more efficiently). Ontology based word similarity measures has been used to find exact semantically similar words, words are first disambiguated using otologies (as in 2.2.1 disambiguation is also a popular application area). [51] worked on similarities on text documents to a query due to a search engine and try to discover the similarity computations of search engines. In this study, dot-product (which is a common similarity measurement method) similarity function has been used. The work in [31] used corpus-based word statistics methods to find word similarities and these were used for the improvement of search quality in general and specific domains.

Another problem is that there are many search engines and some have advantages over others. A combination of such engines (combination of advantages to create an optimal search engine) is a solution to this problem. There are several studies in this topic and semantic word similarity studies can be used on these search engines. In the study, [52] a new open source search engine named ”Helios” is introduced. It runs on the top of 18 search engines (in Web, Books, News, and Academic publication domains). Moreover, it is flexible and new search engines can be easily plugged into the system.

2.2.3 Similar Studies in Different Domains

There are also some related work such as [53]. As the quantity of text, which is available for training and processing, increases rapidly, it is now necessary to consider developing new frameworks. In study [53], the requirements, initial design, and exploratory implementation

of a high performance NLP infrastructure are described. [54] presents an information about theoretic definition of similarity that is applicable if there is a probabilistic model, so this work shows how this definition is used in different domains in terms of measuring similarity. A different related study is [55] where authors presents an unsupervised approach to extract semantic networks from large text corpora. In [56], a similar unsupervised method, which is widely used in semantic word similarity, is applied to discover inference rules from text in the domain of question answering systems. Authors claim that this approach discovers many inference rules that are easily missed by humans. A new similarity measure is proposed in [57] which is used in data mining and may also be used in measuring word similarity. [58] can also be categorized in this section where different similarity measures are evaluated in the Maximum Marginal Relevance (MMR) framework for meeting summarization on the ICSI meeting corpus. In [59], sentence similarity is used and this work presents a new framework for representing the meaning of phrases and sentences.

As all these studies shows that semantic similarity of words or texts is involved in many of the recent studies conducted from the researchers and this makes it very important and hot research topic in the field of Computer Science.

2.3 TESTING SIMILARITIES OF WORDS

Two common methods are used for the success of word similarity methods proposed in the literature. One of them is the ratio of correctly answering test questions. One and the most important of these are TOEFL synonym questions. This method can be used for testing the quality of the all four categories of approaches in the literature of semantic word similarity. [60] shows the success rates of some of the studies conducted so far.

The other method is based on human-made thesaurus, dictionaries, and so on. The most important example of such a source is [3]. [61] is an open source module which implements a variety of semantic similarity and relatedness measures based on WordNet. In this thesis, JAVA version of this module ([62]) was used to evaluate the results. We used two similarity measure methods one of which was developed by Jiang and Conrath ([29]) and the other was developed by Lin ([54]).

CHAPTER 3

MEASURING WORD SIMILARITY

In this study ***word similarity*** is always used with the prefixed word ***semantic***. This is important because if ***word similarity*** is used without the prefixed word ***semantic*** it most probably misunderstood as ***syntactical word similarity***. In this thesis we focus on the ***meaning*** **(*semantic*)** not the ***syntax*** of the words and try to find words which have similar meanings.

One of the common entry points for measuring semantic similarity between words is using word statistics of text corpora which are collections of large set of texts from various resources. This is one of the most popular automated approaches on semantic word similarity researches. Even if these kind of text collections may have noise and contain irrelevant data (such as figures, tables, or unnecessary information in terms of word similarity), it is still possible to retrieve useful information for semantic word similarity studies. This is probable because these text collections are large enough and contains many words. Using the statistics or word patterns, text corpora can be used in semantic word similarity researches. Because of the common usage of computers among people, availability of storing large scale of data and computers with high processing power, corpus-based approaches becoming popular in these researches.

The simple idea behind corpus-based semantic word similarity measuring methods is that ***similar words have similar contexts or similar words have similar words nearby***. This means similar words tend to have near similar words. Actually this is the main idea of several studies which make use of text corpora for semantic word similarity in the literature.

For defining the nearby words of a headword, several context methods have been proposed and each have advantages and disadvantages. Decision of which context method to be used is the first step of semantic word similarity studies. After determining which context method to

be used, the next step is choosing the model where the contexts of each word will be located. For the context, we have chosen ***window method*** and for the model we have selected ***vector space model*** in this study. After these decisions, the last step is defining similarity measure methods and their parameters.

3.1 CONTEXT

In this study the term ***context*** is used as *"discourse that surrounds a language unit and helps to determine its interpretation"*, which is the first sense of this term in the WordNet [3]. Here *language unit* represents the term *word* in text corpora.

Several context methods have been proposed in the researches about word similarity. In [10] and [31] some of the existing context methods are evaluated. According to [10], ***window methods, Cass, Minipar*** and ***RASP*** are the known and important approaches used for defining context of a word. Moreover new context approaches are presented in [10] and [31]. Some of these methods have better results in semantic word similarity applications but, at the same time, they are very complex both in terms of implementation and processing. These kind of methods requires too much processing time and resources. On the other hand some other methods are easy to implement and requires relatively less computer resources but have poor results. Actually most of the time, also there are some exceptions, there is an direct relation between necessary requirements and quality of results.

Window context method was used in this thesis. The reason of using this method is that, according to the findings of [10], the window context method is one of the least sophisticated context method but at the same time it has very promising results. Moreover this method is the most commonly used method in the literature. Since it is not very complicated window based contexts of headwords can be found without much resource compared to other context methods. In addition to its low complexity, this method can also be applied to text corpora of any language.

Simply window methods are the methods that define the context of the headword in terms of the neighboring words within a limited distance. A fixed number of word width sliding window with respect to the headword is used to collect words that often occur with this headword.

There are four parameter affecting window of this context method:

Width: Number of words does the window cover in each side of the headword.

Boundaries: Whether the boundaries of the sentence, paragraph or document will be preserved or not (i.e. if the boundary exist within the window, then the words other side of the boundary is ignored in computations).

Direction: Whether the direction information of the neighbor word will be kept or not.

Position: Whether the position information of the neighbor word will be kept or not.

We defined the context definition in a similar way with [10]. A Context is a tuple *(w, r, w')* where *w* is a headword occurring in some relation type *r*, with another word *w'* in one or more sentences. Each occurrence extracted from pre-processed text is an instance of a context. We refer to the tuple *(r, w')*, as an attribute of *w* and *r*, describes the particular relationship between two words. *W* represents window, L_n represents n^{th} nearest left word and R_n represents n^{th} nearest right word of the given headword.

For example $W(L_2R_1)$ is a window of three_words size and it covers two nearest left words and the nearest right word of the given headword. There is one more sign *asterix* (*) which is used if the positions of the words are not stored. With $W(L_1R_1*)$ the window covers the nearest left and right words of the headword but not stores the position information.

Simple example given below shows the usage of the symbols defined.

Let the following sentence is given.

"The color of your dreams may depend on what sort of television your family owned."

For the window method $W(L_1R_1)$ window will cover the nearest left and right word of the headword *dreams* which are *your* and *may*. Since $W(L_1R_1)$ doesn't contain *asterix*, the position of the words are preserved. The context relations that will be found from this example in the *(w, r, w')* tuple form are *(dreams,left,your)* and *(dreams, right, may)*. Hence, the attributes of dreams are *(left, your)*, *(right, may)* and *r* are *left*, *right*. If the *asterix* was in the representation of the window as $W(L_1R_1*)$ then *r* values would be empty. Therefore for window $W(L_1R_1*)$ *(dreams, Ø, your)* and (*dreams, Ø, may)* relations are produced. A detailed example for each parameters is given in section 4.2.4.

3.2 VECTOR SPACE MODEL

The contextual representation of each headword in the dictionary has been constructed from corpus and compiled into a vector space representation where each entry in the vector space contains the frequency value of each context. In vector space based approaches the important factor for finding similarity between words is that *similar words appear in similar contexts*, i.e. they have similar context vectors.

Let the following text is the corpus from which vector space is constructed. (Imagine this is a special language consisting of 'A', 'B' and 'C' characters and each word is represented from two adjacent of these characters.)

"AA AB AA AB BB. CA CC AC AA CC BB. AA AB AB BC BB CA AC. CC AB AC AB AA AB AC BB BA AB. AC AB AA AB CA CC AA CB AB CA CC AC AA. CB AB AA AB AC BB BA AB BC BB BA AB AA BC BA BB CC."

From this corpus following headwords (different words) are found:

AA, AB, AC, BA, BB, BC, CA, CB and *CC*

Moreover let our window method is *W(L_1R_1*)* in which the window covers the nearest left and right words of the head word but not stores the position information. Then following contexts might be extracted from the corpus.

(AA, Ø, AA), (AA, Ø, AB), (AA, Ø, AC), (AA, Ø, BA), (AA, Ø, BB), (AA, Ø, BC), (AA, Ø, CA), (AA, Ø, CB), (AA, Ø, CC), (AB, Ø, AB), (AB, Ø, AC), (AB, Ø, BA), (AB, Ø, BB), (AB, Ø, BC), (AB, Ø, CA), (AB, Ø, CB), (AB, Ø, CC), (AC, Ø, AC), (AC, Ø, BA), (AC, Ø, BB), (AC, Ø, BC), (AC, Ø, CA), (AC, Ø, CB), (AC, Ø, CC), (BA, Ø, BA), (BA, Ø, BB), (BA, Ø, BC), (BA, Ø, CA), (BA, Ø, CB), (BA, Ø, CC), (BB, Ø, BB), (BB, Ø, BC), (BB, Ø, CA), (BB, Ø, CB), (BB, Ø, CC), (BC, Ø, BC), (BC, Ø, CA), (BC, Ø, CB), (BC, Ø, CC), (CA, Ø, CA), (CA, Ø, CB), (CA, Ø, CC), (CB, Ø, CB), (CB, Ø, CC), (CC, Ø, CC)

In a vector space all these headwords and possible found context information can be kept. If the columns are used for keeping contexts information, the rows will be used for the headwords. Moreover, the values in the vector space contains the frequencies of headword context tuple, as *(w, r, w')*, thus all necessary data is kept in a vector space. In this vector space each

Table 3.1: Sample Vector Space

*	AA	AB	AC	BA	BB	BC	CA	CB	CC
AA	0	11	1	0	1	1	0	2	2
AB	11	2	6	3	1	2	2	2	1
AC	1	6	0	0	2	0	1	0	3
BA	0	3	0	0	4	1	0	0	0
BB	1	0	2	4	0	2	2	0	1
BC	1	2	0	1	2	0	0	0	0
CA	0	2	1	0	2	0	0	0	3
CB	2	2	0	0	0	0	0	0	0
CC	2	1	3	0	1	0	3	0	0

row is the vector of the corresponding headword. Considering these information and corpus provided, vector space in Table 3.1 is created.

For example let us consider the value in the first row second column which is **11**. This value means that considering nearest words of headword *AA*, word *AB* exist **11** times in the corpus. Since with window method $W(L_1R_1*)$ location information is not kept the value for the headword *AB* is equal (second row first column which means word *AA* appears **11** times near the headword *AB*). Since the location information is not preserved, the matrix inside the vector space is symmetric. Moreover, this time the value **11** is also means the total count of *(AA, Ø, AB)* or *(AB, Ø, AA)* contexts (since the relation type is empty, both represent the same context).

However if the window method doesn't contain *asterix* ($W(L_1R_1)$), then position information should be kept and *(AA, left, AB)* and *(AA, right, AB)* contexts are generated. Hence, total count of *(AA, left, AB)* and *(AA, right, AB)* contexts will be **11** which is found from the summation of context *(AA, left, AB)* count, which is **5**, and context *(AA, right, AB)* count, which is **6**.

3.3 SIMILARITY MEASURE METHODS AND PARAMETERS

In this section, similarity measure function and weight function definitions are given in a similar way as in [10].

- ***Asterisk (*)*** indicates a set of values ranging over all existing values of that component

of the relation tuple.

- ***(w, *, *)*** ≡ {***(r, w')***|∃***(w, r, w')***} for all the attributes of the given headword ***w*** on a given corpus. In our case it represents row of the headword ***w*** in the vector space. (For example row of the headword *AA* in Table 3.1)

- ***wgt(w, r, w')*** represents applied weight function to the given relation *(w, r, w')*. Weight functions are explained in detail in subsection 3.3.2.

- $$\sum wgt(w_m, *_r, *'_w).wgt(w_n, *_r, *'_w)$$

  is a notational abbreviation of

  $$\sum_{(r,w')\in(w_m,*,*)\cap(w_n,*,*)} wgt(w_m, r, w').wgt(w_n, r, w')$$

  This function sums the common weighted attribute values in the vector space. Let w_m is headword *AA* and w_n is headword *AB* in Table 3.1 and let the weight function is ignored then the above definition can be calculated as following:

  $$\sum wgt(AA, *_r, *'_w).wgt(AB, *_r, *'_w)$$

  = (0x11) + (11x2) + (1x6) + (0x3) + (1x1) + (1x2) + (0x2) + (2x2) + (2x1)

  = 0 + 22 + 6 + 0 + 1 + 2 + 0 + 4 + 2

  = 37

Using these definitions, semantic distance measure functions and weight functions used in this thesis are defined as following:

3.3.1 Measure Functions:

Below are the word similarity distance measure functions which can also be used for measuring the distance between two vectors.

3.3.1.1 L_1 Norm:

The L_1 **Norm** is also called the *Manhattan* or *Levenshtein* distance.

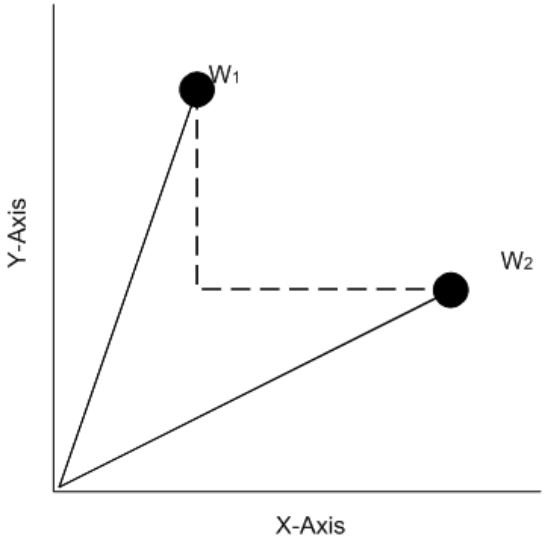

Figure 3.1: w_1 and w_2 are the headword vectors, length of the dashed line is the L_{1Norm} value between these vectors

If the vectors of the headwords (w_1 and w_2) are consisting of two values (i.e there are two different contexts in vector space model) then length of the dashed line between these vectors in figure 3.1 is the Manhattan distance. As the vectors getting closer to each others this distance will converges to the 0, and semantic similarity between the headwords increases.

$$L_{1Norm}(w_1, w_2) = \sum |wgt(w_1, *_r, *'_w) - wgt(w_2, *_r, *'_w)|$$

This method measures the absolute between two vectors in dimension-wise. For example *AA* and *AB* are the headwords for which L_{1Norm} is going to be calculated. Since vector of headword *AA* = {0, 11, 1, 0, 1, 1, 0, 2, 2} and vector of headword *AB* = {11, 2, 6, 3, 1, 2, 2, 2, 1} then Manhattan distance between these two vectors are calculated as following:

$$L_{1Norm}(AA, AB) =$$

$$= (|0 - 11| + |11 - 2| + |1 - 6| + |0 - 3| + |1 - 1| + |1 - 2| + |0 - 2| + |2 - 2| + |2 - 1|)$$

$$= 11 + 9 + 5 + 3 + 0 + 1 + 2 + 0 + 1$$

$$= 32$$

Here the returned value increases as the similarity between two headwords decreases. Therefore there is an inverse ratio between the semantic similarity of headwords and the calculated value.

3.3.1.2 L_2 Norm:

The **L2 Norm** is also called the *Euclidean* distance.

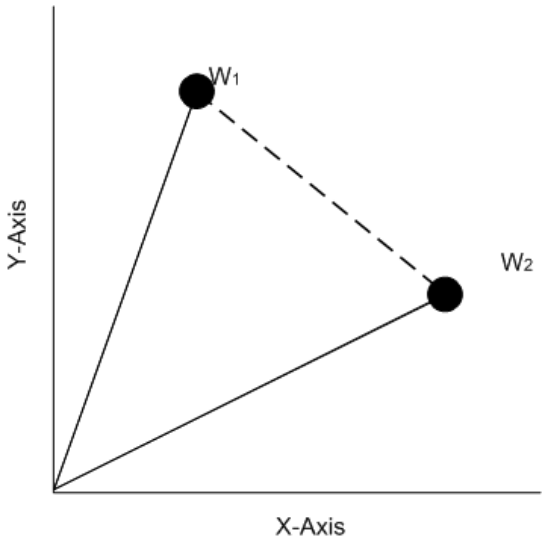

Figure 3.2: w_1 and w_2 are the headword vectors, length of the dashed line is the L_{2Norm} value between these vectors

If the vectors of the headwords (w_1 and w_2) are consisting of two values (i.e there are two different contexts in vector space model) then length of the dashed line between these vectors

in figure 3.2 is the Euclidean distance. As the vectors getting closer to each others, this distance will converges to the 0, and semantic similarity between the headwords increases.

$$L_{2Norm}(w_1, w_2) = \sqrt{\sum (wgt(w_1, *_r, *'_w) - wgt(w_2, *_r, *'_w))^2}$$

Again if semantic distance between the same headword vectors (headword *AA* and headword *AB*) are used then the Euclidean distance between these vectors are calculated as following:

$$L_{2Norm}(AA, AB) =$$

$$= \sqrt{(0-11)^2 + (11-2)^2 + (1-6)^2 + (0-3)^2 + (1-1)^2 + (1-2)^2 + (0-2)^2 + (2-2) + (2-1)^2}$$

$$= \sqrt{121 + 81 + 25 + 9 + 0 + 1 + 4 + 0 + 1}$$

$$= \sqrt{242}$$

$$= 15.5563$$

Similar to the L_{1Norm} there is an inverse ratio between the semantic similarity of headwords and the calculated L_{2Norm} value.

3.3.1.3 Cosine:

This is the angular **cosine** distance between headword vectors. Cosine is a function which is heavily used in linear algebra and become a standard measure function in IR.

If the vectors of the headwords (w_1 and w_2) are consisting of two values (i.e there are two different contexts in vector space model) then the angle between these vectors will be as in

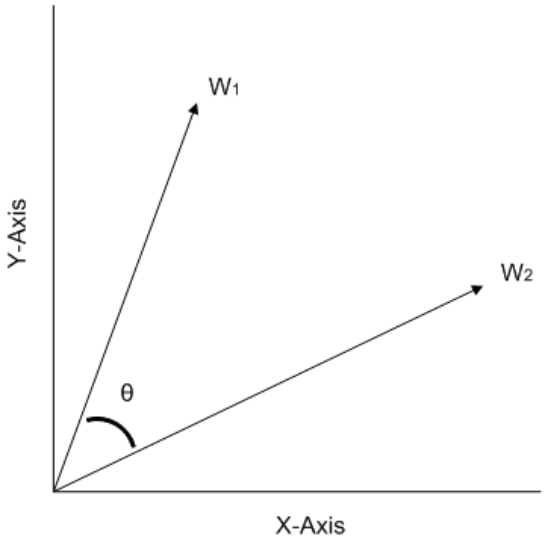

Figure 3.3: w_1 and w_2 are the headword vectors, θ is the angle between these vectors

figure 3.3. As the vectors getting closer to each others the cosine of the angle θ is increases and so the semantic similarities of the words w_1 and w_2.

$$Cosine(w_1, w_2) = \frac{\sum wgt(w_1, *_r, *'_w).wgt(w_2, *_r, *'_w)}{\sqrt{\sum wgt(w_1, *, *)^2 . \sum wgt(w_2, *, *)^2}}$$

For the headwords of *AA* and *AB* cosine is calculated as following:

$$Cosine(AA, AB) =$$

$$= \frac{(0x11) + (11x2) + (1x6) + (0x3) + (1x1) + (1x2) + (0x2) + (2x2) + (2x1)}{\sqrt{(0^2 + 11^2 + 1^2 + 0^2 + 1^2 + 1^2 + 0^2 + 2^2 + 2^2)x(11^2 + 2^2 + 6^2 + 3^2 + 1^2 + 2^2 + 2^2 + 2^2 + 1)}}$$

$$= \frac{37}{\sqrt{132x184}}$$

$$= 0.2374$$

As the angle between the vectors shortens, the cosine angle approaches 1, meaning that the

two vectors are getting closer, and the similarity of whatever is represented by the vectors increases.

3.3.1.4 Jaccard:

Jaccard (from Paul Jaccard 1901) calculates the intersection divided by the union of the headwords vectors:

$$Jaccard(w_1, w_2) = \frac{\sum min(wgt(w_1, *_r, *'_w), wgt(w_2, *_r, *'_w))}{\sum max(wgt(w_1, *_r, *'_w), wgt(w_2, *_r, *'_w))}$$

For the headwords of *AA* and *AB* Jaccard is calculated as following:

$$Jaccard(AA, AB) =$$

$$= \frac{min(0, 11) + min(11, 2) + min(1, 6) + .. + min(2, 1)}{max(0, 11) + m_x(11, 2) + max(1, 6) + ... + m_x(2, 1)}$$

$$= \frac{0 + 2 + 1 + 0 + 1 + 1 + 0 + 2 + 1}{11 + 11 + 6 + 3 + 1 + 2 + 2 + 2 + 2}$$

$$= \frac{8}{40} = 0.2$$

In Jaccard similarity function as the calculated value increases as the semantic similarity between the two headwords. Therefore there is a direct ratio between the semantic similarity of the headwords and the calculated Jaccard value. Moreover, the calculated value is always in between 0 and 1.

3.3.2 Weight Functions:

3.3.2.1 Identity:

This function returns **1** if a relation exist and returns **0** otherwise. From this definition after applying ***identify*** method to the headword *AA*'s vector, which is *AA* = {0, 11, 1, 0, 1, 1, 0, 2, 2} the resulted vector is *AA* = {0, 1, 1, 0, 1, 1, 0, 1, 1}.

3.3.2.2 Frequency:

This function returns the cell value of the relation in the vector space. In other words, it returns the frequency of the relation in the corpus. Therefore, if frequency function is applied to each element of headword vector *AA* (*AA* = {0, 11, 1, 0, 1, 1, 0, 2, 2}), the resulted vector will be the same vector, which is *AA* = {0, 11, 1, 0, 1, 1, 0, 2, 2}.

CHAPTER 4

THE PROPOSED SYSTEM

The details of the proposed system are given in the following sections.

4.1 THE PROPOSED METHOD

Starting point of this thesis is J. R. Kurran's doctoral dissertation [10] on word similarity. Here we present an improvement on finding semantically similar words. This is completely an unsupervised method and this improvement can be applied easily to many of the existing unsupervised methods which are based on vector space model. This thesis can be explained in two main steps:

- Reprocessing the vector space using previous results
- Merging existing improvements from different researches on this topic

This study is a combination of some of the previous improvements which are directly applicable to semantic word similarity. It also introduces a novel approach for vector space based semantic word similarity. This novel approach is about reprocessing the existing data with some feedbacks from the findings of previous process.

The most important properties of this approach are below:

- It is applicable to most of the existing similarity methods
- It is flexible and open for the future improvements.

Details of this novel approach is given in section 4.2.5.

4.1.1 REPROCESSING THE VECTOR SPACE

Artificial Neural Network (ANN) is a computational model used in computer science which tries to simulate the biological counterpart of the brain. ANN is consisting of neurons (which are the building blocks of the ANN structure) and a kind of ***feedback/feed-forward mechanism*** exist in their working principle. This mechanism is the most important part of ANN. Actually, in this novel ***reprocessing mechanism*** we inspired from the ***feedback/feed-forward mechanism*** of the neurons in the ANN. Our simple idea is ***"previously found similar words can be used as a feedback for the same system in order to improve the quality"***.

This novel approach can be simply defined as reprocessing of the previous results with the existing methods. After processing a headword 200 semantically similar words are found according to the semantic word distance measure functions. This is a sorted list according to the minimum semantic similarity distance between the words found and the headword. Therefore the most similar word at the top of the list, which is the headword itself, is presented and the second word in the list is the most semantically similar word to the headword. The others are listed in descending order according to the semantic similarity between the headword.

Finding semantically similar words using distance-weight measure functions for the first time is called as ***First Process (FP)*** in this study. After finding the lists of semantically similar words for each headword, the process selects a number of words from top of the result list (including the first one) and retrieves their vectors from the vector space. How many of them will be selected depends on the ***reprocess ratios***. ***Reprocess ratios*** are the multiplication factors of the vector values. Therefore next step is assignment of different multiplication factor to each of the selected words vector in the vector space. Vector values are multiplied by these multiplication factors and then these new found vectors are summed to a new vector. This new vector is used as the headword vector in the next process. Hence, found vector is considered as the updated version of the existing headword's vector in the vector space.

Let trace this ***reprocess*** issue in an example. Let headword whose semantic similar words to be found be the headword *AA* in Table 3.1. After processing the vector space using a distance-weigh method tuple let 6 words are found including the headword itself. Assume the list is the following:

We found list of most similar 6 words of the headword ***AA*** according to descending semantic

similarity is ***AA, AB ,AC, BA, BB*** and ***BC***.

Let reprocess ratios be ***1, 0.5, 0.34, 0.25, 0.2*** which means that the first 5 most similar words of the above words will be considered which are ***AA, AB ,AC, BA, BB***. Following are the vectors of these headwords:

AA = {0, 11, 1, 0, 1, 1, 0, 2, 2}

AB = {11, 2, 6, 3, 1, 2, 2, 2, 1}

AC = {1, 6, 0, 0, 2, 0, 1, 0, 3}

BA = {0, 3, 0, 0, 4, 1, 0, 0, 0}

BB = {1, 0, 2, 4, 0, 2, 2, 0, 1}

Now following are the new vectors found after applying reprocess ratios:

The result of the multiplication vector of headword ***AA*** with scaler 1 is:

AA = {0, 11, 1, 0, 1, 1, 0, 2, 2}

The result of the multiplication vector of headword ***AB*** with scaler 0.5 is:

AB = {5.5, 1, 3, 1.5, 0.5, 1, 1, 1, 0.5}

The result of the multiplication vector of headword ***AC*** with scaler 0.34 is:

AC = {0.34, 2, 0, 0, 0.68, 0, 0.34, 0, 1}

The result of the multiplication vector of headword ***BA*** with scaler 0.25 is:

BA = {0, 1, 0, 0, 1.34, 0.34, 0, 0, 0}

The result of the multiplication vector of headword ***BC*** with scaler 0.2 is:

BB = {0.34, 0, 0.68, 1.34, 0, 0.68, 0.68, 0, 0.34}

After these multiplications are applied, next operation is summing up all these vectors which is also the updated version of the headword ***AA*** and used in the next step.

Updated headword vector of ***AA*** = **{*6.18, 15, 4.68, 2.84, 3.52, 3, 2, 3, 3.84*}**

After all these operations, the updated vector of headword ***AA*** is used as if it is the original found vector from the text corpora at the beginning. Next step is recalculation of the distance between the headword and the words in the corpus for finding a new similarity list. In this study finding new similarity list using the updated version of the headwords vector, instead of its original one, is called as ***Second Process (SP)***.

We assign a decreasing multiplication factor to the vectors of the words going from top to bottom of the list. Our assumption (as in most of the similar studies in the literature) is that ***similar words have similar contexts***. According to this assumption, main aim in this vector addition process is trying to strengthen the headword vector with its most similar words' vectors. Moreover, since statistical based methods have better results with headwords frequently occurred in corpus, with this addition the headwords with very few occurrences in the corpus will be treated as the other words that are frequent in the corpus. For example, assume a headword is very rare in the corpus and in ***FP*** semantically similar words which are more common in the corpus are found. With the help of vector addition process, the second pass will treat the rare headword as its found semantically similar words so more semantically similar words in the ***SP*** can be found. This process can also be continued with third, forth... iterations and stopped when the similar list converges to a stable point or the number of modifications in the list is below some threshold. In this thesis only ***FP*** and ***SP*** results are evaluated.

The results of evaluations in most of the research in the corpus-based semantic word similarity area in the literature show that as the frequency of a headword increases in a corpus, the number of found similar words increases as well. Thus, there is a direct relation between the frequency of the headword and quality of the results in terms of semantic similarity. Moreover, if a word is rare in the corpus then its vector will contain more zero and less non-zero values. Since the main idea is ***similar words have similar contexts*** and the values in the vector of a headword represents its contexts, rare words appear in less number of contexts so finding similar contexts of a rare word is very difficult. Therefore, finding semantically similar words of a rare headword with the existing methods is expected to have poor results. However, if a similar word with high frequency is found for a very rare headword in ***FP*** this makes it possible to find several other similar words to this headword in ***SP.***

4.1.2 MERGING EXISTING IMPROVEMENTS

In [15], it is stated that pre-processing of the corpora affects the quality of the results. Therefore some of the pre-processing techniques are applied in this thesis to the raw corpus data. Moreover [16] gives importance on misspelled words in the corpus. These and several other studies are taken into consideration to come up with an optimal solution.

For the selection of similarity measure functions J. R. Kurran's experiments ([10]) are considered. According to his study one of the most promising word semantic similarity measure function is **Jaccard** (3.3.1.4). For similarity functions some other methods are also selected ($L_1 norm$, $L_2 norm$ and $Cosine$) which have also promising results and which are frequently used in researches about semantic word similarity .

4.2 EXPERIMENTAL EVALUATION

Semantically similar words of headwords are found as a result of the following steps:

- Corpus Selection
- Corpus Pre-processing
- Headword Selection
- Window Method Selection
- Defining Reprocess Ratios

4.2.1 CORPUS SELECTION

Quality and ***size of the corpus*** are the two important parameters that determine the success of results. We used one of the most popular and freely available text corpus for researches Reuters Corpus Volume 1 (RCV1) [11] in our experiments. Reuters Ltd prepared a large collection of News stories for the use of research and development of NLP, IR and Machine Learning [11]. This corpus is collection of Reuters News stories 1996-08-20 to 1997-08-19. It contains about 810,000 Reuters, English Language News stories. It consists of 8.1M

sentences and 207M words. News articles marked-up with some meta-data using an XML schema. Since the data is structured paragraph and header information can be easily retrieved from a news article.

British National Corpus (BNC) [12] is the other most popular text corpus. It is also structured and its quality in terms of less noise is better than RCV1. BNC is a 100 million word collection of samples of written and spoken language from various sources. BNC is used from several studies conducted so far but not in our study.

The size of the corpus RCV1 is sufficient but its noisy. We tried to reduce the noise to increase the information gain with pre-processing. Since the data is so large we made use the of ***High Performance Computing (HPC)*** facility of METU Computer Engineering Department [63]. This is a system containing 46 x 2 = 92 CPUs, 46 x 2 x 4 = 368 Cores, 46 x 16 GB = 736 GB Memory, 46 x 146 GB = 6.5 TB Local Disk (halved by RAID), 2 x 3 TB = 6 TB common storage area (halved by RAID). With this system complex tasks can be done in reasonable amount of time (we can run our methods in several hours). Actually without HPC (Figure:4.2.1) this thesis couldn't have been implemented successfully.

4.2.2 CORPUS PRE-PROCESSING

In this step, first we parsed the XML news files then marked each header and paragraph with special characters. Then we merged the content of each XML files from a day to a text file. While doing these we removed the stop words (stop word list is available in appendix A). In addition to this stop word list we also ignored the words above some frequency threshold and the words containing invalid characters. Moreover, we tried to fix spelling errors. At the end of pre-process, we reduced noise in data and merge the XML files for the same day to text files. Hence, resulting text files only contain one day news data.

Jazzy (The Java Open Source Spell Checker) [64] is used for fixing spelling errors. Spell checking is done in four steps. First we check whether the word exist in the Longman American Defining Vocabulary ([65]), which contains most common 2,000 English words other than stop words. If the word exists in the dictionary then we keep it and continue with the next word, otherwise we take the first suggested word from Jazzy and check the existence of the word in the vocabulary. If it exists, we keep the word and continue with the next,

Figure 4.1: A picture of HPC Room (http://hpc.ceng.metu.edu.tr/pictures)

otherwise we stem it using Porter's algorithm ([66]) and check whether the word exist in the vocabulary. If the word exists, we keep it and continue with the next one, otherwise we keep the first suggested word by Jazzy.

After these steps, we obtained stemmed and less misspelled words. Since sometimes stemming a word resulted in a corrupted word, we get suggestion from Jazzy to reconstruct the correct stemmed version word. For example, if we stem the word **"taking"**, we will get **"tak"** which is meaningless and corrupted version of the word **"take"**. If we get suggestion for the word **"tak"**, we get **"take"** which fixes the word. Therefore, after stemming the word **"taking"** we retrieve successfully the correct stemmed version of the word which is **"take"**. Below is the pseudo code of the stemming and spell checking algorithm:

for i = 1 to number_of_words_in_corpus **do**
  word ← getword_at_position(i)
  if exist_in_dictionary(word) **then**
    keepword(word)

```
    else
      word ← getJazzySuggestion(word)
      if exist_in_dictionary(word) then
        keepword(word)
      else
        word = stemmword(word)
        if exist_in_dictionary(word) then
          keepword(word)
        else
          word ← getJazzySuggestion(word)
          keepword(word)
        end if
      end if
    end if
    i ← i + 1
  end for
```

In addition to stemming and fixing misspelled words, in the pre-processing phase we also checked wrong punctuation mark usages. For example if two words were attached like *play,swim* we separated them as *play* and *swim*. We considered proper names and kept them as they are. Moreover, we treated words containing numbers differently from the others (we replace). Some other special rules were also applied to reduce the noise of the corpus.

In the experiments, two different sized subset of corpus were used. One of them contained randomly chosen 7 days news data named ***Week Corpus*** (word count=4,060,447) and the other contained 30 days news data named ***Month Corpus*** (word count= 16,953,356). We also removed the words which had frequency below a threshold in the corpus. For the *Week Corpus* threshold value was 15 and for *Month Corpus* it was 60. These values were selected in accordance with the number of words in related corpus, word informativeness and cluster memory sizes in HPC ([63]). After these, new word counts for *Week Corpus* was 1,766,549 and for *Month Corpus* was 7,347,390.

Table 4.1: Head Words and Occurrence Frequencies

Headword	Frequency in Week Corpus	Frequency in Month Corpus
announcement	346	1821
apple	346	1828
ball	168	681
car	1178	4075
company	8276	35262
country	3085	14293
dollar	2816	12772
energy	749	3550
floor	221	863
gas	1552	5995
house	1540	6406
magazine	181	702
market	11057	45074
mix	461	1860
newspaper	860	3330
opinion	289	1458
people	2692	11146
problem	1202	5130
size	415	1593
stock	5129	21043
taste	20	150
thing	517	2108
time	3603	14941
village	163	856
word	222	1022

4.2.3 HEADWORD SELECTION

We selected 25 out of 300 headwords from [10] for evaluation set. Selected headwords are ***announcement, apple, ball, car, company, country, dollar, energy, floor, gas, house, magazine, market, mix, newspaper, opinion, people, problem, size, stock, taste, thing, time, village*** and ***word***. They are listed in the Table 4.1 with their occurrence frequencies in *Week Corpus* and *Month Corpus*.

About half of these were selected according to their frequencies in the corpus and the other half randomly. For the first half of headwords, we tried to cluster the words according to their occurrence frequencies and selected some from each cluster (i.e. we selected the headwords with very high, average and low corpus occurrence frequencies). If frequencies of the words are analyzed in terms of corpus type, the ratio of frequencies between Month Corpus and Week Corpus is about **4**.

Data available in Table 4.1 is used in chapter 5 for detailed discussion of the results in terms

Table 4.2: Window Settings

Left	Right	Keep Direction	KeepPosition	BoundaryLevel
1	1	FALSE	FALSE	1
1	1	TRUE	FALSE	1
1	1	FALSE	FALSE	2
1	1	TRUE	TRUE	2
2	0	FALSE	FALSE	1
2	0	FALSE	FALSE	2
2	0	FALSE	TRUE	1
2	0	TRUE	TRUE	2
1	0	FALSE	FALSE	1
1	0	FALSE	FALSE	2
1	0	FALSE	FALSE	0
2	0	FALSE	FALSE	0
0	1	FALSE	FALSE	1
0	1	FALSE	FALSE	2
0	1	FALSE	FALSE	0
2	0	TRUE	TRUE	0
2	1	FALSE	FALSE	1
2	1	FALSE	FALSE	0
2	1	FALSE	FALSE	2
2	1	TRUE	TRUE	1

of word frequencies.

4.2.4 WINDOW METHOD SELECTION

The experiments contains $\boldsymbol{W(R_1)}$, $\boldsymbol{W(L_1)}$, $\boldsymbol{W(L_1R_1)}$, $\boldsymbol{W(L_2)}$, $\boldsymbol{W(L_2R_1)}$ windows. Moreover, in the configuration settings keeping the word direction, position were also used as parameters. Furthermore, considering sentence or paragraph boundaries were also configurable in our settings. Hence, with different settings we used 20 different window settings for each corpus. These window settings are available in appendix B.

For determining these settings we tried to select the ones which have the most promising results stated in [10].

Again let the following sentence is going to be processed and the headword is ***dream***.

"The color of your dreams may depend on what sort of television your family owned."

After the pre-process and stop words removal the remaining words are:

color, dream, depend, sort, television, you, family, own.

For window method ***W(R_1)*** window will contain ***dream*** and ***depend***.

For window method ***W(L_1)*** window will contain ***color*** and ***dream***.

For window method ***W(L_1R_1)*** window will contain ***color***, ***dream*** and ***depend***.

For window method ***W(L_2)*** window will contain ***color*** and ***dream***. Since there is only one word in the left side of the headword ***dream*** window only contains one word instead of two words.

For window method ***W(L_2R_1)*** window will contain ***color***, ***dream*** and ***depend***.

4.2.5 REPROCESS RATIOS

We tried several vector multiplication factors in reprocess step (SP). Our reprocess ratios are as follows:

- (**1**) This reprocess ratio means only semantic similar word list of **FP** is going to be found.
- (**1.0, 0.5, 0.34, 0.25, 0.2**) This reprocess values are calculated from the following algorithm:

  $ratio \leftarrow 1$
  for $i = 1$ to 5 **do**
    $newratio \leftarrow$ ratio
    keep_new_ratio(newratio)
    $ratio \leftarrow (\frac{1}{i})$
    $i \leftarrow i + 1$
  end for

- (**1.0, 0.75, 0.5, 0.25**) This reprocess values are calculated from the following algorithm:

  $ratio \leftarrow 1$
  for $i = 1$ to 4 **do**
    $newratio \leftarrow$ ratio

keep_new_ratio(newratio)

ratio ← 1-(0.25 x i)

i ← *i* + 1

end for

- (**1.34**) This reprocess ratio means only semantic similar word list of **FP** is going to be found but this time the corresponding headword vector values are multiplied by 1.34.

- (**1.5, 0.5, 0.5, 0.25, 0.25, 0.25, 0.25**) This reprocess values are calculated from the following algorithm:

  ratio ← 1.5

  j ← 1

  for *i* = 3 to 1 **do**

    newratio ← ratio

    for *k* = *j* to 0 **do**

      keep_new_ratio(newratio)

      k ← *k* − 1

    end for

    ratio ← ratio/i

    i ← *i* − 1

    j ← *j* + 1

  end for

- (**10.0**) Only FP is needed.

- (**2.0**) Only FP is needed.

The reprocess ratios which requires only **FP** are for evaluation purposes. The other reprocess ratios require both FP and SP. The common point in reprocess ratios, which requires SP, is decreasing reprocess values. The detailed analysis of these values and how they affect the quality of the results are discussed in chapter 5.

CHAPTER 5

RESULTS

The results are slitted into two categories according to corpus size. The first category contains the results found from the *Week Corpus* and second one from the *Month Corpus*. Our experiments consist of 70,000 different cases and 35,000 cases for each different sized corpus. All entries in this result set contains 25 different result values according different semantic word similarity value found between the related headword and its found semantically similar words from the experiments. Therefore 25 X 70.000 = **1.750.000** values are calculated in terms of semantic word similarity in the experiments. Since providing all these results is not meaningful, results are given after some filtering operations are applied.

5.1 DEFINITIONS IN TABLES

The parameters, which are the column names of the tables in the next sections, change in each case can be listed as follows. Moreover, for each case we found 200 similar words listed in ascending order according to their similarity distance from the given headword.

Here is the abbreviations that are used in the tables and their definitions:

- **Left window size** (**LW**): Represents how many words from the left side of the headword will be in the window. **0**, **1** and **2** cases have been evaluated.
- **Right window size** (**RW**): Represents how many words from the right side of the headword will be in the window. **0** and **1** cases have been evaluated.
- **Keep position of the word in window** (**KP**): Position information of the words (how many words exist between the word and the headword) in the window is preserved or

not. Both two possible cases have been evaluated.

- **Keep direction of the word in window** (**KD**): Direction information of the words (whether word is located in the left or right side of the headword) in the window is preserved or not. Both two possible cases have been evaluated.

- **Boundary level for the window** (**BL**): Boundary level is used when window size permits the word near the headword to put it into the window but the sentences or the paragraphs of the words are not same. There are three different levels exist in the experiments for boundary level. For level 0 no boundary limit is assumed, for level 1 the words in the window should be in the same paragraph with headword and for level 2 they should be in the same sentence.

- **Semantic distance level method** (**Method**): *L1norm*, *L2norm*, *Cosine* and *Jaccard* semantic word similarity measure functions have been applied.

- **Weight function used with the method** (**Weight**): *Frequency* and *Identity* weight functions have been used.

- **Find a similar word at first position average** (**Av. P1**): Arithmetic average of the evaluated headwords results according to finding a semantically similar word in the first position in the resulted list. If a similar word found in the first position then P1 value is 1 otherwise 0.

- **Average number of similar words found up to position '#'** (**Av. P#**): Arithmetic average of the evaluated headwords results according to number of found semantically similar words up to position # in the resulted list. For example if 5 similar words have been found in first 100 of 200 words then **P100** value is 5, and **#** represents **100**. In the experiments **1, 5, 10, 20, 50, 100** and **200** are the numbers that can be replaced in the place of #.

- **Average similarity value calculated according to [54] between the headword and the next most similar word in the list** (**Av. Lin1**): Arithmetic average of the evaluated headwords results according to the similarity value calculated, which is WordNet based semantic similarity method proposed by Dekang Lin [54]. This value is the calculated similarity value between the headword and the first most similar word in the list. The

proposed semantic word similarity method returns a value between 0 and 1 where 1 represents a higher similarity and 0 represents low or no similarity.

- **Average total similarity value calculated according to [54] between the headword and first '#' of words in the found list** (**Av. Lin#**): Arithmetic average of the evaluated headwords results based on the similarity value calculated according to WordNet based semantic similarity method proposed by Dekang Lin [54]. This value is the calculated total similarity value between the headword and the first # of words in the list.

- **Average similarity value calculated according to [29] between the headword and the next most similar word in the list** (**Av. Jcn1**): Arithmetic average of the evaluated headwords results according to the similarity value calculated, which is WordNet based semantic similarity method proposed by Jiang and Conrath [29]. This value is the calculated similarity value between the headword and the first most similar word in the list. The proposed semantic word similarity method returns a value between 0 and 5.15E+9, as the similarity increases this calculated value also increases.

- **Average total similarity value calculated according to [29] between the headword and first '#' of words in the found list** (**Av. Jcn#**): Arithmetic average of the evaluated headwords results based on the similarity value calculated according to WordNet based semantic similarity method proposed by Jiang and Conrath [54]. This value is the calculated total similarity value between the headword and the first # of words in the list.

- **Average inverse rank** (**Av.Inv.**): Arithmetic average inverse rank value of the evaluated 24 headwords results according to whether a word is similar or not and its position in the resulted list. For example if words in the 1^{th} 2^{nd} and 4^{th} positions are semantically similar with headword then inverse rank is 1 + 1/2 + 1/4 = 1.75.

- **Reprocess ratios** (**RP**): This is the new addition to existing methods with this thesis. These pre-process ratios are given in section 4.2.5. Below are the reprocess ratios and their abbreviations in the next sections.

  - 1 ← M1
  - 1.0, 0.5, 0.34, 0.25, 0.2 ← M2
  - 1.0, 0.75, 0.5, 0.25 ← M3

Table 5.1: Corpus-Based Average Values

Corpus	P1	P10	P200	Inv Rank	Lin1	Lin100	Jcn1	Jcn100
Week	0.1153	0.6156	19.5817	0.3203	0.2544	19.5817	121738737	1253466515
Month	0.1246	0.7505	5.7299	0.3753	0.2811	21.3757	89223785	1516095056

- 1.34 ← M4
- 1.5, 0.5, 0.5, 0.25, 0.25, 0.25, 0.25 ← M5
- 10.0 ← M6
- 2.0 ← M7

Moby [67] and WordNet [3] are used for determining whether two word are similar or not for calculating **P#** values. If they are similar we used **1** otherwise **0** when preparing the results.

In most of the tables in this chapter the columns are average **P1**, **P10**, **P200**, **Inv. Rank**, **Lin1**, **Lin100**, **Jcn1** and **Jcn100**. We choose these parameters because they are from all semantic measure techniques. Moreover, analysis of the results according to up the 1^th, 10^th, 100^th and 200^th of 200 similar words gives sufficient idea when comparing the results and retrieving information about the success of the results.

5.2 GENERAL RESULTS

Since there are two corpora (*Week Corpus* and *Month Corpus*) general results can be listed as in Table 5.1. In this table all average values of the related measure parameters are given. When a comparison is made between the values of *Week Corpus* and *Month Corpus*, it is seen that *Month Corpus* has better results in terms of semantic word similarities than *Week Corpus*. Since the word count is more in *Month Corpus* this is an expected behavior. As stated in the literature survey (chapter 2) there is a direct proportion in the quality of the resulted similar lists in terms of semantic word similarity and the corpus size.

5.3 REPROCESS RATIO BASED RESULTS

Table 5.2 and Table 5.3 contain the results according to the reprocess ratio values.

When Table 5.2 is examined (which contains *Week Corpus* based results) the following results can be retrieved:

- In terms of **P1** values M6 is the worst reprocess ratio and the next worst one is the M1 which produces the original **FP** results. Therefore 5 out of 6 reprocess values have better results than **M1**. This means that when thinking generally (no method-weight filtering is applied), most of the time our approach improves the results in terms of **P1** results.
- Other than last column there is not an explicit improvement or worsening when reprocess ratio M1 values are compared to the other reprocess ratio values. However in the last column there is a worsening in terms of average **Jcn100** values when the proposed new approach is used.
- In general **M4** and **M7** have better results.

From these findings it is seen that, in general perspective, new approach has influence on the ordering of the words found in the resulted lists. That means, it increases the probability of finding a similar word at the top of the found lists.

When Table 5.3 (which contains *Month Corpus* based results) following information can be retrieved:

- There is not an explicit improvement or worsening when M1 values are compared to the other reprocess ratio values in all columns in *Month Corpus* results.
- When reprocess ratio based comparison is made, in most of the column values M4 and M7 have better results than all other reprocess ratios.

When these findings are considered there is not an explicit improvement or worsening when all reprocess ratios other than M1 is considered however some of them, especially M4 and M7 have better results than all other reprocess ratios.

From these two tables (Table 5.2 and Table 5.3) some of the reprocess ratios have better results than others. Actually ***average inverse rank value*** is one the most import parameter that defines the success of the results. Because in this, both the number of similar words

Table 5.2: Reprocess Ratio Based Average Values in *Week Corpus*

Method	P1	P10	P200	Inv Rank	Lin1	Lin100	Jcn1	Jcn100
M1	0.1115	0.6306	5.1489	0.3205	0.2595	19.4476	129632857	1333108001
M2	0.1254	0.5861	4.8032	0.3171	0.254	18.9389	155385368	1207074592
M3	0.1192	0.5836	4.9046	0.3163	0.254	19.3309	130303963	1144083464
M4	0.1126	0.6324	5.1902	0.3227	0.2598	19.5592	123533940	1296254358
M5	0.1188	0.6042	4.7976	0.317	0.246	19.1519	141409846	1216018155
M6	0.1002	0.6154	5.4454	0.311	0.2479	20.673	72061465	1277815555
M7	0.1192	0.6567	5.3039	0.3373	0.2596	19.9703	99836634.1	1299863692

Table 5.3: Reprocess Ratio Based Average Values in *Month Corpus*

Method	P1	P10	P200	Inv Rank	Lin1	Lin100	Jcn1	Jcn100
M1	0.1256	0.7858	5.8296	0.3849	0.2846	21.1534	96550191.4	1641535749
M2	0.1206	0.7074	5.4178	0.3576	0.2698	20.8506	111991934	1469987890
M3	0.1226	0.7178	5.5186	0.3618	0.2773	21.2082	104567726	1430868809
M4	0.131	0.7868	5.8754	0.3915	0.2909	21.3218	97579640.9	1617077451
M5	0.1176	0.7182	5.4038	0.3558	0.2684	21.0564	78425794.4	1375400253
M6	0.1214	0.7436	6.0772	0.3756	0.2862	22.3387	58460560	1497311417
M7	0.1332	0.7942	5.9886	0.3999	0.2905	21.701	76990650.9	1580483822

found and the found words positions are considered. In terms of this parameter again M4 and M7 have better results in both corpora and this means proposed method have better affects on the semantic similarity methods. Moreover, from these general results, which reprocess ratios should be used for a general improvement can be found. The common point between M4 (1.34) and M7 (2.0) is that in both cases headword vector is multiplied by a scalar.

5.4 HEADWORD BASED RESULTS

Table 5.4 contains calculated average inverse rank values of the results based on headwords. Since there is not any semantically similar word to the headword *apple*, in section 5.6 we ignored results of this headword. When Table 5.4 is examined with Table 4.1 in chapter 4.2 it is discovered that as the frequency of the headword is increases as the quality of the results. This is an expected result according to the studies in the literature. Moreover again *Moth Corpus* have better results than *Week Corpus*.

An interesting result is that as the word frequencies increases the quality difference between the two corpora decreases. Moreover up to a number, the headword occurrence frequency increases the quality of the results. However, if the headword exists in sufficient frequency in

Table 5.4: Headword Based Average Inverse Rank Values

headword	Week Corpus	Month Corpus
announcement	0.1847	0.2044
apple	0	0
ball	0.2392	0.3099
car	0.1754	0.3647
company	0.6215	0.6067
country	0.6153	0.6095
dollar	0.4153	0.2677
energy	0.2326	0.2359
floor	0.2621	0.223
gas	0.5733	0.8529
house	0.5641	0.5289
magazine	0.2843	0.2995
market	0.4744	0.5728
mix	0.0249	0.0244
newspaper	0.334	0.4294
opinion	0.0688	0.1553
people	0.1227	0.212
problem	0.2823	0.4202
size	0.1703	0.1597
stock	0.9959	0.9873
taste	0.1191	0.3362
thing	0.2096	0.3902
time	0.4264	0.4314
village	0.3967	0.5171
word	0.2134	0.2438

the corpus, increasing the occurrence frequency of the headword shall either very low or no affect on the quality of the results. In addition to this, in such a case increasing the size of a corpus is either very low or no affect on the quality of the results.

5.5 DISTANCE MEASURE METHOD BASED RESULTS

Table 5.5 and Table 5.6 are the two table containing distance measure method results in different corpora. When these results are examined **Jaccard** outperforms the other semantic word similarity distance measure functions. Then the next successful results are from **Cosine**, then L2Norm and the worst results are from L1Norm function. Actually these results are consistent when the study [10] is considered. This also approves the correctness of the thesis and the found results in experiments.

Table 5.5: Distance Based Similarity Quality Analysis In *Week Corpus*

	P1	P10	P200	Inv Rank	Lin1	Lin100	Jcn1	Jcn100
Cosine	0.1576	0.8047	6.03	0.4105	0.3102	21.9712	116916050	1433629722
Jacard	0.2193	0.9067	6.7293	0.5035	0.3893	24.4948	220730344	1408563662
L1 Norm	0.0934	0.4684	2.65	0.2337	0.2034	11.015	151744187	1046470441
L2 Norm	0.0998	0.4929	3.5287	0.2593	0.2311	14.2019	119390060	1201529023

Table 5.6: Distance Based Similarity Quality Analysis In *Month Corpus*

	P1	P10	P200	Inv Rank	Lin1	Lin100	Jcn1	Jcn100
Cosine	0.188	0.8829	6.4173	0.4673	0.3629	23.0807	117941153	1541471936
Jacard	0.216	0.9936	7.1931	0.5271	0.4122	25.2618	191763041	1441653110
L1 Norm	0.1091	0.7447	3.9684	0.3448	0.2461	14.7579	68207366.2	1656040712
L2 Norm	0.1069	0.6836	4.6766	0.3365	0.2441	17.229	68207366.2	1711533942

An important information that can be retrieved from these tables is that as the quality of the distance measure function increases, the importance of the headword occurrence frequency in the the corpus decreases. This is a very crucial finding because with better distance measure methods, even with small sized corpus, better results can be found. However, for the distance measure functions which have poor results, in order to get better results, corpus size should be increased. Increasing the corpus size is a time and storage consuming operation in terms of computing.

5.6 DETAILED RESULTS

In this section detailed analysis of the results are evaluated. Average **P1, Inverse Rank, Lin1** and **Jcn1** results in terms of promising *window methods* and *distance-weight* methods are considered. Analysis of these results is crucial because from these results which parameters really affect the quality of results can be discovered.

5.6.1 Detailed Average P1 Results

When average **P1** results are analyzed from Table 5.7 and Table 5.8 following information can be extracted:

1. The general rule about relationship between the quality of the results and the corpus size persists in these tables (i.e as the corpus size increases the quality of the results also increase). However, as the quality of the methods increases, the increase caused from the corpus size decreases. There are also cases where the results of a method are better in the *Corpus Week* than in the *Corpus Month*. From these first rule can be

extracted as **"If the window-distance-weight method combination produces poor results in terms of semantic word similarity then increasing the corpus size improves the results when considering average P1 values of all headwords. However if the window-distance-weight combination produces good results then increasing corpus size does not always means an improvement in the quality of the results."**

2. In terms of average **P1** results in almost all cases, in both *Corpus Week* and *Corpus Month*, proposed novel approach has better results. When these findings are compared to the findings of section 5.2, as the success of the results increases in terms of **P1** averages and **window-distance-weight** combination produces good results in **FP**, **SP** improves the results of **FP**. This case is valid especially in reprocess methods M2, M3 and M5 in which reprocess ratios covers the semantically similar words other than the headword itself. From these findings, it is found that the novel approach proposed in this thesis has more positive affects when **FP** has good results than when **FP** has poor results. Hence, the second rule can be extracted as **"The novel approach proposed in this thesis improves the results more if the FP produces good results than if the FP produces poor results."**. Actually this is very important because, improving the most promising methods means producing even better results. This shows the real success of this novel approach.

3. When distance function based analysis is made **Jaccard** is the most successful one.

4. When distance-weight combination based analysis is made **Jaccard-Frequency** combination produces the best results.

5. When window method based analysis is made window with **LW 1, RW 1, KD 1, KP 1** and **BL 2** produces the best results.

6. When window-distance-weight combination based analysis is made window with **LW 1, RW 1, KD 1, KP 1, BL 2** and **Jaccard-Frequency** produces the best results.

7. When the best window-distance-weight combination is analyzed new approach produces up to **37.5%** (comparison of M1 and M2) better results compared to FP results in *Week Corpus* and **42%** (comparison of M1 and M4) better results compared to FP results in *Month Corpus*.

Table 5.7: Detailed Average P1 Results in *Week Corpus*

						RP	M1	M2	M3	M4	M5	M6	M7
LW	RW	KD	KP	BL	Method	Weight	Av.P1	Av.P1	Av.P1	Av.P1	Av.P1	Av.P1	Av.P1
1	1	1	0	1	l1norm	identity	0.125	0.125	0.125	0.125	0.125	0.125	0.125
1	1	1	0	1	l1norm	frequency	0.0834	0.041	0.0417	0.0834	0.0834	0.0417	0.0834
1	1	1	0	1	l2norm	identity	0.125	0.125	0.125	0.125	0.125	0.125	0.125
1	1	1	0	1	l2norm	frequency	0.125	0.125	0.125	0.1667	0.125	0.1667	0.1667
1	1	1	0	1	cosine	identity	0.25	0.3334	0.25	0.25	0.125	0.25	0.25
1	1	1	0	1	cosine	frequency	0.1668	0.125	0.25	0.1667	0.084	0.1667	0.1667
1	1	1	0	1	Jaccard	identity	0.25	0.375	0.375	0.25	0.292	0.25	0.25
1	1	1	0	1	Jaccard	frequency	0.3333	0.4167	0.375	0.375	0.4167	0.125	0.375
1	1	1	1	2	l1norm	identity	0.084	0.125	0.125	0.0833	0.125	0.0833	0.0833
1	1	1	1	2	l1norm	frequency	0.084	0.084	0.084	0.0833	0.084	0.0417	0.0625
1	1	1	1	2	l2norm	identity	0.084	0.125	0.125	0.0833	0.125	0.0833	0.0833
1	1	1	1	2	l2norm	frequency	0.125	0.125	0.125	0.0833	0.125	0.1875	0.125
1	1	1	1	2	cosine	identity	0.292	0.3334	0.2917	0.2292	0.25	0.2292	0.2292
1	1	1	1	2	cosine	frequency	0.25	0.35	0.25	0.2292	0.25	0.2292	0.2292
1	1	1	1	2	Jaccard	identity	0.208	0.375	0.375	0.2292	0.3334	0.2292	0.2292
1	1	1	1	2	Jaccard	frequency	0.3333	0.4584	0.4167	0.2708	0.3334	0.125	0.3958
2	1	1	1	1	l1norm	identity	0.0417	0.0834	0.125	0.0417	0.1667	0.0417	0.0417
2	1	1	1	1	l1norm	frequency	0.0834	0.0834	0.0417	0.0833	0.0834	0.0833	0.125
2	1	1	1	1	l2norm	identity	0.0417	0.0834	0.125	0.0417	0.1667	0.0417	0.0417
2	1	1	1	1	l2norm	frequency	0.0834	0.125	0.2084	0.125	0.0834	0.2083	0.125
2	1	1	1	1	cosine	identity	0.125	0.2084	0.0834	0.125	0.2084	0.125	0.125
2	1	1	1	1	cosine	frequency	0.125	0.125	0.2034	0.125	0.0417	0.125	0.125
2	1	1	1	1	Jaccard	identity	0.125	0.1667	0.2917	0.125	0.1667	0.125	0.125
2	1	1	1	1	Jaccard	frequency	0.2917	0.3333	0.3334	0.3333	0.375	0.125	0.3333
2	1	0	0	1	l1norm	identity	0.125	0.0834	0.0834	0.125	0.1667	0.125	0.125
2	1	0	0	1	l1norm	frequency	0.125	0.125	0.0834	0.125	0.0834	0.0833	0.125
2	1	0	0	1	l2norm	identity	0.125	0.0834	0.0834	0.125	0.1667	0.0833	0.125
2	1	0	0	1	l2norm	frequency	0.0834	0.0834	0.125	0.0833	0.0834	0.0833	0.125
2	1	0	0	1	cosine	identity	0.1667	0.2084	0.2084	0.1667	0.2084	0.1667	0.1667
2	1	0	0	1	cosine	frequency	0.1667	0.125	0.125	0.1667	0.0417	0.1667	0.1667
2	1	0	0	1	Jaccard	identity	0.25	0.2917	0.1667	0.25	0.1667	0.25	0.25
2	1	0	0	1	Jaccard	frequency	0.2917	0.2917	0.3334	0.3333	0.375	0.1667	0.2917
						ColumnAv.	0.1615	0.192	0.19	0.1627	0.1746	0.14	0.1689

8. Best result in *Week Corpus* is found from window with **LW 1, RW 1, KD 1, KP 1, BL 2** and **Jaccard-Frequency** combination with method **M2**.

9. Best result in *Month Corpus* is found from window with **LW 1, RW 1, KD 1, KP 1, BL 2** and **Jaccard-Frequency** combination with method **M4**.

10. If best distance-weight combination is selected **M6** produces the worst results. This is expected because multiplication of the headword vector with a scalar of **10** in **M6** is not very feasible and meaningful.

5.6.2 Detailed Average Inverse Rank Results

When average **inverse rank (Av.Inv)** results are analyzed from Table 5.9 and Table 5.10 following information can be extracted:

1. Similar results to the detailed average **P1** analysis results in item 1 can be extracted. However, when compared to average **P1** results with average **Inv.** results, in the letter

Table 5.8: Detailed Average P1 Results in *Month Corpus*

						RP	M1	M2	M3	M4	M5	M6	M7
LW	RW	KD	KP	BL	Method	Weight	Av.P1	Av.P1	Av.P1	Av.P1	Av.P1	Av.P1	Av.P1
1	1	1	0	1	l1norm	identity	0.1167	0.125	0.25	0.1667	0.2084	0.1667	0.1667
1	1	1	0	1	l1norm	frequency	0.125	0.125	0.125	0.125	0.125	0.125	0.0833
1	1	1	0	1	l2norm	identity	0.1667	0.125	0.25	0.1667	0.2084	0.1667	0.1667
1	1	1	0	1	l2norm	frequency	0.125	0.125	0.125	0.1667	0.0417	0.2083	0.1667
1	1	1	0	1	cosine	identity	0.3334	0.1667	0.2084	0.3333	0.1667	0.3333	0.3333
1	1	1	0	1	cosine	frequency	0.125	0.0834	0.0834	0.125	0.125	0.125	0.125
1	1	1	0	1	Jaccard	identity	0.25	0.2084	0.2917	0.25	0.25	0.25	0.25
1	1	1	0	1	Jaccard	frequency	0.25	0.375	0.3334	0.4167	0.3334	0.1667	0.4167
1	1	1	1	2	l1norm	identity	0.2084	0.1667	0.2084	0.2083	0.1667	0.2083	0.2083
1	1	1	1	2	l1norm	frequency	0.16667	0.125	0.125	0.1667	0.125	0.125	0.0833
1	1	1	1	2	l2norm	identity	0.2084	0.1667	0.2084	0.2083	0.1667	0.2083	0.2083
1	1	1	1	2	l2norm	frequency	0.125	0.125	0.125	0.2083	0.1667	0.2083	0.2083
1	1	1	1	2	cosine	identity	0.375	0.25	0.25	0.375	0.25	0.375	0.375
1	1	1	1	2	cosine	frequency	0.16667	0.125	0.1667	0.1667	0.2084	0.1667	0.1667
1	1	1	1	2	Jaccard	identity	0.2917	0.2084	0.2917	0.2917	0.2917	0.2917	0.2917
1	1	1	1	2	Jaccard	frequency	0.3224	0.375	0.375	0.4583	0.3334	0.2083	0.375
2	1	1	1	1	l1norm	identity	0.25	0.1667	0.1667	0.25	0.125	0.25	0.25
2	1	1	1	1	l1norm	frequency	0.125	0.125	0.152	0.0833	0.125	0.125	0.0417
2	1	1	1	1	l2norm	identity	0.25	0.1667	0.1667	0.25	0.125	0.25	0.25
2	1	1	1	1	l2norm	frequency	0.0834	0.0834	0.1667	0.1667	0.125	0.2083	0.25
2	1	1	1	1	cosine	identity	0.1667	0.1667	0.25	0.1667	0.125	0.1667	0.1667
2	1	1	1	1	cosine	frequency	0.0834	0.125	0.042	0.0833	0.125	0.0833	0.0833
2	1	1	1	1	Jaccard	identity	0.2084	0.2084	0.25	0.2083	0.1667	0.2083	0.2083
2	1	1	1	1	Jaccard	frequency	0.25	0.2917	0.2916	0.1667	0.375	0.1667	0.4167
2	1	0	0	1	l1norm	identity	0.0417	0.0417	0.125	0.0417	0.0834	0.0417	0.0417
2	1	0	0	1	l1norm	frequency	0.0834	0	0.0417	0	0.0417	0.125	0.1667
2	1	0	0	1	l2norm	identity	0.0417	0.0417	0.125	0.0417	0.0834	0.0417	0.0417
2	1	0	0	1	l2norm	frequency	0.0417	0.0417	0.0417	0.0417	0.125	0.25	0.0417
2	1	0	0	1	cosine	identity	0.291	0.2917	0.2917	0.2917	0.25	0.2917	0.2917
2	1	0	0	1	cosine	frequency	0.1667	0.1667	0.2084	0.1667	0.2084	0.1667	0.1667
2	1	0	0	1	Jaccard	identity	0.1667	0.2083	0.2084	0.25	0.2084	0.1667	0.1667
2	1	0	0	1	Jaccard	frequency	0.0834	0.1667	0.1667	0.25	0.375	0.1667	0.2083
						ColumnAv.	0.1778	0.1615	0.191	0.1966	0.1823	0.1888	0.2

one corpus size have more affect on the quality of the results. This information can be extracted from **ColumnAv.** (related columns average) values. When these average column values are considered in all methods results of *Corpus Month* are better than the results of *Corpus Month.*

2. In terms of average **Inv. Rank** results in almost all cases in *Corpus Week* proposed novel approach has better results, but not in *Corpus Month*. When these findings are compared to the findings of section 5.2, as the success of the results increases in terms of **Inv. Rank** averages and **window-distance-weight** combination produces good results in **FP**, **SP** improves the results of **FP**. Therefore again same rules can be extracted as detailed average **P1** analysis results in item 2.

3. When distance function based analysis is made **Jaccard** is the most successful one (similar to the detailed average **P1** analysis results in item 3).

4. When distance-weight combination based analysis is made **Jaccard-Frequency** combination produces the best results (similar to the detailed average **P1** analysis results in item 4).

5. When window method based analysis is made window with **LW 1, RW 1, KD 1, KP 1** and **BL 2** produces the best results (similar to the detailed average **P1** analysis results in item 5).

6. When window-distance-weight combination based analysis is made window with **LW 1, RW 1, KD 1, KP 1, BL 2** and **Jaccard-Frequency** produces the best results (similar to the detailed average **P1** analysis results in item 6).

7. When the best window-distance-weight combination is analyzed new approach produces up to **12%** (comparison of M1 and M2) better results compared to FP results in *Week Corpus* and **7%** (comparison of M1 and M4) better results compared to FP results in *Month Corpus*. These improvement percentages seems smaller when compared with the improvement in average **P1** values. Actually this not because inverse ranking both covers how many similar words have been found and found similar words position in the list. Another useful information is that total number of similar words with a headword may not be too much and if the most of the similar words have been found in FP finding the remaining is very difficult in SP.

8. Best result in *Week Corpus* is found from window with **LW 1, RW 1, KD 1, KP 1, BL 2** and **Jaccard-Frequency** combination with method **M2** (similar to the detailed average **P1** analysis results in item 8).

9. Best result in *Month Corpus* is found from window with **LW 1, RW 1, KD 1, KP 1, BL 2** and **Jaccard-Frequency** combination with method **M4** (similar to the detailed average **P1** analysis results in item 9).

10. If the detailed tables of average **P1** and average **Inv. Rank** values are considered an another important rule also extracted. This rule is **"This new approach have better affects in corpus with less number of word if the best window-distance-weight method combination is used"**.

5.6.3 Detailed Average Lin1 Results

When average **Lin1** results are analyzed from Table 5.11 and Table 5.12 following information can be extracted:

Table 5.9: Detailed Average Inverse Rank Results in *Week Corpus*

						RP	M1	M2	M3	M4	M5	M6	M7
LW	RW	KD	KP	BL	Method	Weight	Av.Inv.	Av.Inv.	Av.Inv.	Av.Inv.	Av.Inv.	Av.Inv.	Av.Inv.
1	1	1	0	1	l1norm	identity	0.2353	0.2354	0.2466	0.2353	0.2505	0.2353	0.2353
1	1	1	0	1	l1norm	frequency	0.1688	0.1518	0.1583	0.1715	0.2357	0.1977	0.2881
1	1	1	0	1	l2norm	identity	0.2353	0.2354	0.2466	0.2353	0.2504	0.2353	0.2353
1	1	1	0	1	l2norm	frequency	0.2857	0.3123	0.3306	0.3589	0.3211	0.4178	0.3817
1	1	1	0	1	cosine	identity	0.5876	0.6368	0.5655	0.587	0.391	0.587	0.587
1	1	1	0	1	cosine	frequency	0.3907	0.3643	0.3246	0.3724	0.318	0.3907	0.3907
1	1	1	0	1	Jaccard	identity	0.6081	0.6771	0.6809	0.6081	0.6567	0.6081	0.6081
1	1	1	0	1	Jaccard	frequency	0.7162	0.7603	0.7349	0.7413	0.8107	0.3745	0.7194
1	1	1	1	2	l1norm	identity	0.2405	0.2693	0.281	0.2405	0.2828	0.2405	0.2405
1	1	1	1	2	l1norm	frequency	0.195	0.1987	0.2038	0.1969	0.243	0.2157	0.2794
1	1	1	1	2	l2norm	identity	0.2405	0.2693	0.281	0.2405	0.2828	0.2405	0.2405
1	1	1	1	2	l2norm	frequency	0.2957	0.322	0.3397	0.3573	0.3458	0.5434	0.04222
1	1	1	1	2	cosine	identity	0.6161	0.6316	0.6417	0.6161	0.5523	0.6161	0.6161
1	1	1	1	2	cosine	frequency	0.4975	0.5049	0.481	0.4779	0.4892	0.4975	0.4975
1	1	1	1	2	Jaccard	identity	0.607	0.6714	0.6771	0.607	0.6314	0.607	0.607
1	1	1	1	2	Jaccard	frequency	0.7331	0.8212	0.7796	0.7328	0.7308	0.4135	0.7501
2	1	1	1	1	l1norm	identity	0.1977	0.2125	0.2373	0.1977	0.2573	0.1977	0.1977
2	1	1	1	1	l1norm	frequency	0.1718	0.1813	0.1565	0.1674	0.2203	0.2028	0.2585
2	1	1	1	1	l2norm	identity	0.1977	0.2125	0.2373	0.1977	0.2573	0.1977	0.1977
2	1	1	1	1	l2norm	frequency	0.2682	0.3294	0.3377	0.3324	0.2973	0.4525	0.3661
2	1	1	1	1	cosine	identity	0.4948	0.4397	0.4818	0.4948	0.6102	0.4948	0.4948
2	1	1	1	1	cosine	frequency	0.3908	0.3827	0.343	0.3829	0.3596	0.3903	0.3903
2	1	1	1	1	Jaccard	identity	0.4919	0.4943	0.563	0.4919	0.4549	0.4919	0.4919
2	1	1	1	1	Jaccard	frequency	0.6661	0.7085	0.6863	0.6756	0.6648	0.3568	0.6634
2	1	0	0	1	l1norm	identity	0.3378	0.3173	0.2125	0.3378	0.2573	0.3378	0.3378
2	1	0	0	1	l1norm	frequency	0.2823	0.3	0.1813	0.2992	0.2203	0.2446	0.3573
2	1	0	0	1	l2norm	identity	0.3378	0.3173	0.2125	0.3378	0.2573	0.3378	0.3378
2	1	0	0	1	l2norm	frequency	0.2757	0.2264	0.3294	0.2495	0.2973	0.3645	0.2966
2	1	0	0	1	cosine	identity	0.5143	0.4717	0.4397	0.5143	0.6102	0.5143	0.5143
2	1	0	0	1	cosine	frequency	0.4356	0.395	0.3827	0.4247	0.3596	0.4356	0.4356
2	1	0	0	1	Jaccard	identity	0.5474	0.5627	0.4943	0.5474	0.4549	0.5474	0.5474
2	1	0	0	1	Jaccard	frequency	0.5649	0.5964	0.7085	0.6044	0.6648	0.4247	0.5832
						ColumnAv.	0.4009	0.4128	0.4055	0.4073	0.4074	0.3878	0.4122

Table 5.10: Detailed Average Inverse Rank Results in *Month Corpus*

						RP	M1	M2	M3	M4	M5	M6	M7
LW	RW	KD	KP	BL	Method	Weight	Av.Inv.	Av.Inv.	Av.Inv.	Av.Inv.	Av.Inv.	Av.Inv.	Av.Inv.
1	1	1	0	1	l1norm	identity	0.4775	0.4769	0.5608	0.4775	0.5087	0.4775	0.4775
1	1	1	0	1	l1norm	frequency	0.3589	0.3944	0.4029	0.3651	0.3954	0.3425	0.3839
1	1	1	0	1	l2norm	identity	0.4775	0.4769	0.5608	0.4775	0.5087	0.4775	0.4775
1	1	1	0	1	l2norm	frequency	0.3109	0.3341	0.3308	0.3851	0.3101	0.4666	0.3894
1	1	1	0	1	cosine	identity	0.7349	0.4974	0.5613	0.7349	0.5296	0.7349	0.7349
1	1	1	0	1	cosine	frequency	0.3757	0.3068	0.3101	0.3712	0.343	0.3757	0.3757
1	1	1	0	1	Jaccard	identity	0.7114	0.5968	0.6694	0.7114	0.5983	0.7114	0.7114
1	1	1	0	1	Jaccard	frequency	0.7094	0.7549	0.7392	0.7945	0.7518	0.4144	0.7914
1	1	1	1	2	l1norm	identity	0.523	0.5188	0.5433	0.523	0.49151	0.523	0.523
1	1	1	1	2	l1norm	frequency	0.3773	0.3837	0.3845	0.3857	0.4	0.3547	0.3746
1	1	1	1	2	l2norm	identity	0.523	0.5188	0.5433	0.523	0.4951	0.523	0.523
1	1	1	1	2	l2norm	frequency	0.3477	0.3537	0.349	0.3874	0.4005	0.56	0.4382
1	1	1	1	2	cosine	identity	0.7674	0.5608	0.5862	0.7674	0.5339	0.7674	0.7674
1	1	1	1	2	cosine	frequency	0.4954	0.4021	0.4	0.4895	0.516	0.4954	0.4954
1	1	1	1	2	Jaccard	identity	0.7173	0.6622	0.6778	0.7173	0.63334	0.7173	0.7173
1	1	1	1	2	Jaccard	frequency	0.7758	0.7797	0.801	0.8297	0.7085	0.4503	0.7946
2	1	1	1	1	l1norm	identity	0.5292	0.4543	0.4779	0.5292	0.4279	0.5292	0.5292
2	1	1	1	1	l1norm	frequency	0.3801	0.3865	0.3884	0.3441	0.4049	0.327	0.3462
2	1	1	1	1	l2norm	identity	0.5292	0.4543	0.4779	0.5292	0.4279	0.5292	0.5292
2	1	1	1	1	l2norm	frequency	0.3267	0.3302	0.3989	0.3952	0.365	0.4905	0.4082
2	1	1	1	1	cosine	identity	0.5883	0.4894	0.5332	0.5883	0.4238	0.5883	0.5883
2	1	1	1	1	cosine	frequency	0.3975	0.4212	0.2936	0.3719	0.4092	0.3975	0.3975
2	1	1	1	1	Jaccard	identity	0.6124	0.5107	0.5674	0.6124	0.4746	0.6124	0.6124
2	1	1	1	1	Jaccard	frequency	0.6522	0.6894	0.6577	0.6805	0.7733	0.3953	0.7603
2	1	0	0	1	l1norm	identity	0.3598	0.3266	0.3789	0.3598	0.342	0.3598	0.3598
2	1	0	0	1	l1norm	frequency	0.32	0.265	0.292	0.293	0.3273	0.3261	0.3927
2	1	0	0	1	l2norm	identity	0.3598	0.3266	0.3786	0.3598	0.342	0.3598	0.3598
2	1	0	0	1	l2norm	frequency	0.2462	0.2176	0.2212	0.2424	0.301	0.5029	0.2428
2	1	0	0	1	cosine	identity	0.6227	0.582	0.5585	0.6227	0.4993	0.6227	0.6227
2	1	0	0	1	cosine	frequency	0.4373	0.4272	0.472	0.4318	0.4568	0.4373	0.4373
2	1	0	0	1	Jaccard	identity	0.4942	0.5154	0.5246	0.4942	0.4661	0.4942	0.4942
2	1	0	0	1	Jaccard	frequency	0.3973	0.4978	0.5277	0.5202	0.6729	0.3768	0.5701
						ColumnAv.	0.498	0.466	0.4865	0.5098	0.4762	0.4919	0.52

1. Best distance method is without confusion **Jaccard** in both *Corpus Week* and *Corpus Month*.

2. The best distance-weight combination is **Jaccard-Frequency** when average **P1** values and the average **Inv. Rank** values are considered. However, when average **Lin1** values are considered best combination of distance-weight functions are **Jaccard-Frequency** and in some cases **Jaccard-Identity**.

3. In addition to the window method with parameters **LW 1, RW 1, KD 1, KP 1, BL 2**, window method with **LW 2, RW 1, KD 1, KP 1, BL 1** and **LW 1, RW 1, KD 1, KP 0, BL 1** have also better results in terms of semantic similarity.

4. When considering **ColumnAv.** values only **M4** have better results than **M1** values in both *Corpus Week* and *Corpus Month*, however when best distance measure method based analysis is made other reprocess methods have better results than M1 results in most of the time.

5. *Corpus Month* results are better than *Corpus Week* results however improvement is more with the new approach in *Corpus Week*.

5.6.4 Detailed Average Jcn1 Results

When average **Jcn1** results are analyzed from Table 5.13 and Table 5.14 following information can be extracted:

1. Best distance method is without confusion **Jaccard** in both *Corpus Week* and *Corpus Month*.

2. The best distance-weight combination is without confusion **Jaccard-Frequency**.

3. In most of the cases new approach produces better results compared to the M1 results in both *Corpus Week* and *Corpus Month*.

4. M2, M3 and M4 are the most promising reprocess methods according to the **Jcn1** results.

5. In general *Corpus Week* has better results, but in best window-distance-weight function combination *Corpus Month* has better results.

Table 5.11: Detailed Average Lin1 Results in *Week Corpus*

						RP	M1	M2	M3	M4	M5	M6	M7
LW	RW	KD	KP	BL	Method	Weight	Av.P1	Av.P1	Av.P1	Av.P1	Av.P1	Av.P1	Av.P1
1	1	1	0	1	l1norm	identity	0.242	0.1992	0.2383	0.242	0.2199	0.242	0.242
1	1	1	0	1	l1norm	frequency	0.2259	0.2361	0.2234	0.2603	0.2489	0.2817	0.2065
1	1	1	0	1	l2norm	identity	0.242	0.1992	0.2383	0.242	0.2199	0.242	0.242
1	1	1	0	1	l2norm	frequency	0.2667	0.2653	0.2667	0.3193	0.2296	0.3365	0.2231
1	1	1	0	1	cosine	identity	0.3975	0.4479	0.3919	0.3975	0.2558	0.3975	0.3975
1	1	1	0	1	cosine	frequency	0.2816	0.2453	0.2558	0.2888	0.2502	0.2816	0.2816
1	1	1	0	1	Jaccard	identity	0.4285	0.4547	0.505	0.4285	0.4226	0.4285	0.4285
1	1	1	0	1	Jaccard	frequency	0.5548	0.4957	0.4592	0.5225	0.4837	0.2885	0.5064
1	1	1	1	2	l1norm	identity	0.1979	0.2197	0.2197	0.1979	0.1917	0.1979	0.1979
1	1	1	1	2	l1norm	frequency	0.2226	0.1966	0.1792	0.2345	0.1776	0.2046	0.178
1	1	1	1	2	l2norm	identity	0.1979	0.2197	0.2197	0.1979	0.1917	0.1979	0.1979
1	1	1	1	2	l2norm	frequency	0.2616	0.2843	0.3045	0.286	0.335	0.3203	0.3219
1	1	1	1	2	cosine	identity	0.3816	0.4048	0.3998	0.3816	0.385	0.3816	0.3816
1	1	1	1	2	cosine	frequency	0.2886	0.3418	0.348	0.2886	0.3651	0.2886	0.2886
1	1	1	1	2	Jaccard	identity	0.4007	0.5211	0.5083	0.4007	0.397	0.4007	0.4007
1	1	1	1	2	Jaccard	frequency	0.5235	0.5072	0.4565	0.5114	0.5184	0.3433	0.4899
2	1	1	1	1	l1norm	identity	0.1609	0.1482	0.1742	0.1609	0.2104	0.1609	0.1609
2	1	1	1	1	l1norm	frequency	0.2291	0.2306	0.2264	0.2517	0.2024	0.2494	0.2055
2	1	1	1	1	l2norm	identity	0.1609	0.1482	0.1742	0.1609	0.2104	0.1609	0.1609
2	1	1	1	1	l2norm	frequency	0.2705	0.2752	0.2713	0.2923	0.2788	0.3248	0.2717
2	1	1	1	1	cosine	identity	0.4362	0.4062	0.3843	0.4362	0.4523	0.4362	0.4362
2	1	1	1	1	cosine	frequency	0.3112	0.3246	0.2741	0.3092	0.3347	0.3112	0.3112
2	1	1	1	1	Jaccard	identity	0.5087	0.4735	0.4362	0.5087	0.308	0.5087	0.5087
2	1	1	1	1	Jaccard	frequency	0.5298	0.4871	0.4942	0.4754	0.4927	0.3035	0.4888
2	1	0	0	1	l1norm	identity	0.2717	0.235	0.2328	0.2717	0.2416	0.2717	0.2717
2	1	0	0	1	l1norm	frequency	0.2526	0.22	0.1987	0.237	0.2379	0.2569	0.2371
2	1	0	0	1	l2norm	identity	0.2717	0.235	0.2328	0.2717	0.2416	0.2717	0.2717
2	1	0	0	1	l2norm	frequency	0.2468	0.243	0.2383	0.2165	0.2813	0.3053	0.2708
2	1	0	0	1	cosine	identity	0.3477	0.3138	0.3127	0.3477	0.3853	0.3477	0.3477
2	1	0	0	1	cosine	frequency	0.3097	0.3831	0.3142	0.3097	0.295	0.3097	0.3097
2	1	0	0	1	Jaccard	identity	0.4846	0.4489	0.409	0.4846	0.4016	0.4846	0.4846
2	1	0	0	1	Jaccard	frequency	0.4547	0.4049	0.4677	0.4437	0.3519	0.3124	0.4163
						ColumnAv.	0.3238	0.3192	0.3142	0.3243	0.3069	0.3077	0.3168

Table 5.12: Detailed Average Lin1 Results in *Month Corpus*

						RP	M1	M2	M3	M4	M5	M6	M7
LW	RW	KD	KP	BL	Method	Weight	Av.P1	Av.P1	Av.P1	Av.P1	Av.P1	Av.P1	Av.P1
1	1	1	0	1	l1norm	identity	0.2238	0.2238	0.2986	0.2238	0.3165	0.2238	0.2238
1	1	1	0	1	l1norm	frequency	0.2707	0.287	0.313	0.3218	0.3165	0.4524	0.3053
1	1	1	0	1	l2norm	identity	0.2238	0.2238	0.2986	0.2238	0.3165	0.2238	0.2238
1	1	1	0	1	l2norm	frequency	0.2685	0.2277	0.2543	0.3009	0.1899	0.3782	0.2546
1	1	1	0	1	cosine	identity	0.5665	0.3984	0.4036	0.5665	0.4489	0.5665	0.5665
1	1	1	0	1	cosine	frequency	0.374	0.3244	0.2481	0.3668	0.2899	0.374	0.374
1	1	1	0	1	Jaccard	identity	0.515	0.4536	0.4958	0.515	0.4318	0.515	0.515
1	1	1	0	1	Jaccard	frequency	0.4674	0.4879	0.5157	0.5896	0.533	0.3548	0.5573
1	1	1	1	2	l1norm	identity	0.2795	0.2557	0.2728	0.2795	0.2818	0.2795	0.2795
1	1	1	1	2	l1norm	frequency	0.2848	0.2405	0.3086	0.2828	0.275	0.4128	0.3089
1	1	1	1	2	l2norm	identity	0.2795	0.2557	0.2728	0.2795	0.2818	0.2795	0.2795
1	1	1	1	2	l2norm	frequency	0.298	0.3256	0.3334	0.3294	0.2849	0.3911	0.316
1	1	1	1	2	cosine	identity	0.5724	0.4607	0.4362	0.5724	0.5071	0.5724	0.5724
1	1	1	1	2	cosine	frequency	0.3021	0.3065	0.3431	0.3021	0.3663	0.3021	0.3021
1	1	1	1	2	Jaccard	identity	0.5254	0.4535	0.5097	0.5254	0.4846	0.5254	0.5254
1	1	1	1	2	Jaccard	frequency	0.5258	0.5498	0.5293	0.5673	0.5102	0.3979	0.5427
2	1	1	1	1	l1norm	identity	0.2463	0.2612	0.2435	0.2463	0.2641	0.2463	0.2463
2	1	1	1	1	l1norm	frequency	0.2691	0.3068	0.2808	0.2717	0.3289	0.3846	0.2951
2	1	1	1	1	l2norm	identity	0.2463	0.2612	0.2435	0.2463	0.2641	0.2463	0.2463
2	1	1	1	1	l2norm	frequency	0.2779	0.2854	0.3321	0.3603	0.2601	0.3813	0.2516
2	1	1	1	1	cosine	identity	0.4522	0.3957	0.4023	0.4522	0.4074	0.4522	0.4522
2	1	1	1	1	cosine	frequency	0.3545	0.4128	0.325	0.3545	0.3276	0.3545	0.3545
2	1	1	1	1	Jaccard	identity	0.4957	0.3708	0.4165	0.4957	0.3189	0.4957	0.4957
2	1	1	1	1	Jaccard	frequency	0.538	0.5795	0.5513	0.5505	0.5036	0.3506	0.5554
2	1	0	0	1	l1norm	identity	0.2651	0.2326	0.2772	0.2651	0.2506	0.2651	0.2651
2	1	0	0	1	l1norm	frequency	0.2713	0.2	0.2449	0.2451	0.2333	0.2545	0.2803
2	1	0	0	1	l2norm	identity	0.2651	0.2326	0.2772	0.2651	0.2506	0.2651	0.2651
2	1	0	0	1	l2norm	frequency	0.2226	0.3015	0.2748	0.2383	0.321	0.3908	0.2987
2	1	0	0	1	cosine	identity	0.4818	0.3652	0.3353	0.4818	0.415	0.4818	0.4818
2	1	0	0	1	cosine	frequency	0.3606	0.454	0.4746	0.3419	0.4518	0.3606	0.3606
2	1	0	0	1	Jaccard	identity	0.5164	0.4593	0.3836	0.5164	0.4136	0.5164	0.5164
2	1	0	0	1	Jaccard	frequency	0.3189	0.3844	0.385	0.4851	0.4548	0.3682	0.4831
						ColumnAv.	0.3612	0.3430	0.3525	0.3770	0.3531	0.3770	0.3748

Table 5.13: Detailed Average Jcn1 Results in *Week Corpus*

						RP	M1	M2	M3	M4	M5	M6	M7
LW	RW	KD	KP	BL	Method	Weight	Av.P1	Av.P1	Av.P1	Av.P1	Av.P1	Av.P1	Av.P1
1	1	1	0	1	l1norm	identity	428937290	214468645	214468645	428937290	214468645	428937290	428937290
1	1	1	0	1	l1norm	frequency	214468645	214468645	214468645	214468645	214468645	0.51384583	214468645
1	1	1	0	1	l2norm	identity	428937290	214468645	214468645	428937290	214468645	428937290	428937290
1	1	1	0	1	l2norm	frequency	214468645	214468645	214468645	214468645	0.10585	0.53602917	214468645
1	1	1	0	1	cosine	identity	0.54487083	643405936	428937291	0.54487083	0.08550417	0.54487083	0.54487083
1	1	1	0	1	cosine	frequency	214468645	214468645	214468645	214468645	0.078175	214468645	214468645
1	1	1	0	1	Jaccard	identity	214468646	643405936	857874581	214468646	428937291	214468646	214468646
1	1	1	0	1	Jaccard	frequency	428937291	428937291	428937291	428937291	214468646	0.1005875	214468646
1	1	1	1	2	l1norm	identity	214468645	214468645	214468645	214468645	214468645	214468645	214468645
1	1	1	1	2	l1norm	frequency	214468645	214468645	214468645	214468645	214468645	0.4855625	214468645
1	1	1	1	2	l2norm	identity	214468645	214468645	214468645	214468645	214468645	214468645	214468645
1	1	1	1	2	l2norm	frequency	214468645	214468645	214468645	214468645	214468645	0.52694583	214468645
1	1	1	1	2	cosine	identity	0.54485	214468646	214468646	0.54485	643405935	0.54485	0.54485
1	1	1	1	2	cosine	frequency	214468645	214468645	214468645	214468645	214468645	214468645	214468645
1	1	1	1	2	Jaccard	identity	0.573325	643405936	428937291	0.573325	428937290	0.573325	0.573325
1	1	1	1	2	Jaccard	frequency	428937291	428937291	428937291	428937291	428937290	0.10761667	214468646
2	1	1	1	1	l1norm	identity	214468645	0.06185833	0.07572083	214468645	214468645	214468645	214468645
2	1	1	1	1	l1norm	frequency	214468645	214468645	214468645	214468645	214468645	0.5108625	214468645
2	1	1	1	1	l2norm	identity	214468645	0.06	0.07	214468645	428937291	214468645	214468645
2	1	1	1	1	l2norm	frequency	214468645	214468645	0.11	214468645	0.10	0.536	0.10
2	1	1	1	1	cosine	identity	0.55	214468645	0.55	0.55	214468646	0.55	0.55
2	1	1	1	1	cosine	frequency	214468645	214468645	214468645	214468645	0.106	214468645	214468645
2	1	1	1	1	Jaccard	identity	0.78	428937291	428937291	0.78	214468645	0.78	0.78
2	1	1	1	1	Jaccard	frequency	428937291	428937291	428937291	428937291	428937291	0.103	214468645
2	1	0	0	1	l1norm	identity	428937291	214468645	214468645	428937291	214468645	428937291	428937291
2	1	0	0	1	l1norm	frequency	214468645	0.09	0.08	0.09	0.52	0.102	0.513
2	1	0	0	1	l2norm	identity	428937291	214468645	214468645	428937291	214468645	428937291	428937291
2	1	0	0	1	l2norm	frequency	0.09	0.09	0.09	0.09	0.52	0.52	0.52
2	1	0	0	1	cosine	identity	0.55	214468645	214468645	0.55	214468645	0.55	0.55
2	1	0	0	1	cosine	frequency	214468645	214468645	0.11	214468645	214468645	214468645	214468645
2	1	0	0	1	Jaccard	identity	214468645	428937291	428937291	214468645	428937291	214468645	214468645
2	1	0	0	1	Jaccard	frequency	214468645	428937291	428937291	428937291	428937291	0.11	214468645
						ColumnAv.	274519866	306383779	308834849	285958194	291677357	275745401	251767540

6. In addition to the window method with parameters **LW 1, RW 1, KD 1, KP 1, BL 2**, window method with **LW 1, RW 1, KD 1, KP 0, BL 1** have also better results in terms of semantic similarity.

7. When considering **ColumnAv.** values **M6** produces the worst results.

8. *Corpus Month* results are better than *Corpus Week* results.

9. **Jcn** can give very big numbers and very small numbers when comparing two words. Therefore, analyzing the values of **Jcn** in Table 5.13 and Table 5.14 is not so easy and may lead misinterpretations.

10. With these **Jcn1** values how many **exactly similar words with the related headword** found in each combination of methods can be found. From this information best combination is in *Corpus Month*. In this combination window method is consisting of **LW 1, RW 1, KD 1, KP 1, BL 2** parameters and **Jaccard-Frequency** distance-weight methods are used and **M1, M2, M3, M4** reprocess methods produces the best results.

Table 5.14: Detailed Average Jcn1 Results in *Month Corpus*

						RP	M1	M2	M3	M4	M5	M6	M7
LW	RW	KD	KP	BL	Method	Weight	Av.P1	Av.P1	Av.P1	Av.P1	Av.P1	Av.P1	Av.P1
1	1	1	0	1	l1norm	identity	214468645	214468645	214468645	214468645	214468645	214468645	214468645
1	1	1	0	1	l1norm	frequency	214468645	0.1187	0.1271	214468645	0.1255	0.5922	0.1271
1	1	1	0	1	l2norm	identity	214468645	214468645	214468645	214468645	214468645	214468645	214468645
1	1	1	0	1	l2norm	frequency	214468645	0.1086	0.1115	214468645	0.0839	0.5512	0.1126
1	1	1	0	1	cosine	identity	214468646	0.6138	214468646	214468646	0.5756	214468646	214468646
1	1	1	0	1	cosine	frequency	214468645	0.1276	0.1012	214468645	0.123	214468645	214468645
1	1	1	0	1	Jaccard	identity	214468646	643405936	428937291	214468646	428937291	214468646	214468646
1	1	1	0	1	Jaccard	frequency	214468646	428937291	643405936	643405936	428937291	0.1108	428937291
1	1	1	1	2	l1norm	identity	214468645	214468645	214468645	214468645	214468645	214468645	214468645
1	1	1	1	2	l1norm	frequency	214468645	0.1009	0.1143	214468645	0.1078	0.5969	0.1261
1	1	1	1	2	l2norm	identity	214468645	214468645	214468645	214468645	214468645	214468645	214468645
1	1	1	1	2	l2norm	frequency	214468645	214468645	214468645	214468645	214468645	0.1348	214468645
1	1	1	1	2	cosine	identity	214468646	214468646	0.6221	214468646	0.6562	214468646	214468646
1	1	1	1	2	cosine	frequency	214468645	214468645	214468645	214468645	0.1372	214468645	214468645
1	1	1	1	2	Jaccard	identity	214468646	643405936	643405936	214468646	214468646	214468646	214468646
1	1	1	1	2	Jaccard	frequency	643405936	643405936	643405936	643405936	428937291	0.1166	428937291
2	1	1	1	1	l1norm	identity	214468645	214468645	0.1053	214468645	0.1113	214468645	214468645
2	1	1	1	1	l1norm	frequency	214468645	0.1222	0.1205	0.1016	0.1315	0.1474	0.1262
2	1	1	1	1	l2norm	identity	214468645	214468645	0.1053	214468645	0.1113	214468645	214468645
2	1	1	1	1	l2norm	frequency	214468645	0.1573	0.178	214468645	0.1421	0.5467	0.1132
2	1	1	1	1	cosine	identity	0.5997	214468646	214468646	0.5997	0.5782	0.5997	0.5997
2	1	1	1	1	cosine	frequency	214468645	0.1578	0.1371	214468645	0.1341	214468645	214468645
2	1	1	1	1	Jaccard	identity	0.6555	428937291	428937291	0.6555	0.5777	0.6555	0.6555
2	1	1	1	1	Jaccard	frequency	428937291	428937291	428937291	428937291	428937291	0.1106	428937291
2	1	0	0	1	l1norm	identity	0.1733	0.1443	0.1848	0.1733	0.1827	0.1733	0.1733
2	1	0	0	1	l1norm	frequency	0.101	0.0924	0.1011	0.1161	0.0974	0.1032	0.5468
2	1	0	0	1	l2norm	identity	0.1733	0.1443	0.1848	0.1733	0.1827	0.1733	0.1733
2	1	0	0	1	l2norm	frequency	0.1196	0.1277	0.1176	0.1043	214468645	0.5482	0.1409
2	1	0	0	1	cosine	identity	0.2321	0.2145	428937291	0.2321	214468646	0.2321	0.2321
2	1	0	0	1	cosine	frequency	214468645	214468645	214468645	214468645	214468645	214468645	214468645
2	1	0	0	1	Jaccard	identity	0.6488	428937291	643405936	0.6488	643405936	0.6488	0.6488
2	1	0	0	1	Jaccard	frequency	0.555	214468646	214468646	428937291	0.1149	0.1149	0.6421
						ColumnAv.	174255774	194362209	201064355	194362209	134042903	93830032	140745048

5.7 DISCUSSION

In this section findings of sections 5.2, 5.3, 5.4, 5.5 and 5.6 are going to be discussed and several rules are offered. These rules can be listed as following:

- In general (i.e. in most of the cases) as the corpus size increases the quality of the results also increase. From this information *Corpus Month* has better results than *Corpus Week*. However, there is a trade off between the corpus size and the required resources, such as necessary time and memory for processing.

- As the headword frequency increases in the corpus the success of the found list in terms of semantic similarity increases.

- Without any confusion **Jaccard** is the best semantic distance measure function among the distance measure methods L1Norm, L2Norm, Cosine and Jaccard. This is not only expected from the experiments of [10] but also it approves the correctness of these experiments.

- Most of the time best measure-weight method combination is **Jaccard-Frequency**.
- Window method $W(L_1R_1)$ has better results compared to the others.
- In general **Frequency** weight function have better results compared to the **Identity** weight function.
- In terms of average **P1** and **Inv. Rank** values proposed new approach has better results especially in *Corpus Week*.
- In most of the cases if the FP results are better, proposed approach produces even better results.
- **M6** is the most instable (i.e it sometimes produces very good results but sometimes very poor) proposed method among 6 different proposed reprocess methods and this is an expected results.
- Among all reprocess methods M2, M3 and M4 and M5 produces better and stable (i.e. most of the time provide good) results. Especially with best window-distance-weight method combinations **M2** and **M3** methods provide very good results.
- When all cases are considered the best window-distance-weight combination is window with parameters **LW1, RW1, KP1, KD1, BL2** and **Jaccard-Frequency** distance-weight method combination.
- When best window-distance-weight combinations are considered in most of the cases proposed method produces better results than M1 results.
- New proposed reprocess approach generate best improvement when best window-distance-weight combination is used with **Corpus Week**. This is a very important result because new approach not only improves the best producible results but also does this with small sized corpus. Small sized corpus means less computer and time resources.
- Improvements with the new approach are in considerable amounts especially in *Week Corpus*.
- Improvements with the new approach in rare headwords in terms of low occurrence frequencies are greater than common headwords. This is also expected and a crucial enhancement. By making use of the proposed approach even with rare headwords more similar words can be found.

- This new reprocessing approach is also a solution to the ***data sparseness issue***, which is considered as a common problem for Statistical Corpus based approaches. Since rare headword vectors contain more 0 (empty) values (i.e. this vectors data is sparse), the distance measure functions can not produce promising results. After reprocess of the existing results, rare headword vectors resembles vectors of common headwords. This solves the data sparseness problem. Therefore, improvements especially on the rare headwords are due to the provided solution for the data sparseness problem.

From all these the most crucial enhancement of reprocess approach is ***even with small corpora and rare headwords, better results in terms of semantic word similarity can be found***.

CHAPTER 6

CONCLUSION

6.1 SUMMARY OF THE STUDY

In this thesis semantically similar words are seeked with completely unsupervised methods. Not only the existing semantic distance measure methods are used but also a novel technique is introduced. The success of the results are evaluated using the human-made sources such as WordNet ([3]).

Novel technique is a kind of reprocess mechanism in which the outputs of the existing methods are used to update the representation of related headword vectors to improve the results further. Moreover, this novel technique provides a solution to the data sparseness problem in headword vectors (especially for the vectors of rare headwords) which is considered as a common problem for Statistical Corpus based approaches. In addition to this novel approach, some of latest improvements from different related studies in the literature are merged to come up with better solution.

A large number of experiments have been made (70.000 cases with 25 evaluated values for each). The experimental results are analyzed in general and in specific conditions. Existing methods are compared with the new introduced ones. According to these evaluations some rules are proposed.

From the results of the experiments, proposed novel approach has promising results. Moreover since new approach introduces a very new technique in terms of semantic similarity, it is open for future improvements.

6.2 FUTURE WORK

This new approach has promising results and the results can be further improved with the following future work:

- Reprocess ratio values can be improved. This can be done by selecting some portion of the corpus as training data and the remaining portion as the test data. In every iteration, the reprocess values are changed and the best reprocess ratios can be found.
- Third and forth pass results can be evaluated and these may have even better results. Stopping iteration when the results converge to a static word list may be an option.
- Reprocess ratio values can be defined considering the found semantic distance values from distance measure functions.
- Other method-weight combinations can be used.
- According to the studies in the literature, the type of the window may also be selected according to the headword type (whether it is a **noun**, **adjective**, **adverb** or **verb**). For example, if the headword is an *adjective* then window which contains the words on the right side of the headword may have better results than the left side. Because the adjectives are semantically related with the their right near nouns such as ***old*** *building*, ***red*** *car*... If the headword is a *noun* then a wide window is expected to have better results because the semantic meaning of the nouns are related with the nouns which may not very close to the headword. For example, if a paragraph is about *hospital* and if the headword is *doctor*, the other nouns in this paragraph most probably related with the *doctor*. Let *"Nurse got the patient. Hospital was very cold. Doctor was coming to the surgery room."* is given. In such a sentence the headword *doctor* is semantically related with the words *hospital*, *nurse*, *patient* although some of them are far away from the word *doctor*. For the other headword types (*verb* or *adverb*) such rules can be defined to improve the results. However, knowing the type of the headword from human made resources will broke the golden rule ***"every thing will be found automatically and in an unsupervised manner"***.
- Pre-process phase can be improved. Actually using a good Part-Of-Speech Tagger may improve the results.

- Other improvements in the literature, such as in [16], can be integrated. For example, word order in the resulted word list can be improved from **LCS string matching algorithm**.
- Other than **RCV1** ([11]) different corpora can be used such as BNC ([12]), which has less noise.
- The results can be used in different domains such as WSD, Semantic Web or TOEFL question answering (section 2.2 contains detailed information about application areas).

REFERENCES

[1] ResearchOver Helping to Research More Efficiently. Language as a living organism. http://www.researchover.com/termpaper/Language_as_a_Living_Organism-107350.html, Last access 2009-07-22.

[2] Y.E. Esin, O. Alan and F. N. Alpaslan. Improvement on corpus-based word similarity using vector space models. METU Ankara, Turkey, 2009. ISCIS - 2009.

[3] C. Fellbaum. *WordNet: An Electronic Lexical Database (Language, Speech, and Communication)*. The MIT Press, 2007. http://wordnet.princeton.edu/.

[4] Wikipedia The Free Encyclopedia. Library of congress subject headings. http://en.wikipedia.org/wiki/Library_of_Congress_Subject_Headings, Last access 2009-07-13.

[5] E. Altintas, E. Karsligil and V. Coskun. A new semantic similarity measure evaluated in word sense disambiguation, 2005. http://phon.joensuu.fi/nodalida/index.shtml.

[6] P. Bhattacharyya and N Unny. Word sense disambiguation and text similarity measurement using wordnet. *In 1st International Conference on Global WordNet, Mysore, India*, 2002.

[7] Y. Bollegala, D. Matsuo and M. Ishizuka. Measuring semantic similarity between words using web search engines. In *WWW '07: Proceedings of the 16th international conference on World Wide Web*, pages 757–766, New York, NY, USA, 2007. ACM.

[8] Elias Iosif and Alexandros Potamianos. Unsupervised semantic similarity computation usingweb search engines. In *WI '07: Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence*, pages 381–387, Washington, DC, USA, 2007. IEEE Computer Society.

[9] D. Lin. Automatic retrieval and clustering of similar words. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, Volume 2*, pages 768–774, Montreal, Quebec, Canada, 1998. Association for Computational Linguistics. http://www.aclweb.org/anthology/P98-2005.

[10] J. Curran. *From Distributional to Semantic Similarity*. PhD thesis, Institute for Communicating and Collaborative Systems, School of Informatics, University of Edinburgh, 2004.

[11] Reuters. Reuters corpus, volume 1. volume 1. NIST, 1996–1997. http://trec.nist.gov/data/reuters/reuters.html, Last access 2009-06-24.

[12] Oxford University Computing Services. The british national corpus. University of Oxford. http://www.natcorp.ox.ac.uk/corpus/index.xml, Last access 2009-06-24.

[13] E. Terra and C. L. A. Clarke. Frequency estimates for statistical word similarity measures. In *NAACL '03: Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pages 165–172, Morristown, NJ, USA, 2003. Association for Computational Linguistics. http://dx.doi.org/10.3115/1073445.1073477.

[14] Courtney Corley and Rada Mihalcea. Measuring the semantic similarity of texts. In *Proceedings of the ACL Workshop on Empirical Modeling of Semantic Equivalence and Entailment*, pages 13–18, Ann Arbor, Michigan, June 2005. Association for Computational Linguistics.

[15] R. Mihalcea, C. Corley and C. Strapparava. Corpus-based and knowledge-based measures of text semantic similarity. In *AAAI06*, pages 775–780, 2006. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.65.3690.

[16] A. Islam and D Inkpen. Semantic text similarity using corpus-based word similarity and string similarity. *ACM Trans. Knowl. Discov. Data*, 2:1–25, 2008.

[17] X. Achananuparp, P. Hu and X. Shen. The evaluation of sentence similarity measures. In *DaWaK '08: Proceedings of the 10th international conference on Data Warehousing and Knowledge Discovery*, pages 305–316, Berlin, Heidelberg, 2008. Springer-Verlag. http://dx.doi.org/10.1007/978-3-540-85836-2_29.

[18] Issei Sato, Minoru Yoshida, and Hiroshi Nakagawa. Knowledge discovery of semantic relationships between words using nonparametric bayesian graph model. In *KDD '08: Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 587–595, New York, NY, USA, 2008. ACM.

[19] Lillian Lee. Measures of distributional similarity. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*, pages 25–32, Morristown, NJ, USA, 1999. Association for Computational Linguistics.

[20] Pan J Cheng CP, Lau GT. Domain-specific ontology mapping by corpus-based semantic similarity. In *Proceedings of 2008 NSF CMMI engineering research and innovation conference*, Knoxville, TN, USA, 2008.

[21] Dequan Zheng, Tiejun Zhao, Sheng Li, and Hao Yu. Research on a novel word co-occurrence model and its application. pages 437–446. 2007.

[22] Evgeniy Gabrilovich and Shaul Markovitch. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *In Proceedings of the 20th International Joint Conference on Artificial Intelligence*, pages 1606–1611, 2007.

[23] K. Chen and J. You. A study on word similarity using context vector models. In *International Journal of Computational Linguistics and Chinese Language Processing*, pages 37–58, 2002.

[24] Yves Peirsman. Word space models of semantic similarity and relatedness. In *In Proceedings of the ESSLLI-2008 Student Session*, Hamburg, Germany, 2008.

[25] P. W. Landauer, T. K. Foltz and D. Laham. An introduction to latent semantic analysis. *Discourse Processes*, 25(2):259–284, 1998.

[26] University of Colorado at Boulder. Latent semantic analysis. http://lsa.colorado.edu/, Last access 2009-07-23.

[27] J.A. Bullinaria and J.P. Levy. Extracting semantic representations from word co-occurrence statistics. In *A Computational Study. Behavior Research Methods*, pages 510–526, 2007.

[28] Joseph P. Levy and John A. Bullinaria. Learning lexical properties from word usage patterns: Which context words should be used? In *Connectionist Models of Learning Development and Evolution: Proceedings of the 6 th Neural Computation and Psychology Works hop*, pages 273–282. Springer, 2001.

[29] J. J. Jiang and W. C. David. Semantic similarity based on corpus statistics and lexical taxonomy. *CoRR*, 1997. http://arxiv.org/abs/cmp-lg/9709008.

[30] Gregory Grefenstette. Sextant: Exploring unexplored contexts for semantic extraction from syntactic analysis. In *in Proceedings of the 30st annual meeting of the Association for Computational Linguistics, ACL*, pages 324–326, 1992.

[31] H. Al-Mubaid and P. Chen. Context-based similar words detection and its application in specialized search engines. In *IUI '05: Proceedings of the 10th international conference on Intelligent user interfaces*, pages 260–262, New York, NY, USA, 2005. ACM. http://doi.acm.org/10.1145/1040830.1040890.

[32] Peter D. Turney. A uniform approach to analogies, synonyms, antonyms, and associations. *CoRR*, abs/0809.0124, 2008.

[33] Eneko Agirre, Enrique Alfonseca, Keith Hall, Jana Kravalova, Mariusand Pasca, and Aitor Soroa. A study on similarity and relatedness using distributional and wordnet-based approaches. 2009.

[34] Diana Inkpen. Semantic similarity knowledge and its applications. volume 0, pages 9–10, Cluj-Napoca (Romania), 2007.

[35] David Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. In *In Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, pages 189–196, 1995.

[36] Y. Karov and S. Edelman. Similarity-based word sense disambiguation. Technical report, Jerusalem, Israel, Israel, 1996.

[37] Gina Levow. Corpus-based techniques for word sense disambiguation. Technical report, Cambridge, MA, USA, 1997.

[38] Ted Pedersen, Satanjeev Banerjee, and Siddharth Patwardhan. Maximizing semantic relatedness to perform word sense disambiguation, 2003.

[39] David Blei Jordan Boyd-Graber and Xiaojin Zhu. A topic model forword sense disambiguation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1024–1033. Association for Computational Linguistics, June 2007.

[40] Ravi Sinha and Rada Mihalcea. Unsupervised graph-basedword sense disambiguation using measures of word semantic similarity. In *ICSC '07: Proceedings of the International Conference on Semantic Computing*, pages 363–369, Washington, DC, USA, 2007. IEEE Computer Society.

[41] Satanjeev Banerjee Siddharth Patwardhan and Ted Pedersen. Unsupervised word sense disambiguation using contextual semantic relatedness. In *Proceedings of the 4th International Workshop on Semantic Evaluations*, Prague, 2007. Association for Computational Linguistics.

[42] W. Bowes C. Chen, P. Ding and David Brown. A fully unsupervised word sense disambiguation method using dependency knowledge. In *Human Language Technologies The 2009 Annual Conference of the North American Chapter of the ACL*, Boulder, Colorado, 2009. Association for Computational Linguistics.

[43] Xinglong Wang and John Carroll. Word sense disambiguation using sense examples automatically acquired from a second language. In *HLT '05: Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 547–554, Morristown, NJ, USA, 2005. Association for Computational Linguistics.

[44] Diana McCarthy. Relating wordnet senses for word sense disambiguation. *In Proc. of ACL Workshop on Making Sense of Sense*, 2006.

[45] Ioannis P. Klapaftis and Suresh Manandhar. Unsupervised word sense disambiguation using the www. *Proceedings of the ECAI Starting AI Researcher Symposium - STAIRS, Riva del Garda, Italy*, 2006.

[46] Yee Seng Chan, Hwee Tou Ng, and David Chiang. Word sense disambiguation improves statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 33–40, Prague, Czech Republic, June 2007. Association for Computational Linguistics.

[47] Graeme Hirst Saif Mohammad and Philip Resnik. Tor, tormd: Distributional profiles of concepts for unsupervised word sense disambiguation. In *Proceedings of the 4th International Workshop on Semantic Evaluations*, Prague, 2007. Association for Computational Linguistics.

[48] Upali S. Kohomban and Wee Sun Lee. Learning semantic classes for word sense disambiguation. In *ACL '05: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 34–41, Morristown, NJ, USA, 2005. Association for Computational Linguistics.

[49] Google, 2009. http://www.google.com.tr/help/features.html, Last access 2009-07-10.

[50] Ralf Schenkel, Anja Theobald, and Gerhard Weikum. Semantic similarity search on semistructured data with the xxl search engine. *Inf. Retr.*, 8(4):521–545, 2005.

[51] King-Kup Liu, Weiyi Meng, and Clement Yu. Discovery of similarity computations of search engines. In *CIKM '00: Proceedings of the ninth international conference on Information and knowledge management*, pages 290–297, New York, NY, USA, 2000. ACM.

[52] A. Gulli and A. Signorini. Building an open source meta-search engine. In *WWW '05: Special interest tracks and posters of the 14th international conference on World Wide Web*, pages 1004–1005, New York, NY, USA, 2005. ACM.

[53] James R. Curran. Blueprint for a high performance nlp infrastructure. In *SEALTS '03: Proceedings of the HLT-NAACL 2003 workshop on Software engineering and architecture of language technology systems*, pages 39–44, Morristown, NJ, USA, 2003. Association for Computational Linguistics.

[54] Dekang Lin. An information-theoretic definition of similarity. In *ICML '98: Proceedings of the Fifteenth International Conference on Machine Learning*, pages 296–304, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc.

[55] Stanley Kok and Pedro Domingos. Extracting semantic networks from text via relational clustering. In *ECML PKDD '08: Proceedings of the 2008 European Conference on Machine Learning and Knowledge Discovery in Databases - Part I*, pages 624–639, Berlin, Heidelberg, 2008. Springer-Verlag.

[56] Dekang Lin and Patrick Pantel. Discovery of inference rules for question-answering. *Nat. Lang. Eng.*, 7(4):343–360, 2001.

[57] Abdellali Kelil and Shengrui Wang. Scs: A new similarity measure for categorical sequences. *Data Mining, IEEE International Conference on*, 0:343–352, 2008.

[58] Shasha Xie and Yang Liu. Using corpus and knowledge-based similarity measure in maximum marginal relevance for meeting summarization. pages 4985–4988. Acoustics, Speech and Signal Processing, 2008. ICASSP 2008. IEEE International Conference on March 31 2008-April 4 2008, 2008.

[59] Jeff Mitchell and Mirella Lapata. Vector-based models of semantic composition. In *Proceedings of ACL-08: HLT*, pages 236–244, Columbus, Ohio, June 2008. Association for Computational Linguistics.

[60] Toefl synonym questions (aclwiki), 2009. http://www.aclweb.org/aclwiki/index.php?title=TOEFL_Synonym_Questions, Last access 2009-07-10.

[61] Wordnet::similarity, 2009. http://wn-similarity.sourceforge.net/, Last access 2009-07-10.

[62] Pure java wordnet similarity library, 2009. http://nlp.shef.ac.uk/result/software.html, Last access 2009-07-10.

[63] High performance computing. http://hpc.ceng.metu.edu.tr/, Last access 2009-07-12.

[64] Mindaugas Idzelis. The java open source spell checker. http://jazzy.sourceforge.net/, Last access 2009-06-24.

[65] Longman, editor. *Longman Dictionary of American English*. Pearson ESL; 3 edition, September 2004.

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

[67] Grady Ward. Moby thesaurus list, 2002. http://www.gutenberg.org/etext/3202, Last access 2009-07-12.

[68] Vision Media. Php common word list filter. http://vision-media.ca/resources/php/php-common-word-list-filter, Last access 2009-07-23.

APPENDIX A

STOP WORDS

Stop words ([68]) that are used in the pre-process phase of the corpora are available in Table A.1.

Table A.1: Stop Words

about	besides	five	latterly	ours	thereafter	whereupon
above	between	for	least	ourselves	thereby	wherever
across	beyond	former	less	out	therefore	whether
after	bill	formerly	ltd	over	therein	which
afterwards	both	forty	made	own	thereupon	while
again	bottom	found	many	part	these	whither
against	but	four	may	per	they	who
ago	by	from	me	perhaps	thick	whoever
all	call	front	meanwhile	please	thin	whole
almost	can	full	might	put	third	whom
alone	cannot	further	mill	rather	this	whose
along	cant	get	mine	re	those	why
already	co	give	more	same	though	will
also	computer	go	moreover	see	three	with
although	con	had	most	seem	through	within
always	could	has	mostly	seemed	throughout	without
am	couldnt	hasnt	move	seeming	thru	would
among	cry	have	much	seems	thus	yet
amongst	de	he	must	serious	to	you
amoungst	describe	hence	my	several	together	your
amount	detail	her	myself	she	too	yours
an	do	here	name	should	top	yourself
and	done	hereafter	namely	show	toward	yourselves
another	down	hereby	neither	side	towards	
any	due	herein	never	since	twelve	
anybody	during	hereupon	nevertheless	sincere	twenty	
anyhow	each	hers	next	six	two	
anyone	eg	herself	nine	sixty	un	
anything	eight	him	no	so	under	
anyway	either	himself	nobody	some	until	
anywhere	eleven	his	none	somehow	up	
are	else	how	noone	someone	upon	
around	elsewhere	however	nor	something	us	
as	empty	hundred	not	sometime	very	
at	enough	i	nothing	sometimes	via	
away	etc	ie	now	somewhere	was	
back	even	if	nowhere	still	we	
be	ever	in	of	such	well	
became	every	inc	off	system	were	
because	everyone	indeed	often	take	what	
become	everything	interest	on	ten	what's	
becomes	everywhere	into	once	than	whatever	
becoming	except	is	one	that	when	
been	few	it	only	the	whence	
before	fifteen	its	onto	their	whenever	
beforehand	fify	itself	or	them	where	
behind	fill	just	other	themselves	whereafter	
being	find	keep	others	then	whereas	
below	fire	last	otherwise	thence	whereby	
beside	first	latter	our	there	wherein	

APPENDIX B

WINDOW SETTINGS

Window settings used in the experiments are available in Table B.1. In this table *Left* means left word size, *Right* means right word size, *Keep Direction* means whether position information of the words in the window are kept or not and *Boundary Level* means whether paragraph or sentence boundaries are preserved or not while defining the words inside the window.

Table B.1: Window Settings

Left	Right	Keep Direction	KeepPosition	BoundaryLevel
1	1	FALSE	FALSE	1
1	1	TRUE	FALSE	1
1	1	FALSE	FALSE	2
1	1	TRUE	TRUE	2
2	0	FALSE	FALSE	1
2	0	FALSE	FALSE	2
2	0	FALSE	TRUE	1
2	0	TRUE	TRUE	2
1	0	FALSE	FALSE	1
1	0	FALSE	FALSE	2
1	0	FALSE	FALSE	0
2	0	FALSE	FALSE	0
0	1	FALSE	FALSE	1
0	1	FALSE	FALSE	2
0	1	FALSE	FALSE	0
2	0	TRUE	TRUE	0
2	1	FALSE	FALSE	1
2	1	FALSE	FALSE	0
2	1	FALSE	FALSE	2
2	1	TRUE	TRUE	1