Word Sense Disambiguation Applied to Information Retrieval

by

Ozlem Uzuner

Submitted to the Department of Electrical Engineering and Computer Science

in Partial Fulfillment of the Requirements for the Degrees of

Bachelor of Science in Computer Science and Engineering and

Master of Engineering in Electrical Engineering and Computer Science

at the Massachusetts Institute of Technology

May 27, 1998

© 1998 Ozlem Uzuner. All rights reserved.

The author hereby grants to M.I.T. permission to reproduce and
distribute publicly paper and electronic copies of this thesis

and to grant others the right to do so.

Author

Department of Electrical Engineering and Computer Science

May 27, 1998

Certified by

Boris Katz

Thesis Supervisor

Accepted by

Arthur C. Smith

Chairman, Department Committee on Graduate Theses
Table of Contents

TABLE OF CONTENTS ................................................................. 2
ABSTRACT ................................................................................. 5
ACKNOWLEDGEMENTS................................................................. 6
INTRODUCTION ........................................................................... 7
BACKGROUND AND RELATED WORK IN WSD.......................... 10
  METHODS OF CORPUS-BASED WSD ........................................ 11
  EVALUATING WSD ALGORITHMS ........................................... 16
WORD SENSE DISAMBIGUATION IN INFORMATION RETRIEVAL .... 18
THE PROBLEM ............................................................................ 22
PROPOSED SOLUTION ................................................................ 25
IMPLEMENTATION ...................................................................... 28
EVALUATION OF THE DISAMBIGUATION ALGORITHM ............. 37
  PERFORMANCE OF THE "MOST FREQUENT" HEURISTIC .......... 38
  PERFORMANCE OF OUR ALGORITHM ON SEMCOR ................. 39
DISCUSSION .............................................................................. 40
FUTURE WORK .......................................................................... 43
CONCLUSION ............................................................................ 44
REFERENCES ............................................................................ 45
List of Figures

FIGURE 1: OUTLINE OF SELECTOR EXTRACTION PROCESS ........................................... 32
FIGURE 2: INFORMATION FLOW .................................................................................. 36
List of Tables

TABLE 1: SOME CONTEXTS EXTRACTED FROM THE CORPUS IN DISAMBIGUATING THE SENTENCE ABOVE ................................................................. 30

TABLE 2: CONTENT WORDS AND CONTEXTS ................................................................. 31

TABLE 3: SENSES OF "MANNER" FROM WORDNET .................................................. 34

TABLE 4: SELECTORS THAT MATCH THE CONTEXT "CREATION OF X" .................. 40

TABLE 5: SENSES, HYPERNYMS AND HYpon .IS OF "HEART" ................................. 42
Word Sense Disambiguation Applied to Information Retrieval

by

Ozlem Uzuner

Submitted to the

Department of Electrical Engineering and Computer Science

May 26, 1998

In Partial Fulfillment of the Requirements for the Degrees of

Bachelor of Science in Computer Science and Engineering and

Master of Engineering in Electrical Engineering and Computer Science

ABSTRACT

We believe that word sense ambiguity is one of the main reasons for poor performance of Information Retrieval (IR) systems. For this reason we develop a word sense disambiguation (WSD) algorithm which will be used in disambiguating natural language queries to an IR system. This WSD algorithm uses comparison of WORDNET senses and context selectors of an ambiguous word to determine the correct sense of the word in a context.

Our experiments have shown that this algorithm performs slightly worse than the most-frequent heuristic when tested on the SEMCOR corpus, but there is reason to believe that relatively straightforward modifications to the algorithm can significantly improve its performance.

Thesis Supervisor: Boris Katz

Title: Principal Research Scientist
Acknowledgements

I am grateful to Boris Katz for providing the opportunity and the resources for working on this exciting project, and for his advice and support. I would also like to thank him and the Infolab Group for providing and being a part of a perfect work environment.

I’d also like to thank Deniz Yuret for his help throughout the term, Sue Felshin for proofreading this thesis and the Infolab Group for their valuable input.

Finally, I would like to thank Giorgos Zacharia for all his help, all my friends for their support and my family for always pointing out the light at the end of the tunnel when times were rough.
Introduction

Word-Sense ambiguity is rarely thought of as a problem in our daily life. Most of the time, even when we are presented with a text where an ambiguous word appears in more than one of its different senses, we can easily identify the correct sense of the word without getting confused about alternative senses. The disambiguation process, which helps people identify the correct sense of a word is not difficult for us. We can perform it easily and accurately. However, this disambiguation process is not so easy for computer applications. Given a text including an ambiguous word like “crane”, without performing some sort of disambiguation, it is impossible for a machine to know whether we are talking about the machine that lifts and moves heavy objects or the large long-necked wading bird of marshes and plains in many parts of the world. And, unfortunately, even when after disambiguating words, machines may not be able to resolve ambiguities. The task of word sense disambiguation is to make machines perform as well as people in identifying the senses of ambiguous words in a context.

Information Retrieval is one of the fields where word sense ambiguity is a problem. The rapid increase in the number of electronic documents available on the World Wide Web and the increased desire to obtain useful information from these documents have increased the need for the development of sophisticated Information Retrieval (IR) systems. Especially in research oriented domains the importance of precise and efficient IR systems is indisputable.

Traditionally, IR systems use keywords for indexing and retrieving documents. These systems make use of keywords and their stems in retrieving documents related to the query. These systems end up retrieving a lot of irrelevant information along with
some useful information that the query was intended to elicit. An example given in "Word Sense Disambiguation and Information Retrieval" [20] by Sanderson demonstrates this problem clearly. Sanderson gives an example where "a number of users tried to retrieve articles about the British Prime Minister using the query 'major'. This query caused many articles about 'John Major' to be retrieved. However, in addition many more articles were retrieved where 'major' was used as an adjective or as the name of a military rank".

One method used for improving recall in IR systems includes the expansion of the query with its synonyms and morphological variants. This method has been shown to improve recall at the expense of precision.

The example given by Sanders implies that a successful disambiguation of the word 'major' prior to IR would resolve the problem of retrieving irrelevant documents. However, there is evidence which contradicts this claim. While Schutze and Pedersen [21] describe a sense disambiguator that improves the precision of an IR system by 4%, Sanderson [20] presents results which show that the disambiguation process usually effects the performance of the IR system negatively. He proves that in order to be of any practical use and in order to improve the performance of an IR system, a disambiguation algorithm has to work with at least 90% accuracy.

One thing that attracts our attention about the previous work [22] [23] is that even the most extensive research in the application of word sense disambiguation to information retrieval has focused on retrieving disambiguated information from disambiguated documents. We believe that focusing on disambiguating the query rather than the
documents would be more practical, could eliminate the need for disambiguation of the documents, and could give different and hopefully better results.

The main problem we are trying to solve involves the expansion of an IR query with its synonyms. Therefore, given a natural language query, we use WSD in identifying the relevant synonyms in its context. We would like to see if expanding a query with its relevant synonyms only can improve the performance of an IR system. Since we are applying WSD to a very specific domain, we can afford not to look at the ambiguity in documents.

In this thesis, we describe our work in which we try to develop a word sense disambiguator which can be used to improve precision as well as recall of an IR system. With the help of this research, we hope to discover if disambiguating an IR query can be helpful in improving the performance of an IR system.

Our system is based on the following idea: A document relevant to our query might contain either the words in the query or their synonyms. This implies that we should be able to improve recall by considering the synonyms as a part of the IR query. However, if we include all possible synonyms of the query in our retrieval, precision will suffer. In order to increase both precision and recall, we need to use only the relevant synonyms in a context. We can identify these relevant synonyms with the help of a disambiguation algorithm.
Background and Related Work in WSD

Words in natural language are known to be highly ambiguous. This is especially true for the frequently occurring words of a language. For example, in the WORDNET dictionary, the average number of senses per noun for the most frequent 121 nouns in English is 7.8, but that of the most frequent 70 verbs is 12.0 [15]. This set of 191 words is estimated to account about 20 percent of all word occurrences in any English free text. Therefore, word sense ambiguity is a prevalent problem in NLP.

Research into automatic resolution of word senses has been going on for at least forty-five years, and there is a large literature describing a variety of different WSD-techniques. Earliest WSD methods used hand-coding of knowledge to disambiguate word senses. In these systems, each word to be disambiguated would need to be hand-tagged with the correct piece of information, e.g., part-of-speech, sense, etc., which would be useful in the disambiguation process. Therefore, it was difficult to come up with a comprehensive set of the necessary disambiguation knowledge and even more difficult to manually maintain and further expand the disambiguation knowledge to handle real world sentences.

In order to solve this problem, some researchers decided to use pre-coded information in the form of machine-readable dictionaries and thesauri [10] [13]. Others started to build their own dictionaries and thesauri with information concluded from statistics over large corpora. This approach is called the corpus-based approach [10]. In contrast to manually hand-coding the disambiguation knowledge into a system, the corpus-based approach uses machine-learning techniques to automatically acquire disambiguation knowledge, e.g., verb-object relations, from large corpora.
Methods of Corpus-Based WSD

Corpus-based WSD systems can broadly be classified into “supervised approach” and “unsupervised approach”. Most research efforts in unsupervised WSD rely on the use of knowledge contained in a machine-readable dictionary. A widely used resource is WORDNET [12], which is a public domain dictionary containing about 95,000 English word forms. Besides being an online, publicly available dictionary, WORDNET is also a large-scale taxonomic class hierarchy, where each English noun sense corresponds to a taxonomic class in the hierarchy. The is-a relationship in WORDNET’s taxonomic class hierarchy is an important source of knowledge exploited in unsupervised WSD algorithms.

Resnik [17] gives a good example of an unsupervised WSD algorithm. In his method, Resnik uses the verbs, adjectives or nouns which modify the word to be disambiguated. For example, in disambiguating the sense of coffee in the test sentence drinking coffee, Resnik’s algorithm uses the verb drinking to find other words which might be modified by it in a similar fashion to coffee. Words like milk, wine, tea are usually used with the verb drinking and the sense common to all these words is the beverage sense. Therefore, he concludes that he is looking for the beverage sense of the word coffee.

In the supervised approach, a WSD program learns the necessary disambiguation knowledge from a large sense-tagged corpus, in which word occurrences have been tagged manually with senses from some wide-coverage dictionary, such as the Longman’s Dictionary of Contemporary English (LDOCE) or WORDNET. After training on a sense-tagged corpus in which all occurrences of the word w have been tagged with
their correct sense, a WSD program is able to assign an appropriate sense to \( w \) appearing in a new sentence, based on the knowledge acquired during the learning phase. Common methods used in supervised WSD include:

1. Forming context vectors.

2. Using context vectors along with a semantic knowledge base and a frequency-based relevance measure.

3. Selecting a good feature, which captures an important source of knowledge critical in determining the sense of the word \( w \).

4. Collecting the surrounding words (unordered) which capture the broad topic of a text.

5. Collecting collocational information to form decision lists: a local collocation refers to a short sequence of words near \( w \), taking the word order into account. Such a sequence of words need not be an idiom to qualify as a local collocation. Collocations differ from surrounding words in that word order is taken into consideration.

6. Syntactic relations: Traditionally, selectional restrictions indicated by syntactic relations such as subject-verb, verb-object, and adjective-noun relations are considered an important source of information that can be used in disambiguation.

7. Parts of speech and morphological forms: Many WSD systems have used the part-of-speech and morphological variants of \( w \) as well as the part-of-speech of the neighboring words of \( w \) in disambiguation. The part-of-speech, for example, can help reduce the number of possible senses of an ambiguous word.

Some preliminary findings suggest that local collocation provides the most important source of disambiguation knowledge; however, the accuracy achieved by the combined
knowledge sources—part-of-speech, morphological forms, syntactic relations, etc.—exceeds that obtained by using any one of the knowledge sources alone [15].

In his paper “Word Space” [19], Schutze describes a corpus-based disambiguation method which builds a vector representation of word meanings for the purpose of identifying different meanings of a word. For this, Schutze forms four-gram co-occurrence matrices and context vectors for each target word by normalizing the sum of the co-occurrence matrices of the four-grams around the target word. He then plots these context vectors in his multi-dimensional space marked by the co-occurrence matrices. In this space, the context vectors which are distance-wise close to each other refer to the same sense of the target word. After hand-labeling each of these clusters with the correct meaning, on a task of pairwise-disambiguating 10 well-known ambiguous words, the system achieved an average accuracy of greater than 92%.

Resnik [17] provides an extension to Schutze’s “word space” by describing an automatic method of labeling the different clusters of contexts. Resnik’s method is based on the observation that: “when given a list of words, a human being will assign to an ambiguous word in the list, the meaning which will match the sense of the rest of the words in the list.” To perform this meaning assignment automatically, Resnik uses the IS-A hierarchy of WORDNET [12]. In this hierarchy, each word is assigned an informativeness level which is inversely proportional to the frequency with which the word occurs in English language. Once he calculates these informativeness levels, for each word in the hierarchy, Resnik picks the sense with the highest informativeness measure by assigning to each sense the sum of the informativeness measures of all the ancestors that support that particular sense. On a task of identifying the 23 different
meanings of the word 'sense', where the human beings achieved 67% accuracy, the machine achieved 60% accuracy.

Yarowsky [25][26][27] describes a decision list approach to solving the WSD problem. In order to implement his algorithm, Yarowsky makes the assumption that, for each ambiguous word, "there is one sense per discourse and one sense per collocation". This assumption means that in a certain text, an ambiguous word will appear in only one of its senses and similarly in a certain word configuration the ambiguous word will always mean the same thing. Based on these assumptions, Yarowsky collects "seed" sentences which provide the system with an example sentence for each of the meanings of an ambiguous word. By analyzing the collocations in these seed sentences, he builds decision lists which help him identify the meaning of an ambiguous word in a certain collocation. Decision lists are very similar to "rules" of a rule-based system and they encode information about collocations indicative of each sense of a word. For example, a decision list for the word "board" would have an entry in the form "+1 word to the right should be members" which would indicate the "committee" meaning of word "board". Building decision lists helps identify a most important informant for each sense of an ambiguous word. The most important informant is placed at the top of the decision list and is the first criterion to be checked. When this criterion is not satisfied, the correct sense of a target word can be identified by checking the rest of the rules in the decision list. When tested on a set of pairwise-ambiguous words, this algorithm achieved an accuracy of 95%.

Ng and Lee present the LEXAS [15] system, which uses an exemplar-based approach to WSD. To simplify the disambiguation problem, LEXAS assumes that its input
sentence has been pre-tagged with correct part-of-speech (POS), so that the possible
senses to consider for a content word $w$ are only those associated with the particular POS
of $w$ in the sentence. Then it starts its supervised exemplar-based learning where for each
example, it learns the word $w$, its morphological variant, the words that frequently occur
with $w$ in the same sentence, and the local collocations containing $w$. For disambiguating
$w$, the verb which takes $w$ as its object is also identified. LEXAS does the same kind of
analysis on test sentences and matches these features to the training examples. The sense
of word $w$ in the test example is the sense of $w$ in the closest matching training example.
When tested on a corpus in which 192,800 word occurrences had been manually tagged
with senses from WORDNET, LEXAS performed better than the default strategy of
picking the most frequent sense [15].

Lin [11] describes a system which uses local context for resolving sense ambiguity. In
Lin's approach, the local context of each word $w_m$ is a triple of \{relation to word $w_k$, word
$w_k$, position with respect to $w_k$\} and these contexts are extracted for all appearances of the
target word $w$. Lin then searches his local context database for words that appeared in an
identical local context as $w$. He calls these words the selectors of $w$. This allows him to
choose a meaning for $w$ that maximizes the similarity between $w$ and its selectors. Using
the "one sense per discourse" heuristic advocated by Gale, Church and Yarowsky [7], the
sense chosen is assigned to all occurrences of $w$ in the input text.
Evaluating WSD Algorithms

As pointed out by Ng and Zelle [16], the evaluation of empirical, corpus-based WSD has not been as rigorously pursued as other areas of corpus-based NLP, such as part-of-speech tagging and syntactic parsing. The lack of standard, large, and widely available test corpora discourages the empirical comparison of various WSD approaches.

Currently, several sense-tagged corpora are available. These include a corpus of 2,094 examples about 6 senses of the noun line; a corpus of 2,369 sentences about 6 senses of the noun interest; SEMCOR, a subset of the BROWN corpus with about 200,000 words in which all context words (nouns, verbs, adjectives, and adverbs) in a running text have manually been tagged with senses from WORDNET; and the DSO corpus [15], consisting of approximately 192,800 word occurrences of the most frequently occurring (and hence most ambiguous) 121 nouns and 70 verbs in English. The last three corpora are publicly available from New Mexico State University, Princeton University, and Linguistic Data Consortium (LDC), respectively.

A baseline called the "most frequent" heuristic has been proposed as a performance measure for the WSD algorithms [7]. This heuristic simply chooses the most frequent sense of the word w and assigns it as the sense of w in test sentences without considering the effect of context. A WSD algorithm must perform better than the most frequent heuristic to be of any significant value.

The success of a WSD algorithm depends on the corpus, the words being tested, the number of senses to a word, and the test corpus from which the words originate as well as the quality of the algorithm. In addition, verbs are typically harder to disambiguate than nouns, and disambiguation accuracy for words chosen from a test corpus composed of a
wide variety of genres and domains, such as the Brown corpus, is lower than the accuracy of words chosen, say, from business articles in the Wall Street Journal [15].

Supervised WSD algorithms have on average achieved an accuracy of about 73% to 76% on the line corpus; 87.4% on the interest corpus; 69% on the SEMCOR corpus; and 58.7% and 75.2% on two sets from the DSO corpus.

The accuracy for unsupervised methods is harder to assess because most are tested on different test sets with varying difficulty. Although supervised WSD algorithms have the drawback of requiring a sense-tagged corpus, they tend to give higher accuracy compared with unsupervised methods. For example, when tested on SEMCOR, Resnik’s [17] unsupervised algorithm achieved accuracy in the range of 35.3% to 44.3% for ambiguous nouns. The most frequent heuristic achieved accuracy of 58.2% for all ambiguous words in this corpus.

WSD is an essential part of many NLP applications. Research done by Schutze and Pedersen has shown that when tested on part of the TREC corpus, a standard information retrieval test-collection, WSD improves precision by about 4.3 percent – from 29.9 to 34.2 percent [21]. We believe that WSD can improve retrieval accuracy of IR systems.
Word Sense Disambiguation in Information Retrieval

The recent and rapid increase in the amount of information available on the Web has increased the need for sophisticated Information Retrieval (IR) systems. The success of IR systems is determined by precision and recall measured on their output, and an ideal IR system returns to its user only the documents and exactly the documents that the user is searching for. Most of the IR systems we know use keywords for indexing and document retrieval. Some others use both keywords and their synonyms.

The first project where a word sense disambiguation algorithm was used with an IR system was by Weiss [24]. Weiss tested his algorithm on five handpicked words in the Aviation Data Internet (ADI) collection. Resolving ambiguities before information retrieval, Weiss showed that he improved performance of his IR system by 1% for the five test words.

The results established by Krovetz and Croft [14] were more interesting and gave more information about the relation between WSD and IR. Krovetz and Croft used CACM and TIME test collections and performed retrieval for each of the standard queries in these collections. For each retrieval, they analyzed the match (or the mismatch) between the intended sense of the word, and the sense of the word in retrieved documents. They manually examined thousands of these query-document pairs. This thorough analysis revealed that the sense mismatches occurred more often when there was a smaller number of words in common between the query and the document. They observed that documents with many words in common with a query, which are ranked highly with regard to that query, tend to use the words in the same sense as the query. Therefore they concluded that WSD did not have a very important impact on IR, but that
disambiguation could be beneficial to IR when there were a few words in common between the query and the document.

Voorhees [22] presented a large-scale test of applying word sense disambiguation to an IR system. For this test, Voorhees built an automatic indexer, which could also be called a sense disambiguator, based on the is-a relations contained within \textsc{wordnet} thesaurus [12] and a set of nouns contained in a text. The main idea of this disambiguator was based on the relation between different nouns in a text. Voorhees claims that the nouns in a sentence should be indicative of each other's meanings. For example, the list of words \textit{base}, \textit{bat}, \textit{glove} and \textit{hit} indicate the \textit{sports equipment} meaning of the word "bat". In a case like this, the disambiguation of the word "bat" depends mainly on identification of the category of objects that "bat" belongs to. This category can be determined by adding up the votes of the words that appear in the context of word "bat". Making this part of her main idea, Voorhees used "hoods" to identify the category to which each of the words in a sentence belong. If we consider the hyponymy links of \textsc{wordnet} as the set of vertices and directed edges of a graph, then the hood of a word \( w \) contains only descendents of an ancestor of \( w \), and contains no synonym set that has a descendant that includes another instance of \( w \). Using the "hoods" and considering \textsc{wordnet} as a graph, Voorhees identified the part of \textsc{wordnet} which better matches the context of the query and identifies the meaning of the ambiguous word as the sense that appears in that part of \textsc{wordnet}.

Using this indexer/disambiguator, Voorhees produced a vector in which some of the terms represented word senses and some represented word stems. This indexer was then applied to the CACM, CISI, CRAN, MED and TIME collections. "Retrieval experiments
comparing the effectiveness of these sense-based vectors vs. stem-based vectors showed the stem-based vectors to be superior overall, although the sense-based vectors do improve the performance of some queries." From these experiments, Voorhees concludes that the overall degradation of performance is mostly due to the difficulty of disambiguating senses in short query statements.

Wallis [23] used a more complex disambiguator which could replace the words in a text collection with the text of their dictionary definitions. The purpose of this complex disambiguation was that, instead of providing one-word senses for a word providing a full text dictionary definition would allow synonymous words to be represented in a similar manner. Wallis hoped that this would increase the chance for the IR system to retrieve these documents together. However, the tests showed no significant improvement in IR performance after this disambiguation.

Arampatzis et al. [2], have shown that full natural language queries or noun phrase queries give better performance on IR systems compared to keywords. The noun phrases used in the Profile project of Arampatzis are represented in the form of \{fume, stink, air, pollution\}. This context representation allows different phrases with the same meaning to match to the same context definition. For example, a document with the noun phrase \textit{polluted air}” would match a query of the form \textit{air pollution by stinking fumes}”.

Krovetz and Croft [14] have conducted some experiments with morphology and part-of-speech, and they concluded that grouping morphological variants together significantly improves the IR performance, that more than half of the words in a dictionary that differ only in part-of-speech are related in meaning, e.g., the verb
definition of “change” and the noun definition of “change” have related meanings, and that it is crucial to assign credit to the component words of a phrase.
The Problem

Traditionally, the input to a WSD program consists of unrestricted, real-world English sentences. In the output, each word occurrence $w$ is tagged with its correct sense number (which appears in a previously agreed dictionary) according to the context. For our work, we use the sense definitions as given in WORDNET, which is comparable to a good desktop printed dictionary in its coverage and sense distinction [15]. Since WORDNET only provides sense definitions for content words (i.e., nouns, verbs, adjectives and adverbs), we are only concerned with disambiguating the senses of content words. Almost all previous work, as well, in WSD deals only with disambiguating content words, [15].

Most WSD algorithms focus simply on better disambiguation, rather than making use of a disambiguating algorithm. For this reason, previous work has focused on different learning methods and different ways of inferring the meaning of the words in a context. The applications of WSD to other unsolved problems like IR have not been investigated in as much depth and have so far revealed contradicting answers.

It is our belief that word sense ambiguity can be one of the causes of poor performance in IR systems. Polysemy (a single word form having more than one meaning) and synonymy (multiple words having the same meaning) both reduce the performance of IR systems. Polysemy reduces precision by causing false matches whereas synonymy reduces recall by causing true conceptual matches to be missed [22]. Therefore, especially in the systems which expand the query on the synonyms of the word before processing the query, the IR performance can be improved if the query can be perfectly disambiguated. This claim contradicts the information provided by Sanderson [20].
Sanderson's conclusion from his research, conducted before the web, on probabilistic IR systems is that "word sense ambiguity is only problematic to an IR system when it is retrieving from very short queries. In addition, if a word-sense disambiguator is to be of any use to an IR system, then it must be able to resolve word sense to a high degree of accuracy" [20].

Other research has indicated that WSD may or may not provide us with a solution to this problem [20]. For example, Voorhees [22] tried doing information retrieval on a disambiguated corpus and showed that the performance decreased rather than increasing. She also showed that trying to disambiguate the query in addition to the corpus made the results worse, especially in cases where the query was very short. Wallis [23] showed that running IR on a disambiguated corpus did not improve the performance of the IR system.

The main thing that strikes us about these two systems is the fact that researchers have tried to disambiguate the corpus as well as the query. Both of these systems try to improve the performance of IR by retrieving documents from a disambiguated corpus, rather than retrieving disambiguated queries from a regular corpus. We focus our research not on retrieving information from a disambiguated corpus but on retrieving disambiguated information from a regular corpus.

More specifically, we believe that if we can identify the correct synonyms of the content words in a natural language query, we can increase both precision and recall by expanding the query with the correct synonyms of these words.

Therefore, our main goal in this thesis is to come up with a WSD algorithm which can correctly identify the senses of content words in a specific context. In the next
chapters of this thesis, we are going describe our implementation of such a disambiguation algorithm and the results of our evaluations.
Proposed Solution

Most previous corpus-based WSD algorithms determine the meanings of polysemous words by exploiting their local contexts. Different systems have different definitions of local context. In Ng and Lee [15], the local context of a word consists of an ordered sequence of six surrounding part-of-speech tags, the word's morphological features, and a set of collocations which describe the most frequent patterns in which the word appears. Lin [11] defines local context in terms of syntactic dependencies between the word and other words in the same sentence. In our case, local context is an ordered list of words which appear within $n$ words of the ambiguous word. The words that appear in exactly the same local context as the targeted ambiguous word are called the selectors of that ambiguous word. For example, in the sentences below:

1. During lunch, I consumed three plates of food.
2. During lunch, I ate three plates of food.
3. During lunch, I consumed three cups of coke.

Sentences 1 and 2 share the context “During lunch, I [...] three plates of food”. Comparing these two sentences, we see that verbs consumed and ate have been used in exactly the same context. Since these two words also have synonym sets which might refer to each other, i.e., eat has a synonym set which includes the word consume and vice versa, in this specific context, consumed and ate have similar meanings and are selectors of each other. In sentences 1 and 3 on the other hand, the context is “During lunch, I consumed three [...]”. This context helps us identify “plates of food” and “cups of coke” as selectors of each other.
Most previous corpus-based WSD algorithms determine the meanings of polysemous words by exploiting their local contexts. The basic intuition that underlies these algorithms is that two occurrences of the same word should have identical meanings if they have similar local contexts.

In other words, in order to disambiguate a certain word, most previous corpus based WSD algorithms observe the previous usages of that word and learn classifiers for it. Each of these classifiers holds the information necessary for identifying one sense of the ambiguous word. There are several disadvantages to this approach. First of all, it is very difficult to learn good classifiers for each word since a word must be encountered thousands of times before a good classifier can be learned for it. In Yarowsky's experiment [27], an average of 3936 examples of each sense of the word were used to disambiguate two sentences. In Ng and Lee's experiments [15], 192,800 occurrences of 191 words were used as training examples. There are thousands of polysemous words. For example, there are 11,562 polysemous words in WordNet. In order for each polysemous word to appear thousands of times each in a corpus, the corpus must contain billions of words. The second major drawback is that since these algorithms learn to disambiguate a word from its previous usages, these algorithms cannot deal with the words for which classifiers have not been learned yet.

In order to avoid these drawbacks, we can use an algorithm which does not require learning classifiers for each polysemous word. Instead we can try to learn about the meaning of each polysemous word by looking at the context it is used in and by comparing it with other words that appear in the same exact context. This method still
allows us to use the information implied about the meaning of the word by its context; however, it saves us the trouble of having to learn classifiers for each word.

The main idea is that the context of a word \( w \) indicates the meaning of \( w \). Therefore, words used in the same context as \( w \) should have similar meanings to \( w \), or at least give us a good idea about which sense \( w \) is used in. In other words, two occurrences of a word and its synonym \textit{belong to the same sense} if they have similar local contexts. This means that we can use local contexts, and the synonyms of a word used in those contexts to identify which of its senses an ambiguous word is used in.

This approach does not require an ambiguous word to exist in the corpus since it does not need to learn the meaning of the word from its previous occurrences. Other advantages of this approach include the following:

- No specific classifier needs to be learned for each word. Instead we can use the same knowledge sources for all words. In our case the main knowledge source is the corpus where we search for words appearing in a context.

- In order to accomplish thorough training, most algorithms which learn classifiers use very large sense-tagged corpora. Our algorithm can identify the senses of the ambiguous words that best fit into a context without needing sense-tagged corpora.

- The frequency with which a certain ambiguous word appears does not affect the performance of the system. But the frequency with which the context appears in the corpus affects our chance of finding the best sense of the word in that context. So, in theory, this algorithm should be able to deal with words that are infrequent or do not even appear in the corpus.
Implementation

We base our solution to this problem on the synonym sets of each word available in WORDNET [12]. Using the data, index files and utility functions of WORDNET, we can extract information about the synonym sets as well as hypernyms and hyponyms of words. Using WORDNET simplifies our WSD problem to a certain degree, and disambiguation of a certain word reduces to identification of the correct synonym set of the word in a certain context.

We believe that we can identify the "correct", i.e., the most relevant, synonym set of a word in a context by matching the synonym sets of this word from WORDNET against a set of words which have been used in the same context as our query word \( w_m \). As shown by psychological experiments, people can resolve sense ambiguities by looking at a narrow window of words surrounding the ambiguous word [5]. This fact leads us to expect to identify the correct meaning, i.e., the best synonym set of a word \( w_m \), by looking at its context [28]. What is more, the content words, i.e., nouns, verb, adjectives, and adverbs, that appear in the same context are usually related to specific senses of each other [28]. Therefore, identifying the correct sense of one of the words in a context can help us identify the sense of other words in the same context. Alternatively, the use of certain content words in a context can help us identify the general sense of the context. Content words which contain information about each other are called informants of each other.

Our implementation of this algorithm takes natural language input made of words \( w_1 \), \( w_2 \), \( w_3 \), \( w_4 \), \( w_5 \), ..., \( w_n \). We parse this plain English input into words and process each word separately from others. For each word \( w_m \) in our query, we find its morphological root by
using the *morphword* and *morphstr* functions provided by *WORDNET*. Finding the morphological root of each word is a crucial step in our algorithm since in *WORDNET* words are only found in root form. In other words, *WORDNET* can provide us with information, e.g., hypernyms, hyponyms and synonym sets, about the word "compare" while it contains no such information about the word "comparing".

After finding the morphological root of each of the words in our query, we extract their synonym sets. In *WORDNET*, the synonym sets are arranged into different groups according to the parts of speech of the word they match. For example, the word "leave" has two groups of synonym sets: one group for its noun sense, e.g., leave of absence, and the second for its verb sense. In each of these groups, the synonym sets are ordered from the most frequently used to the least frequently used. We extract all synonym sets for all possible parts of speech of all words in a query. Optionally, we add, to the synonym sets, hypernyms and hyponyms of each of the synonym sets from the *WORDNET* data files.

Once the synonym sets are extracted, we start the process of disambiguation by calculating the scores of the individual synonym sets. The score of each synonym set depends on the matches that it has with the *selectors* of the word to be disambiguated.

*Selectors* of $w_m$ are words which can logically and grammatically be used in exactly the same context as $w_m$. The best way to obtain these selectors is to extract them from a regular text. If our theory holds, the *selectors* of the word $w_m$ in a particular context will best match the correct synonym set of $w_m$ for that context.

For the purpose of finding the selectors, we used a corpus of approximately 227 million words, consisting of articles taken from the Associate Press [28]. While extracting our selectors from this corpus, we did not use a fixed window size, i.e., the
number of words in our contexts was not fixed. Rather, the size of the context of each word depended completely on the location of the closest informant of the word in question.

For each word $w_m$ the words that appear between $w_m$ and the first content word to its right, including the content word, form the right context of $w_m$. Similarly, the words that appear between $w_m$ and the first content word to its left, including the content word, form the left context of $w_m$. The context which contains the most information about the sense of $w_m$ is the union of the left and right contexts of $w_m$ and is bounded by the first content words that appear on either side of $w_m$. Consider the following example taken from SEMCOR:

"... the city_executive_committee which had over-all charge of the election deserves the praise and thanks ..."

In disambiguating this sentence our algorithm finds the following contexts in the order presented.

<table>
<thead>
<tr>
<th>Table 1: Some contexts extracted from the corpus in disambiguating the sentence above.</th>
</tr>
</thead>
<tbody>
<tr>
<td>city executive committee which X</td>
</tr>
<tr>
<td>X over-all</td>
</tr>
<tr>
<td>had X</td>
</tr>
<tr>
<td>X charge</td>
</tr>
<tr>
<td>over-all X</td>
</tr>
<tr>
<td>X of the election</td>
</tr>
<tr>
<td>charge of the X</td>
</tr>
<tr>
<td>X deserves</td>
</tr>
<tr>
<td>election X</td>
</tr>
<tr>
<td>X the praise</td>
</tr>
<tr>
<td>deserves the X</td>
</tr>
<tr>
<td>X and thanks</td>
</tr>
<tr>
<td>praise and X</td>
</tr>
</tbody>
</table>
As you can see from the table above, in disambiguating the word “charge” as it appears in this context, we find its two contexts. These contexts are shown in the table below.

Table 2: Content words and contexts.

<table>
<thead>
<tr>
<th>Left</th>
<th>Right</th>
</tr>
</thead>
<tbody>
<tr>
<td>First content word to the left: overall</td>
<td>First content word to the right: election</td>
</tr>
<tr>
<td>Left Context: overall X</td>
<td>Right Context: X of the election</td>
</tr>
</tbody>
</table>

The left context of word "charge" is "over-all X". The right context of the same word is "X of the election". So, if we can identify the best synonym set of "charge" which fits in the place of both of these X's, then we will be able to identify the best synonym set of "charge" in this context.

Once we have extracted the synonym sets, hypernyms, and hyponyms of these synonym sets from WORDNET, and the selectors from our corpus, we match the selectors with the synonym sets. The synonym set which matches the highest number of selectors is considered to be the synonym set which describes the meaning of $w_m$ in that context best.

Similarity in our case is defined as the weighted sum of the words in a synonym set that matches the selectors. The weights of the words in synonym sets are determined by the frequency with which they appear in the given context in the Associated Press corpus. For example, if the weight of synonym $s_m$ is 13, this means that $s_m$ was found 13 times in that specific collocation in the corpus. If a specific context never appears in our corpus, then all synonym sets of the ambiguous word are assigned a weight of zero and we pick the most-frequent sense of the word as its correct sense in that context.
"...the city executive committee which had an over-all charge of the election deserves praise and thanks..."

Pick each word one by one

- Find left context (overall $X$)
- Find right context ($X$ of the election)

- Find selectors that match $X$
- Find selectors that match $X$

Collect all selectors. Keep track of their frequencies

Selectors are ready

Figure 1: Outline of selector extraction process.

In our definition, as in the case with Lin [11], similarity is additive with respect to commonality. So, if A and B have more than one common component, then the similarity of A to B is the sum of the weighted similarities computed over all of the components in the commonality. The similarity between two different objects can not be less than zero.
In the cases when all of the synonym sets of a word score zero when matched against the selectors, we pick the most-frequent sense of that word as its correct sense in that context.

When we first decided to implement this approach, our main concern was the diversity of the selectors that could appear in a context. Initially, we feared that the diversity of the words appearing in a context would make it difficult for us to use the context as an indicator of the correct synonym set. However, paying attention to the details of our algorithm shows that this is not a big issue in our case.

First of all, our algorithm collects all selectors of a word in a given context. Since we do not only collect the most frequent selectors, this reduces the risk of losing the information provided by infrequent, but necessary selectors.

Second, our algorithm tries to find only the best synonym set of a word in a certain context. This means that we are only interested in the synonym set with the highest score. So, even when 90% of our selectors are irrelevant to our target word, since irrelevant words do not match any of the words in the synonym sets of the target word, they will not have an adverse effect on our results. Even in the unluckiest and exceptional case that an irrelevant word might match an incorrect synonym set, the effect of such a match will be minimal, since the probability that relevant words will match the synonym sets is higher than the probability that irrelevant words will match the synonym sets. Therefore, we are convinced that the diversity of the selectors should not affect our results too much.

Now let's look at some examples which will explain the functioning of our algorithm. Our example sentence is: "The city_executive_committee which had an over-all charge
of the election deserves praise and thanks for the manner in which the election was conducted." Let's first try to disambiguate the word "manner".

We start by finding the synonym sets of this word in WORDNET. The synonym sets are shown in table 3.

Next we find the left context of the word "manner" in our sentence. A corpus search for the appropriate replacements of manner in the context "thanks for the X" gives us a list of words—presented below ordered from most frequent to the least—which include memory, memories, honour, warning, way, call, strong, courtesy, compliment, exquisite, remarkable, nation, great, abundance, manner, freedoms, opportunity as well as other words.

Table 3: Senses of "manner" from WORDNET.

<table>
<thead>
<tr>
<th>Sense</th>
<th>manner, mode, style, way, fashion.</th>
<th>manner, personal_manner.</th>
<th>Manner</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specializations:</td>
<td>property</td>
<td>Demeanor, demeanour, behavior, behaviour, conduct, deportment.</td>
<td>Kind, sort, form, variety.</td>
</tr>
</tbody>
</table>

Comparing the selectors with the synonym sets, we see that the first sense of the word "manner" has a score equal to the sum of the number of times way and manner appeared
in the corpus. The score of the other two synonym sets is equal to the number of times the word manner appeared in the corpus. So far we see that the first sense of the word “manner” looks more similar to the sense we want, since it has a higher score than the other two sets. Now we have to do a corpus lookup on the right context of "manner". The corpus lookup on "X in which the election" returns a list of only two words: manner, church.

Since the word "manner" appears in all of the synonym sets, we see that the left context provides the significant information and that the first sense, the sense which includes the word "way", gets a higher score than the alternative synonym sets, and it is correctly identified as the sense of "manner" in this context.
"...the city_executive_committee which had an over-all charge of the election deserves praise and thanks for the manner in which the elections were conducted..."

Find the morphological root of each word

Find synonym sets for each of the words that now appear in root form
  If the words has synonym sets...

No. Done.

Yes, so find left and right contexts, and the selectors.

Compare each synonym set with selectors and calculate scores

Return the part of speech and sense number of the synonym set with the highest score.
Evaluation of the Disambiguation Algorithm

Previous research [15] has shown that the most informative source of information for the purposes of WSD is the local collocation, followed by part-of-speech and morphological form. Using surrounding unordered words gives lower accuracy and verb-object relation is the least informative information source for the purposes of WSD.

Evaluation of WSD algorithms is a more or less subjective process. This is partly due to the lexical resources, like \textsc{WordNet} and LDOCE, which concentrate their efforts in completeness and thus make very subtle distinctions between word senses. This sometimes makes it very difficult for even human beings to distinguish between two senses of the same word. Also, many times two people might disagree on the best sense of a word that would fit into a context. Thus, in some cases, it is almost impossible for a WSD algorithm to distinguish between two senses of a word. Although we believe that it is unnecessary to make such subtle distinctions between senses, we stick to \textsc{WordNet} senses of the words in evaluating the performance of our system.

To our knowledge, there are not too many common data sets available for testing WSD algorithms. One of the most commonly used data sets we know about is the \textsc{SemCor} corpus which is a subset of the \textsc{Brown} corpus. \textsc{SemCor} has a total of 186 files where each word in a sentence has been tagged with its correct part-of-speech and sense number taken from \textsc{WordNet}. Each of these files contains on average 800 sentences. For our tests, we randomly picked 60 of these files.
Performance of the "most frequent" heuristic

The "most frequent heuristic" has been accepted as the baseline for measuring performance of WSD algorithms. As we mentioned before, in order to be of any value, a WSD algorithm should perform at least as well as the "most frequent" heuristic. We implemented the most frequent method by making use of WORDNET. WORDNET orders the synonyms of each word from the most to the least frequent, and it always starts with the noun senses of the words. Therefore, the performance of the "most frequent" heuristic could easily be evaluated by assigning each ambiguous content word the sense of the first synonym set that appears in its noun group.

As we mentioned before, we tested the most frequent heuristic on 63 of the sense-tagged files present in SEMCOR. We observed that out of 40616 words that appeared in our corpus, only 29940 were content words. Of all these sense-tagged content words, 17311 appeared in their most frequent sense, i.e., sense 1 of the first part-of-speech that they appear in when the parts of speech are checked in the order: noun, verb, adjective, adverb. Therefore, the accuracy of the "most frequent" heuristic on this corpus was approximately 57.8%.


Lin evaluated his system on a subset of the SEMCOR corpus. Given the subtle differences between some senses of words in WORDNET, Lin considered the disambiguation to be accurate if the selected sense is close enough to the sense given in WORDNET. In his evaluation, two senses are considered to be close enough if $\text{sim} (s_{\text{answer}}, s_{\text{key}}) = 0.27$ which was determined to be the average similarity between 50,000 randomly generated pairs $(w, w')$. 
In order to find his selectors, Lin parsed 25 million word Wall Street Journal corpus in 126 hours. In order to increase the chance of reasonable disambiguation, he tested his algorithm on the "press reportage" part of SEMCOR which consists of 7 files of about 2000 words. His algorithm performed slightly worse than the most frequent algorithm when the similar, or "close enough", senses of a word were not considered the same.

Next, using the similarity constraint mentioned above, Lin combined the close senses of words and considered them as one sense. This reduced the number of senses of each word and made the disambiguation process a little easier. After combining senses that were similar enough, Lin reported that the performance of this system was better than the most frequent algorithm.

**Performance of our algorithm on SemCor**

We tested two slight variations of our algorithm on 60 files of the SEMCOR corpus. In the first test, we expanded our synonym sets with their hypernyms and hyponyms. We matched these expanded synonym sets with the selectors. Using this method, our algorithm achieved an accuracy of 57.1%. Next, we tested our algorithm using only the synonym sets. This time, too, the algorithm achieved an accuracy of 57.1%.

These experiments showed that our algorithm achieved a slightly lower accuracy than both the "most frequent" heuristic and Lin's algorithm. But we should note that since we do not know how Lin measured the performance of the most frequent algorithm, our comparison with his performance is only an approximation.
Discussion

One major weakness of our system is the vague contexts where many irrelevant words could fit perfectly. Since we are not defining a certain context length, the context length depends completely on the distance of the informants of $w_m$ to $w_m$. We suspect that this might cause us some problems in disambiguation since the closest informant of $w_m$ can be right next to or very close to $w_m$, producing a very short and generic context. As an example, consider the disambiguation of the word "universe" in the context "creation of universe". The left context of "universe" here is of the form "creation of $X"$, which is a very generic context. What is worse, this closest informant could appear as the closest informant of other words which can match a completely incorrect synonym set of $w_m$. In the context "creation of $X"$, $X$ can be replaced by almost any noun. The most frequent selectors that match this context are shown in the table below:

Table 4: Some of the selectors that match the context "creation of $X"

|---------|--------|---------|--------|---------|

On the other hand, the senses of "universe" as extracted from WORDNET are given below:

1. **Universe**, existence, nature, creation, world, cosmos, macrocosm

2. **Universe**, cosmos

3. Population, universe

4. **Universe**, universe of discourse

As you can clearly see from Table 4, the selectors of "universe" in this context are very uninformative about the sense "universe". Therefore, the selectors fail to indicate the correct sense.
Since we are not trying to solve the general WSD problem, in a case like this, we simply do not expand our query, and thus avoid retrieving irrelevant documents which would be returned if we blindly expanded the query with all the synonyms. Note also that, sense 1 is the correct sense here. And, if our corpus were not news oriented, there would probably appear a phrase like "creation of world" which would vote for sense 1, indicating it as the correct sense.

Generic contexts as in the example above provide us with very little (if any) information about the sense of the ambiguous word. In our algorithm, we consider these contexts as informative as the contexts which clearly indicate the meaning of the ambiguous word. These vague contexts lower the performance of our algorithm. We think that this problem can be solved by either making these vague contexts more specific, or by eliminating them from the test corpus altogether.

The other weakness of our system comes from the fact that there are cases where words used in identical contexts will not have similar meanings [11]. As an example, consider this sentence taken from Lin [11]:

"... the condition in which the heart beats between 150 and 200 beats a minute..."

A simple corpus search reveals the following list of words which appear most frequently in the same context, i.e., the context "X beats", as the word "heart" above are: "heart, police, soldiers, allegedly, sun, hearts, brutally, dead, men, rain, never, fatally, troops, forces, severely, pulse, drums, foolish, man, officers, first..."

As you can also see from the synonyms shown on table 4, in this case, again our selectors fail to give us any hint about the body-part sense of "heart". This can misled our algorithm into picking a wrong sense in the given context.
Table 5: Senses, hypernyms and hyponyms of “heart”

<table>
<thead>
<tr>
<th>Senses</th>
<th>Generalizations</th>
<th>Specializations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Heart, bosom</td>
<td>Intuition, hunch, suspicion</td>
<td>athlete's heart</td>
</tr>
<tr>
<td>Heart, pump, ticker</td>
<td>internal_organ</td>
<td>biauriculate_heart</td>
</tr>
<tr>
<td>Heart, mettle, nerve, spunk</td>
<td>Courage, courageousness, bravery</td>
<td></td>
</tr>
<tr>
<td>Center, centre, middle, heart, eye</td>
<td>area country</td>
<td>city_center, central_city, financial_center, hub, civic_center, municipal_center, down_town, inner_city, medical_center, midfield, see, midstream</td>
</tr>
<tr>
<td>Kernel, substance, core, center, essence, gist, heart, inwardness, marrow, meat, hub, pith, sum, nitty-gritty</td>
<td>Content, cognitive_content, mental_object</td>
<td>Hypostasis, quintessence, stuff</td>
</tr>
<tr>
<td>Heart, spirit</td>
<td>Disposition, temperament</td>
<td></td>
</tr>
<tr>
<td>Heart</td>
<td>plane_figure, two-dimensional_figure,</td>
<td></td>
</tr>
<tr>
<td>Heart</td>
<td>variety_meat, organs</td>
<td></td>
</tr>
<tr>
<td>Affection, affectionateness, fondness, tenderness, heart, warmheartedness</td>
<td>feeling</td>
<td>Attachment, fond_regard, protectiveness, regard, respect, soft_spot</td>
</tr>
<tr>
<td>Heart</td>
<td>playing_card</td>
<td></td>
</tr>
</tbody>
</table>
Future Work

The first possible modification to our algorithm involves the selectors. Right now, our algorithm compares the synonym sets with all the selectors of the ambiguous word. If we could identify the correct part-of-speech of the ambiguous word in the context, and match the synonym sets in that part-of-speech only with the selectors that are also in the same part-of-speech, then we might eliminate some unnecessary and possibly confusing information, and improve our performance.

As a second possible modification, if we can redefine our definition of contexts in a way that allows us to get rid of vague contexts, then we might increase our performance.

Once the performance of our algorithm is improved, we plan to test our algorithm with IR systems.
Conclusion

In this thesis, we have presented a WSD algorithm which can resolve sense ambiguities with an accuracy of 57.1%. This performance is slightly worse than the performance of the most frequent heuristic.

Compared to supervised WSD algorithms which use learning in disambiguation, our algorithm is completely unsupervised. Instead of learning the meanings of words from their past usages, it uses the context of the words in getting information about their meanings. This allows it to disambiguate even words which never appear in its our corpus.
References


