Automatic Extraction of Examples for Word Sense Disambiguation

Master’s Thesis in Computational Linguistics

Supervisors: Prof. Dr. Walt Detmar Meurers
Dr. Dale Gerdemann

External Advisor: Prof. Dr. Sandra Kübler

Author: Desislava Zhekova

Submission Date: March 27, 2009
Erklärung


Desislava Zheková
Abstract

In the following thesis we present a memory-based word sense disambiguation system, which makes use of automatic feature selection and minimal parameter optimization. We show that the system performs competitive to other state-of-art systems and use it further for evaluation of automatically acquired data for word sense disambiguation.

The goal of the thesis is to demonstrate that automatically extracted examples for word sense disambiguation can help increase the performance of supervised approaches. We conducted several experiments and discussed their results in order to illustrate the advantages and disadvantages of the automatically acquired data.
Acknowledgements

This thesis would not have been possible unless Professor Dr. Walt Detmar Meurers provided me the opportunity to complete my work at the Department of Linguistics at the University of Tübingen. The valuable guidance and professional environment in the department ensured also by my second supervisor Dr. Dale Gerdemann contributed immensely to my final success.

I would like to express as well my extreme gratitude to my advisor Professor Dr. Sandra Kübler for the constant and exceptionally constructive remarks, which were often the key to my progress.

I am also grateful to all those who provided assistance in numerous ways: David Münzing and Julian Münzing for the manual sense-annotation of a subset of my final data collection - a very laborious endeavor; Franziska Gruber, Ramon Ziai, Dominikus Wetzel and my loving family and friends for their endless support, encouragement and advice.
Contents

1 Introduction 10

2 Basic Approaches to Word Sense Disambiguation 12
   2.1 Knowledge-Based ................................................................. 12
      2.1.1 The Lesk Algorithm ......................................................... 13
      2.1.2 Alternative Methods ....................................................... 14
   2.2 Unsupervised Corpus-Based ..................................................... 15
      2.2.1 Distributional Methods ..................................................... 16
      2.2.2 Translational Equivalence Methods ..................................... 16
   2.3 Supervised Corpus-Based ....................................................... 18
      2.3.1 Sense Inventories ............................................................. 20
      2.3.2 Source Corpora ............................................................... 21
      2.3.3 Data Preprocessing ........................................................... 23
      2.3.4 Feature Vectors ............................................................... 23
      2.3.5 Supervised WSD Algorithms .............................................. 25
   2.4 Semi-Supervised Corpus-Based ................................................. 30

3 Comparability for WSD Systems 32
   3.1 Differences between WSD Systems .......................................... 32
   3.2 Most Frequently Used Baselines .............................................. 33
      3.2.1 The Lesk Algorithm ........................................................ 33
      3.2.2 Most Frequent Sense ....................................................... 34
      3.2.3 Random Choice .............................................................. 34

4 Evaluation of WSD Systems 35
   4.1 Fundamentals in Evaluation of WSD Systems .............................. 35
   4.2 International Evaluation Exercise Senseval .................................. 37
CONTENTS

4.3 Senseval-1 ......................................................... 37
4.4 Senseval-2 ......................................................... 38
4.5 Senseval-3 ......................................................... 39
  4.5.1 The All-Words Task ........................................ 39
  4.5.2 The Lexical Sample Task .................................... 40
  4.5.3 Other Tasks .................................................. 42
4.6 Semeval-1 ......................................................... 42
4.7 Summary ......................................................... 43

5 TiMBL: Tilburg Memory-Based Learner 44
  5.1 Overview ....................................................... 44
  5.2 Application .................................................... 45

6 Automatic Extraction of Examples for WSD 47
  6.1 Overview of the System ........................................ 47
  6.2 Data Collection ................................................ 48
    6.2.1 Sense Inventory ........................................... 48
    6.2.2 Source Corpora ............................................ 49
    6.2.3 Automatic Annotation ...................................... 53
  6.3 Data Preprocessing ........................................... 55
    6.3.1 Basic Preprocessing ....................................... 55
    6.3.2 Part-of-Speech Tagging .................................... 56
  6.4 Training and Test Sets ........................................ 57
    6.4.1 Feature Vectors ........................................... 57
    6.4.2 Training Set ............................................... 58
    6.4.3 Test Set ................................................... 58
  6.5 Algorithm Selection ........................................... 59
  6.6 Parameter Optimizations ...................................... 60
    6.6.1 General Parameter Optimizations ......................... 60
    6.6.2 Automatic Feature Selection ............................... 60
  6.7 Scoring ........................................................ 62
  6.8 Experimental Results ......................................... 62
    6.8.1 Supervised WSD ............................................ 63
    6.8.2 Unsupervised WSD .......................................... 64
    6.8.3 Semi-supervised WSD ....................................... 66
6.8.4 Discussion ................................................................. 69
6.9 Evaluation ........................................................................ 72

7 Conclusion, Future and Related Work .............................. 74
7.1 Related Work ....................................................................... 74
7.2 Future Work ......................................................................... 76
7.3 Conclusion ........................................................................... 76

References ................................................................................ 77

A List of Abbreviations ............................................................. 89

B Pool of Features ..................................................................... 90

C Training and Test Set (Senseval-3 English Lexical Sample Task) ................................................... 92

D Graphs .................................................................................. 93

E Tables ..................................................................................... 96
**List of Tables**

2.1 Basic approaches to WSD as in (Agirre and Edmonds, 2007). .......................... 13
2.2 Performance and short description for the unsupervised systems participating in
the SENSEVAL-3 English lexical sample task. Precision (P) and recall (R) (see
Section 4.1) figures are provided for both fine-grained and coarse-grained scoring
(Mihalcea et al., 2004a). ......................................................................................... 17
2.3 Performance and short description of the supervised systems participating in the
SENSEVAL-3 English lexical sample Word Sense Disambiguation task. Precision
(P) and recall (R) (see Section 4.1) figures are provided for both fine-grained and
coarse-grained scoring (Mihalcea et al., 2004a). ..................................................... 19
2.4 Feature vectors for the sentences in (2). ................................................................. 24
2.5 Supervised word sense disambiguation algorithms. .............................................. 26
4.1 Summary of the Senseval-1 evaluation task. ......................................................... 38
4.2 Summary of the Senseval-2 evaluation task. ......................................................... 39
4.3 Summary of the sense inventory of the words present in the English lexical sample
   task (Mihalcea et al., 2004a). .............................................................................. 40
4.4 Summary of the Senseval-3 Lexical Sample Task. ................................................ 42
6.1 The collection and size of lexicon examples used in our system. ....................... 51
6.2 The collection and size of corpora examples used in our system. ....................... 52
6.3 Features included in the feature vectors of our system and their corresponding
   values from our toy example. ............................................................................... 58
6.4 The total collection of examples used in our system. ......................................... 59
6.5 System performance on manually annotated data without optimization of the $k$
   parameter. ........................................................................................................ 63
6.6 System performance on manually annotated data with optimization of the $k$
   parameter. ........................................................................................................ 63
6.7 Comparison with the three best supervised systems in the Senseval-3 lexical sam-
   ple task (Mihalcea et al., 2004a). ...................................................................... 64
<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.8</td>
<td>System performance on automatically annotated data without optimization of the $k$ parameter.</td>
<td>65</td>
</tr>
<tr>
<td>6.9</td>
<td>System performance on automatically annotated data with optimization of the $k$ parameter.</td>
<td>65</td>
</tr>
<tr>
<td>6.10</td>
<td>Comparison with the four best unsupervised systems in the Senseval-3 lexical sample task (Mihalcea et al., 2004a).</td>
<td>65</td>
</tr>
<tr>
<td>6.11</td>
<td>System performance on automatically annotated data without optimization of the $k$ parameter and filtered for preservation of the distribution of the senses as in the original manually annotated training set.</td>
<td>67</td>
</tr>
<tr>
<td>6.12</td>
<td>System performance on automatically annotated data with optimization of the $k$ parameter and filtered for preservation of the distribution of the senses as in the original manually annotated training set.</td>
<td>68</td>
</tr>
<tr>
<td>6.13</td>
<td>System performance on automatically annotated data without optimization of the $k$ parameter and distance and distribution filtering mixed with the manually annotated training set.</td>
<td>69</td>
</tr>
<tr>
<td>6.14</td>
<td>System performance on automatically annotated data with optimization of the $k$ parameter and distance and distribution filtering mixed with the manually annotated training set.</td>
<td>69</td>
</tr>
<tr>
<td>6.15</td>
<td>Summary of the results for all experiments.</td>
<td>72</td>
</tr>
<tr>
<td>6.16</td>
<td>Comparison of our system’s scores with the upper and lower bound for the experiment and also with the best performing system in the Senseval-3 English lexical sample task.</td>
<td>73</td>
</tr>
<tr>
<td>7.1</td>
<td>The performance of our system on the Senseval-3 Lexical Sample Task data.</td>
<td>97</td>
</tr>
<tr>
<td>7.2</td>
<td>The performance of our system trained on the automatically gathered examples and tested on the provided by Senseval-3 Lexical Sample Task test set.</td>
<td>98</td>
</tr>
<tr>
<td>7.3</td>
<td>System performance on automatically annotated data with optimization of the $k$ parameter and filtered for preservation of the distribution of the senses as in the original manually annotated training set.</td>
<td>99</td>
</tr>
<tr>
<td>7.4</td>
<td>System performance on automatically annotated data with optimization of the $k$ parameter and distance filtering mixed with the manually annotated training set.</td>
<td>100</td>
</tr>
</tbody>
</table>
List of Figures

2.1 Supervised machine learning. .................................................. 20
2.2 General architecture of an memory-based learning system (Daelemans et al., 2007). 29
4.1 Screenshot from Open Mind Word Expert (Chklovski and Mihalcea, 2002). .... 41
6.1 Our semi-supervised WSD system. ....................................... 48
6.2 A screenshot of the result from the online interface of the Cambridge Advanced Learner’s Dictionary when searched for the word activate. ........................... 51
6.3 A screenshot of the result from the online interface of WordNet when searched for the word activate. ............................................. 54
7.1 Accuracy change of the adjective hot. ................................. 93
7.2 Accuracy change of the noun arm. .......................... 94
7.3 Accuracy change of the verb appear. ................................. 95
Chapter 1

Introduction

Ambiguity is one of the characteristics in all human languages and at the same time it is a problem to be solved in the studies of Computational Linguistics. It is represented by the uncertainty of meaning (if something can be understood in two or more senses) and thus requires a deeper linguistic processing of the natural language.

Nowadays it is easy to answer the question "Who understands language better - computers or human beings?". One of the reasons why computers still cannot compete with the human cognition is exactly the fact that ambiguity prevails in any natural language text. Thus, for the last few decades the problem has started to gain interest in the CL society and as Ide and Véronis (1998b) mention, proved to be a key to the solutions for other important areas of Natural Language Processing - Information Retrieval, Machine Translation, Information Extraction, Parsing, Text Mining, Semantic Interpretation, Lexicography, Content and Thematic Analysis, Grammatical Analysis, Speech Processing, Knowledge Acquisition and many others.

Ambiguity can be present in many ways (e.g structural - when it appears in a sentence or a clause, lexical - when it appears in respect to just a single word). Lexical ambiguity is the problem that Word Sense Disambiguation (WSD) is concerned with and has already tackled to a great extend. Human beings resolve lexical ambiguity by looking in the dictionary, while WSD tries to automate this process. Its task is to automatically assign the correct sense of an ambiguous word dependent on the context in which it can be found. However, by now there has not been found a perfect solution to the problem.

Ide and Véronis (1998b) describe many of the still open questions in WSD - How important is the context of the word that is disambiguated?, Which is the optimal choice of possible senses for the target word?, How to compare all different systems and their results?, How much data is needed in order good results to be achieved?, How to provide data for data-driven approaches? Those and many other problems have been considered an issue of great importance from many computational linguists, which is easily proven by the rapidly increasing number of papers in the ACL Anthology ¹ where the term word sense disambiguation is mentioned - approximately

¹http://www.aclweb.org/anthology-new/
CHAPTER 1. INTRODUCTION

850 times.

The advances in different aspects of WSD have resulted in a collection of expert views in the field - (Agirre and Edmonds, 2007). The book reviews basic questions like the nature of word senses - (Kilgarriff, 2007), (Ide and Wilks, 2007); discusses different WSD methods - (Mihalcea, 2007), (Pedersen, 2007), (Màrquez et al., 2007); examines evaluation techniques of automatic word sense disambiguation (Palmer et al., 2007), knowledge sources for WSD - (Agirre and Stevenson, 2007), domain-specific WSD - (Buitelaar et al., 2007) and as well discusses topics like automatic acquisition of lexical information (Gonzalo and Verdejo, 2007) and the application of WSD in Natural Language Processing (Resnik, 2007).

The predicament that we aim to discuss in the following thesis is a yet open question in WSD. It is the quantity and quality of the automatically extracted data for the process of word sense disambiguation - we will approach questions like how much and what kind of data can be acquired without human labor. In our work we present a WSD system and its application to various types of data, which outlines an environment and basis for evaluation of the automatic acquisition of examples for word sense disambiguation. We will not pursue an exhaustive empirical analysis of the latter but rather discuss the most relevant for our approach advantages and disadvantages of its employment.

Since we do not concentrate on a particular WSD approach, but rather compare the results from several such in order to examine them in contrast we consider it extremely important first to examine the fundamentals of word sense disambiguation and all the basic approaches to it (Chapter 2). Chapter 3 and Chapter 4 are respectively concentrated on the ways to compare and evaluate WSD systems in general depending on the used by the system approach. Before reviewing our work, however, in Chapter 5 we will shortly introduce the software which we employed. The following chapter (Chapter 6) will discuss the structure of the suggested by us system and Chapter 7 will give a brief overview of the research that has already been done in that area and the work which will be attempted further on together with our concluding remarks.
Chapter 2

Basic Approaches to Word Sense Disambiguation

Word sense disambiguation is the process of identification of the correct sense of a word in the context in which it is used. WSD has mainly one aim and it can be described in many analogous ways - to find out if two different words or phrases refer to the same entity, to be able to replace one with the other at the right place, to find out a sense that is common for both etc. In the process, however, there is an innumerable amount of unanswered questions and unsolved problems. One such question that turns out to be considerably essential for finding the right sense is the adequacy of the predefined set of senses a word can have (also called sense inventory - see Section 2.3.1). *Are there too many distinctions between the senses? Are there too few?* All such problems thus lead to many distinct approaches in the attempt to find the correct solutions. In the following chapter we will give a short overview of the fundamental divisions in those attempts. In order to explain what kind of WSD technique we use in our work it will be very helpful to first have a look at what kind of approaches to the problem exist according to the current state of art. A very clear overview on the basic approaches in WSD has already been given by Agirre and Edmonds (2007). The authors differ between knowledge-based (dictionary-based) and corpus-based (further divided as unsupervised, supervised and semi-supervised) approaches. In addition to that they discuss combinations of the different approaches which finally results in a variety which they summarize in a very precise manner as in Table 2.1 on page 13.

2.1 Knowledge-Based

Knowledge-based Word Sense Disambiguation is build upon knowledge acquired from sources other than corpora (e.g. lexical knowledge bases, dictionaries, thesauri).
CHAPTER 2. BASIC APPROACHES TO WORD SENSE DISAMBIGUATION

<table>
<thead>
<tr>
<th>Approach</th>
<th>Technique</th>
</tr>
</thead>
<tbody>
<tr>
<td>Knowledge-based</td>
<td>Hand-crafted disambiguation rules</td>
</tr>
<tr>
<td></td>
<td>Selectional restrictions (or preferences), used to filter out inconsistent</td>
</tr>
<tr>
<td></td>
<td>senses</td>
</tr>
<tr>
<td></td>
<td>Comparing dictionary definitions to the context</td>
</tr>
<tr>
<td></td>
<td>(Lesk’s method)</td>
</tr>
<tr>
<td></td>
<td>The sense most similar to its context, using semantic similarity measures</td>
</tr>
<tr>
<td></td>
<td>&quot;One-sense-per-discourse&quot; and other heuristics</td>
</tr>
<tr>
<td>Unsupervised corpus-based</td>
<td>Unsupervised methods that cluster word occurrences</td>
</tr>
<tr>
<td></td>
<td>or contexts, thus inducing senses</td>
</tr>
<tr>
<td></td>
<td>Using an aligned parallel corpus to infer cross-language sense disambiguation</td>
</tr>
<tr>
<td>Supervised corpus-based</td>
<td>Supervised machine learning, trained on a manually-tagged corpus</td>
</tr>
<tr>
<td></td>
<td>Bootstrapping from seed data (semi-supervised)</td>
</tr>
<tr>
<td>Combinatory Approaches</td>
<td>Unsupervised clustering techniques combined with knowledge base similarities</td>
</tr>
<tr>
<td></td>
<td>Using knowledge bases to search for examples for training in supervised WSD</td>
</tr>
<tr>
<td></td>
<td>Using an aligned parallel corpus, combined with knowledge-based methods</td>
</tr>
<tr>
<td></td>
<td>Using domain knowledge and subject codes</td>
</tr>
</tbody>
</table>

Table 2.1: Basic approaches to WSD as in (Agirre and Edmonds, 2007).

2.1.1 The Lesk Algorithm

One highly influential dictionary-based method which provided to a great extent the basis for most of the research in the area is the Lesk method presented in (Lesk, 1986). The algorithm is based on the assumption that words with similar surroundings will usually share a common topic. In other words the contextual overlap (in this approach - among dictionary definitions) is used as a measure to pick up the most likely sense for a given word. Lesk’s method was as well one of the first to work on any word in any text. The reason why this algorithm is classified as knowledge-based is because of the fact that it acquires knowledge only from a set of dictionary entries (a separate entry is needed for each sense of every word) concentrating on the immediate context of the target word. Here, with immediate context we mean the words that closely surround the target word. As an example we can look at the "evergreen" case of disambiguation of *pine cone* that Lesk (1986) first suggested as shown in (1) on page 14.
CHAPTER 2. BASIC APPROACHES TO WORD SENSE DISAMBIGUATION

(1)

**pine** 1 * seven kinds of evergreen tree with needle-shaped leaves
2 pine
3 waste away through sorrow or illness
4 pine for something; pine to do something

**cone** 1 solid body which narrows to a point
2 something of this shape whether solid or hollow
3 * fruit of certain evergreen trees (fir, pine)

As just mentioned, each of the separate senses has its own entry and is compared to the entries for the neighboring words of the target word. The aim of the comparison is to find overlapping words in the targeted entries. Thus, each of the entries of *pine* is compared to each of the entries of *cone*. As a result the largest overlap appears to be between the first sense of *pine* and the third sense of *cone*, both marked with * in example (1), hence those are the senses selected from the Lesk algorithm for both words. In such a way the latter algorithm disambiguates both words *pine* and *cone* if they appear together in the same context. In cases where each of those words is seen in different surroundings, the process of comparison of the dictionary entries for those surrounding words and the word *pine* or respectively *cone* will be pursued.

2.1.2 Alternative Methods

As Mihalcea (2007) notes, another type of knowledge-based method uses semantic similarity measures that are calculated from already existing semantic networks (e.g. WordNet\(^1\), Lexipedia\(^2\) etc.). The nature of this approach relies mostly on the fact that the smallest semantic distance between two words can be used as a trustworthy metric to show how closely related those words are. There are two different ways in the attempt to acquire the semantic similarity - local and global. The only difference between those two approaches is that in order to build threads of meaning the method either looks only in a very small window size of the context of the target word and discards the rest (local) or the entire text/document is considered (global).

Selectional preferences can be used both in knowledge-based as well as in corpus-based approaches for word sense disambiguation. This method tries to account for the fact that linguistic elements favor arguments of a specific semantic class, e.g. a verb like 'drink' selects as object drinkable things while a verb like 'eat' prefers as object edible things, and both prefer as subject animate entities, as for example: "He ate some chocolate.". Selectional preferences however could get a lot more complex than it might seem - consider for example sentences as: "The acid ate the

\(^1\)http://wordnet.princeton.edu/

\(^2\)http://www.lexipedia.com/
This truck eats a lot of fuel.” or “We ate our savings.”, etc. Normally, however, the biggest problem with selectional preferences is their circular relation with WSD. To be able to figure out the correct semantic constraint one needs to have an idea about the senses of the target words and at the same time WSD improves considerably if a big enough set of selectional relations is available.

Apart from the already discussed possibilities for knowledge-based methods there are also some heuristic approaches - most frequent sense (MFS) (see Section 3.2.2) for example. This method is set around the idea that in each case the sense of the target word that has the highest frequency amongst all of its senses is chosen regardless of the context in which this word is considered in. Unlike the other knowledge-based methods the heuristic methods are relatively easy to implement and fast to use with data of a bigger size. It has often been noted (e.g. Mihalcea (2007)) that the Lesk algorithm as well as the selectional preferences algorithm can become excessively computationally-exhaustive if more than few words are in the process of disambiguation. Moreover, the MFS heuristic is as well usually used as a baseline for most of the evaluation exercises for WSD systems (see Chapter 4).

One of the biggest advantages of knowledge-based methods in respect to corpus-based methods is the fact that despite their poor performance in terms of accuracy (percentage of labeled words correctly disambiguated), they can be applied to unrestricted (not domain specific) texts and unlimited amount of target words regardless the existence of already manually annotated sense-tagged corpora.

### 2.2 Unsupervised Corpus-Based

The nature of unsupervised WSD is considerably related to the problem of density estimation in statistics. In the process of disambiguation the aim of an unsupervised method is to discover the different patterns and structures in a predefined data source that has not been manually annotated beforehand. Exactly the possibility to work with data that does not need the extremely expensive human effort for manual annotation makes this approach so appealing. Thus there is already quite much going on in researching those paths: (Schütze, 1998), (Ramakrishnan et al., 2004), (Niu et al., 2004a), (Preiss, 2004), (Litkowski, 2004a), (Buscaldi et al., 2004), (Seo et al., 2004), (Pedersen, 2004), etc.

Unsupervised corpus-based algorithms do not directly assign a sense to a target word but rather try to distinguish all possible senses of the given word based on the information they can gain from the unannotated corpora and then discriminate among those senses. Hence, the process is not dependent on the existence of already predefined sense inventories of the target words. Moreover, as a result, unsupervised corpus-based methods can provide us with sense inventories that are a lot more ”tuned” to different domains which is of a great help for different applications like Machine Translation for example. There are two fundamental approaches to unsu-
CHAPTER 2. BASIC APPROACHES TO WORD SENSE DISAMBIGUATION

supervised corpus-based word sense disambiguation: *distributional* and *translational equivalence* approaches. Both are considered *knowledge-lean* approaches as a result of their dependency only on the existing but not necessarily annotated monolingual corpora or on the word-aligned parallel corpora.

### 2.2.1 Distributional Methods

*Distributional* methods make more use of statistics than lexicology. They do not search through the already predefined senses of the target word and choose the one that fits best but rather cluster the words that have similar environments not taking in consideration any of the already established sense inventories for those words. As a result there are no limits to the number of created clusters which in respect denote the granularity of senses the target words acquire. Pedersen (2007) describes the distributional method as an automated and idealized view of the work lexicographers do in the attempt to find a good definition of a word. However, in this case a good definition is relative to the corpus from which it has been extracted. This means that the clusters will only correspond to the senses that are present in the corpus itself and not to the senses that the target word actually has.

Apart from the discrimination among granularity of senses there is another issue that is of significant importance to distributional methods: the automatic labeling of the clusters. It has been often discussed (e.g. Pedersen (2007)) that such a task is quite difficult as well as very significant. One of the possible solutions to the problem as Pedersen (2007) shows is the extraction of separate sets of words that are related to the created clusters and their usage instead of a normal definition. Such words that capture the contextual similarity of the clusters are acquired by *type based* methods of discrimination. Since it is not our prior aim to describe in depth the *distributional* methods for unsupervised WSD, please refer to (Pedersen, 2007) for more information. However, to illustrate the process let us consider the word *line*. If we have two separate clusters representing the rope-like sense and the one connected with communication the *type based* methods will give us words that could be used as labels of the clusters like (*cord, line, tie, cable*) for the first rope-like cluster and (*telephone, line, call*) for the communication cluster.

### 2.2.2 Translational Equivalence Methods

*Translational equivalence* methods, as their name suggests, have something to do with the translation of the target word into another language. Thus, for this purpose parallel corpora (collections of texts placed alongside their translations) become very handy. *Translational equivalence* methods extract the translations of the target word from the parallel corpora and so create its sense inventory. In other words the considered sense inventory of a target word is build up of all the translations of this target word in the parallel corpora. This approach is believed to be very useful for many purposes, e.g. extraction of sense inventories that are more adequate to specific
application (since as noted above, it is often the case that the sense inventories provided by dictionaries, WordNet, or other sources could be either too fine-grained or not fine-grained enough) or automatic derivation of bilingual dictionaries from parallel corpora (which again gives the opportunity of the dictionaries to be more specific to the given domain).

With the rapidly increasing use of large corpora, which are nowadays permanently available and easily extracted from the World Wide Web, unsupervised corpus-based methods have become more and more interesting to the computational linguistic society. Their application does not necessarily require a lot of linguistic knowledge, which makes them flexible in respect to the variety of languages to which they can be applied. Linguistic knowledge, however, naturally contributes a lot to the power and robustness of the unsupervised methods which is a direction in which they could be further developed, especially for the languages for which such knowledge already exists.

In general unsupervised corpus-based algorithms, as Mihalcea et al. (2004a) report, perform poorer than supervised or knowledge-based algorithms. However, as can be seen in Table 2.2 the results presented in evaluation exercises (e.g. Senseval - see Chapter 4 ) become more competitive than they were in previous years.

<table>
<thead>
<tr>
<th>System/Team</th>
<th>Description</th>
<th>Fine</th>
<th>Coarse</th>
</tr>
</thead>
<tbody>
<tr>
<td>wsdii</td>
<td>An unsupervised system using a Lesk-like similarity between context</td>
<td>66.1</td>
<td>73.9</td>
</tr>
<tr>
<td>IIT Bombay</td>
<td>of ambiguous words, and dictionary definitions. Experiments are</td>
<td>65.7</td>
<td>74.1</td>
</tr>
<tr>
<td>Ramakrishnan et al.</td>
<td>performed for various window sizes, various similarity measures.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cymfony</td>
<td>A Maximum Entropy model for unsupervised clustering, using neighboring</td>
<td>56.3</td>
<td>66.4</td>
</tr>
<tr>
<td>(Nia)</td>
<td>words and syntactic structures as features. A few annotated instances</td>
<td>56.3</td>
<td>66.4</td>
</tr>
<tr>
<td></td>
<td>are used to map context clusters to WordNet/Worsmyth senses.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ProBo</td>
<td>A combination of two unsupervised modules, using basic part of speech</td>
<td>54.7</td>
<td>63.6</td>
</tr>
<tr>
<td>Cambridge U. (Preiss)</td>
<td>and frequency information.</td>
<td>54.7</td>
<td>63.6</td>
</tr>
<tr>
<td>cl04-Is</td>
<td>An unsupervised system relying on definition properties (syntactic, semantic</td>
<td>45.0</td>
<td>55.5</td>
</tr>
<tr>
<td>CL Research</td>
<td>(Litkowski)</td>
<td>45.0</td>
<td>55.5</td>
</tr>
<tr>
<td></td>
<td>subsensegroary patterns, other lexical information), as given in a dictionary.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CIAOSENSO</td>
<td>An unsupervised system that combines the conceptual density idea with the</td>
<td>50.1</td>
<td>49.3</td>
</tr>
<tr>
<td>U. Genova (Buscaldi)</td>
<td>frequency of words to disambiguate; information about domains is also</td>
<td>41.7</td>
<td>49.3</td>
</tr>
<tr>
<td></td>
<td>taken into account.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>KUNLP</td>
<td>An algorithm that disambiguates the senses of a word by selecting a</td>
<td>40.4</td>
<td>52.8</td>
</tr>
<tr>
<td>Korea U. (Soo)</td>
<td>substituent among WordNet relatives (synonyms, hypernyms, etc.). The</td>
<td>40.4</td>
<td>52.8</td>
</tr>
<tr>
<td></td>
<td>selection is done based on co-occurrence frequencies, measured on a large</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>corpus.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Duluth-SenseRelate U.</td>
<td>An algorithm that assigns the sense to a word that is most related to the</td>
<td>40.3</td>
<td>51.0</td>
</tr>
<tr>
<td>Minnesota (Pedersen)</td>
<td>possible senses of its neighbors, using WordNet glosses to measure</td>
<td>38.5</td>
<td>48.7</td>
</tr>
<tr>
<td></td>
<td>relatedness between senses.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DFA-LS-Unsup</td>
<td>A combination of three heuristics: similarity between synonyms and the</td>
<td>23.4</td>
<td>23.4</td>
</tr>
<tr>
<td>UNEED (Fernandez)</td>
<td>context, according to a mutual information measure; lexicosyntactic</td>
<td>23.4</td>
<td>27.4</td>
</tr>
<tr>
<td></td>
<td>patterns extracted from WordNet glosses; the first sense heuristic.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DSL-IU-LS-NOSU</td>
<td>An unsupervised method based on (Magnini &amp; Strapparava 2000) WordNet</td>
<td>19.7</td>
<td>32.2</td>
</tr>
<tr>
<td>U.Allicante (Vazquez)</td>
<td>domains; it exploits information contained in glosses of WordNet domains,</td>
<td>11.7</td>
<td>19.0</td>
</tr>
<tr>
<td></td>
<td>and uses “Relevant Domains”, obtained from association ratio over domains</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>and words.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 2.2: Performance and short description for the unsupervised systems participating in the SENSEVAL-3 English lexical sample task. Precision (P) and recall (R) (see Section 4.1) figures are provided for both fine-grained and coarse-grained scoring (Mihalcea et al., 2004a).
2.3 Supervised Corpus-Based

A considerable part of our approach is based on supervised corpus-based methods which is the reason why we will briefly discuss them as well. However, we will only present the information that is most relevant to our work.

Supervised corpus-based methods for word sense disambiguation in respect to the unsupervised ones are considerably more expensive in regard to the human work that needs to be invested in them. This work consists basically of semantic annotation of examples which for a domain independent application, as Ng (1997b) suggests, need to be at least 1000 instances of 3200 different words. In their earlier work Ng and Lee (1996) also report that the necessary human effort for the creation of a corpus of this size is about 16 human-years. Unquestionably, this is a prize quite hight to pay and of course this leads to consequences such as the knowledge acquisition bottleneck or in other words the lack of semantically annotated instances. This unequivocally illustrates the biggest predicament for supervised corpus-based methods but fails to show their effectiveness. In the last decade the methods of this family proved to be more successful than unsupervised or knowledge based ones for which evaluation exercises (see Chapter 4) provide us with evidence. As can be seen in Table 2.3 on page 19 supervised systems reach up to 79.3% accuracy (the accuracy of a system represents its overall performance - exactly how it is measured is described in Section 4.1) which is already a good distance from the MFS classifier baseline that gives us only about 55.2% for the data in the given experiment.

The idea behind supervised methods for WSD is directly connected with the use of machine learning for classification. Those methods automatically learn to make correct predictions as long as they are provided the possibility to have some observations in advance. There are several advantages of the automated attempt for classification: it is most often much more accurate than human-crafted rules because it is data-driven; its flexibility enables its application on variable training data; no extra effort is needed for the creation of additional classifiers, etc. Along with the good aspects, there are some downsides of the supervised machine learning classification - the biggest one, as we just mentioned, is the fact that it depends on huge amounts of annotated training data. The sequence of actions that supervised machine learning methods employ is visualized by Figure 2.1 on page 20.
## CHAPTER 2. BASIC APPROACHES TO WORD SENSE DISAMBIGUATION

<table>
<thead>
<tr>
<th>System/Team</th>
<th>Description</th>
<th>Fine P</th>
<th>Fine R</th>
<th>Coarse P</th>
<th>Coarse R</th>
</tr>
</thead>
<tbody>
<tr>
<td>biats3</td>
<td>A Naive Bayes system, with correction of the a-priori frequencies, by dividing the output confidence of the senses by ( f_{\text{frequency}}(\alpha = 0.2) )</td>
<td>72.9</td>
<td>72.9</td>
<td>79.3</td>
<td>79.3</td>
</tr>
<tr>
<td>U.Bucharest (Grosea)</td>
<td>Kernel methods for pattern abstraction, paradigmatic and syntagmatic information, and unsupervised term proximity (LSA) on BNC, in an SVM classifier.</td>
<td>72.6</td>
<td>72.6</td>
<td>79.5</td>
<td>79.5</td>
</tr>
<tr>
<td>ITC-IRST (Strupperava)</td>
<td>A combination of knowledge sources (part-of-speech of neighboring words, words in context, local collocations, syntactic relations), in an SVM classifier.</td>
<td>72.4</td>
<td>72.4</td>
<td>78.8</td>
<td>78.8</td>
</tr>
<tr>
<td>nusels Nat.U. Singapore (Lee)</td>
<td>Similar to biats3, with different correction function of a-priori frequencies.</td>
<td>72.4</td>
<td>72.4</td>
<td>78.8</td>
<td>78.8</td>
</tr>
<tr>
<td>BCU comb</td>
<td>Basque Country U. (Agirre &amp; Martinez)</td>
<td>An ensemble of decision lists, SVM, and vectorial similarity, improved with a variety of smoothing techniques. The features consist of local collocations, syntactic dependencies, bag-of-words, domain features.</td>
<td>72.3</td>
<td>72.3</td>
<td>78.9</td>
</tr>
<tr>
<td>biats4</td>
<td>Similar to biats3, but with smaller number of features.</td>
<td>72.2</td>
<td>72.2</td>
<td>78.7</td>
<td>78.7</td>
</tr>
<tr>
<td>rslc-comb</td>
<td>U.Bucharest (Popescu)</td>
<td>A regularized least-square classification (RLSC), using local and topical features, with a term weighting scheme.</td>
<td>72.1</td>
<td>72.1</td>
<td>78.6</td>
</tr>
<tr>
<td>biats2</td>
<td>Similar to biats4, but with smaller number of features.</td>
<td>72.1</td>
<td>72.1</td>
<td>78.6</td>
<td>78.6</td>
</tr>
<tr>
<td>BCU english</td>
<td>Similar to BCU.comb, but with a vectorial space model learning.</td>
<td>72.0</td>
<td>72.0</td>
<td>79.1</td>
<td>79.1</td>
</tr>
<tr>
<td>rslc-lin</td>
<td>Similar to rslc-comb, with a linear kernel, and a binary weighting scheme.</td>
<td>71.8</td>
<td>71.8</td>
<td>78.4</td>
<td>78.4</td>
</tr>
<tr>
<td>HLTC HKUST allH KUST (Carpin et al.)</td>
<td>A voted classifier combining a new kernel PCA method, a Maximum Entropy model, and a boosting-based model, using syntactic and collocational features.</td>
<td>71.4</td>
<td>71.4</td>
<td>78.6</td>
<td>78.6</td>
</tr>
<tr>
<td>BALF</td>
<td>U.Catalunya (Escudero et al.)</td>
<td>A system with per-word feature selection, using a rich feature set. For learning, it uses SVM, and combines two binarization procedures: one vs. all, and constraint learning.</td>
<td>71.3</td>
<td>71.3</td>
<td>78.2</td>
</tr>
<tr>
<td>MC-WSD Brown U. (Claramunt &amp; Johnson)</td>
<td>A multiclass averaged perceptron classifier with two components: one trained on the data provided, the other trained on this data, and on WordNet glosses. Features consist of local and syntactic features.</td>
<td>71.1</td>
<td>71.1</td>
<td>78.1</td>
<td>78.1</td>
</tr>
<tr>
<td>HLTC HKUST all2</td>
<td>Similar to HLTC HKUST all, also adds a Naive Bayes classifier.</td>
<td>70.9</td>
<td>70.9</td>
<td>78.1</td>
<td>78.1</td>
</tr>
<tr>
<td>NBC-Fine</td>
<td>NBC-Turney</td>
<td>Semantic and semantic features, using POS tags and pointwise mutual information on a terabyte corpus. Five basic classifiers are combined with voting.</td>
<td>69.4</td>
<td>69.4</td>
<td>75.9</td>
</tr>
<tr>
<td>NBC-Fine2</td>
<td>Similar to NBC-Fine, with a different threshold for dropping features</td>
<td>69.1</td>
<td>69.1</td>
<td>75.6</td>
<td>75.6</td>
</tr>
<tr>
<td>GAMBL</td>
<td>U. Antwerp (Decadt)</td>
<td>A cascaded memory-based classifier, using two classifiers based on global and local features, with a genetic algorithm for parameter optimization.</td>
<td>67.4</td>
<td>67.4</td>
<td>74.0</td>
</tr>
<tr>
<td>SinequaLex</td>
<td>Sinequa Labs (Crestan)</td>
<td>Semantic classification trees, built on short contexts and document semantics, plus a decision system based on information retrieval techniques.</td>
<td>67.2</td>
<td>67.2</td>
<td>74.2</td>
</tr>
<tr>
<td>CLaC</td>
<td>Concordia U. (Lamjiri)</td>
<td>A Naive Bayes approach using a context window around the target word, which is dynamically adjusted</td>
<td>67.2</td>
<td>67.2</td>
<td>75.1</td>
</tr>
<tr>
<td>SinequaLex2</td>
<td>Supervised learning using Support Vector Machines, using local and wide context features, and also grammatical and expanded contexts.</td>
<td>66.0</td>
<td>66.0</td>
<td>73.7</td>
<td>73.7</td>
</tr>
<tr>
<td>UMD, SST4</td>
<td>U. Maryland (Cabezas)</td>
<td>Supervised learning using Support Vector Machines, using local and wide context features, and also grammatical and expanded contexts.</td>
<td>66.0</td>
<td>66.0</td>
<td>73.7</td>
</tr>
<tr>
<td>Prob1</td>
<td>Cambridge U. (Preiss)</td>
<td>A probabilistic modular WSD system, with individual modules based on separate known approaches to WSD (26 different modules)</td>
<td>65.1</td>
<td>65.1</td>
<td>71.6</td>
</tr>
<tr>
<td>SyntLex-3</td>
<td>U. Toronto (Mohammad)</td>
<td>A supervised system that uses local part of speech features and bigrams, in an ensemble classifier using bagged decision trees.</td>
<td>64.6</td>
<td>64.6</td>
<td>72.0</td>
</tr>
<tr>
<td>UNED</td>
<td>UNED (Artilles)</td>
<td>A similarity-based system, relying on the co-occurrence of nouns and adjectives in the test and training examples.</td>
<td>64.1</td>
<td>64.1</td>
<td>72.0</td>
</tr>
<tr>
<td>SyntLex-4</td>
<td>Similar to SyntLex-3, but with unified decision trees.</td>
<td>63.3</td>
<td>63.3</td>
<td>71.1</td>
<td>71.1</td>
</tr>
<tr>
<td>CLaC2</td>
<td>Syntactic and semantic (WordNet hyponyms) information of neighboring words, fed to a Maximum Entropy learner. See also CLaC1</td>
<td>63.1</td>
<td>63.1</td>
<td>70.3</td>
<td>70.3</td>
</tr>
<tr>
<td>SyntLex-1</td>
<td>Bagged decision trees using local POS features. See also SyntLex-3.</td>
<td>62.4</td>
<td>62.4</td>
<td>69.1</td>
<td>69.1</td>
</tr>
<tr>
<td>SyntLex-2</td>
<td>Similar to SyntLex-1, but using broad context part of speech features.</td>
<td>61.8</td>
<td>61.8</td>
<td>68.4</td>
<td>68.4</td>
</tr>
<tr>
<td>Prob2</td>
<td>Similar to Prob1, but invokes only 12 modules.</td>
<td>61.9</td>
<td>61.9</td>
<td>69.3</td>
<td>69.3</td>
</tr>
<tr>
<td>Dushul-ELSS</td>
<td>U.Minnesota (Pedersen)</td>
<td>An ensemble approach, based on three bagged decision trees, using unigrams, bigrams, and co-occurrence features</td>
<td>61.8</td>
<td>61.8</td>
<td>70.1</td>
</tr>
<tr>
<td>UJAE</td>
<td>U.Jaen (Garcia-Vega)</td>
<td>A Neural Network supervised system, using features based on semantic relations from WordNet extracted from the training data</td>
<td>61.3</td>
<td>61.3</td>
<td>69.5</td>
</tr>
<tr>
<td>R2D2</td>
<td>U. Alkante (Vazquez)</td>
<td>A combination of supervised (Maximum Entropy, HMM Models, Vector Quantization, and unsupervised (domains and conceptual density) systems.</td>
<td>63.4</td>
<td>52.1</td>
<td>69.7</td>
</tr>
<tr>
<td>ITC-IRST (Strupperava)</td>
<td>A generalized pattern abstraction system, based on boosted wrapper induction, using only few syntactic features.</td>
<td>70.6</td>
<td>50.5</td>
<td>76.7</td>
<td>54.8</td>
</tr>
<tr>
<td>NRC-Course</td>
<td>Similar to NRC-Fine; maximizes the coarse score, by training on coarse senses.</td>
<td>48.5</td>
<td>48.5</td>
<td>75.8</td>
<td>75.8</td>
</tr>
<tr>
<td>NRC-Course2</td>
<td>Similar to NRC-Course, with a different threshold for dropping features.</td>
<td>48.4</td>
<td>48.4</td>
<td>75.7</td>
<td>75.7</td>
</tr>
<tr>
<td>DSLI-UA-LS-SU</td>
<td>U.Alicante (Vazquez)</td>
<td>A maximum entropy method and a bootstrapping algorithm (“re-training”) with iterative feeding of training cycles with new high-confidence examples.</td>
<td>78.2</td>
<td>31.0</td>
<td>82.8</td>
</tr>
</tbody>
</table>

Table 2.3: Performance and short description of the supervised systems participating in the SENSEVAL-3 English lexical sample Word Sense Disambiguation task. Precision (P) and recall (R) (see Section 4.1) figures are provided for both fine-grained and coarse-grained scoring (Mihalcea et al., 2004a).
CHAPTER 2. BASIC APPROACHES TO WORD SENSE DISAMBIGUATION

The attempt to solve a problem with a supervised machine learning methods starts first with the identification of the data set that is feasible for the given task (see Section 2.3.2). Then this set is pre-processed (Section 2.3.3) and prepared to be divided in the required training and test sets - Section 2.3.4. Training, however, can first be accomplished after a proper algorithm is depicted together with the parameters that are most likely to give good results (Section 2.3.5). Once this setup is ready, the test set is used to "assess" the future classifier. If the evaluation is satisfying the classifier is finished but in case the evaluation yields not acceptable results the process could be reactivated at any of its previous states. The main idea behind this machinery is that once provided with a training set the selected algorithm induces a specific classifier which is normally described as a hypothesis of a function that is used to map the examples (in our case the test examples) to the different classes that have already been observed in the training cases. We already know, that human annotators label the training data manually, but a good question at this point is where do they know which set of senses they can use for classification. The answer is easy and it is hidden behind the employment of the so called sense inventories.

2.3.1 Sense Inventories

Most often machine readable dictionaries are the sources of the sense inventories that are used for the manual annotation of supervised WSD corpora and thus for the creation of the training and test sets. The most extensively used ones are nowadays WordNet (Miller, 1990; Fellbaum, 1998) for English and EuroWordNet (Vossen, 1998) (e.g. the DSO corpus Ng and Lee (1996), SemCor (Miller et al., 1993), Open Mind Word Expert (OMWE) (Mihalcea and Chklovski, 2003)) for languages as Dutch, Italian, Spanish, German, French, Czech, and Estonian. In Section 2.3.2 we point out which sense inventories have been used for the corresponding sense-annotated corpora but we do not explicitly keep track of their version (e.g. Wordnet 1.6, 1.7, 1.7.1, 2.0) since...
automatic mapping to the most recent version is usually also provided. Other such resources are: Hector (Atkins, 1993), Longman Dictionary of Contemporary English (Procter, 1978), BalkaNet (Stamou et al., 2002), etc.

Theoretically, each dictionary could be used as a sense inventory for WSD. However, there are several problems coming along. First, dictionaries are not always freely available for research, which was the reason why WordNet became in fact the standard sense inventory for the last decade. However, it is still being argued if it is good as such. Since WordNet distinguishes between the senses of each word in an extremely fine-grained manner, it is often hard to use it for WSD, hence there are cases where a coarser distinction is desirable. Calzolari et al. (2002) even argue that the use of WordNet as a sense inventory in WSD yields worse results than using traditional dictionaries. However, it is not WordNet itself but the predefined sense inventory as such that appears to hinder supervised word sense disambiguation. There is a large number of attempts to solve the latter problem and although none of them completely succeed WordNet will continue to be the standard sense inventory for WSD.

Another problem in respect to sense inventories is their compatibility. Each dictionary has its own granularity and representation of senses, which are normally extremely hard if not even impossible to map against each other. Thus systems that use different sense inventories are impossible to compare, since their performance is bound to the inventory they use. Of course, since this issue is a well known problem already, evaluation exercises (see Chapter 4) use a single sense inventory for all the participating systems or it is required that those inventories that are different from the standard one provide mapping to it.

### 2.3.2 Source Corpora

One of the biggest problems for supervised word sense disambiguation (the knowledge acquisition bottleneck problem) is the fact that there are very few annotated corpora that can be used in order good and reliable broad-coverage systems to be trained. This is due to the fact that the creation of such corpora requires a highly laborious human effort. The huge dependency of the method on the provided corpora is the reason why for languages other than English, supervised WSD is extremely difficult if not even impossible. Below follows a brief description of the main data sources for supervised WSD.

**Senseval** provides several corpora not only for English but as well for languages as Italian, Basque, Catalan, Chinese, Romanian and Spanish. The most recent edition of Senseval (Senseval-3) resulted in the following annotated corpora:

- **English all words** - 5000 words were tagged from Penn Treebank text (Marcus et al., 1993) with WordNet as senses.

- **English lexical sample** - 57 words collected via The Open Mind Word Expert interface (Michalcea and Chklovski, 2003) with WordNet sense inventory.
- **Italian all words** - 5000 words from the Italian Treebank (Bosco et al., 2000) semantically tagged according to the sense repository of ItalWordNet (Roventini et al., 2000).

- **Italian lexical sample** - 45 words with sense inventory specially developed for the task (Italian MultiWordNet for Senseval-3)

- **Basque lexical sample** - 40 words with sense inventory manually linked to WordNet.

- **Catalan lexical sample** - 45 words with sense inventory manually linked to WordNet.

- **Spanish lexical sample** - 45 words for which the sense inventory was again specially developed and was manually linked to WordNet.

- **Chinese lexical sample** - 20 words with sense inventory according to the HowNet knowledge base (Dong, 1998).

- **Romanian lexical sample** - 50 words for which senses are collected from the new Romanian WordNet, or DEX (a widely recognized Romanian dictionary). The data is collected via the OMWE (Romanian edition) (see Section 4.5.2).

- **Swedish lexical sample task** unfortunately was cancelled and thus no corpora were provided.

**SemCor** - (Miller et al., 1993) is a lot broader in coverage than Senseval. Around 23 346 words are gathered from The Brown Corpus (about 80%) and the novel *The Red Badge of Courage* (about 20%) and for a sense inventory, WordNet is used.

**The Open Mind Word Expert** - (Mihalcea and Chklovski, 2003) consists of 230 words from the Penn Treebank, *Los Angeles Times* collection, Open Mind Common Sense and others. Here, WordNet is used as sense repository, too. This corpus, however, grows daily since it is being created mostly by volunteers that manually annotate examples on the Web. Further information about the OMWE can be found in Section 4.5.2.

**The DSO corpus** - Ng and Lee (1996) gathered 191 words from The Brown Corpus and The Wall Street Journal and annotated them with senses according WordNet.

**Hector** - (Atkins, 1993) account for about 300 words from the A 20M-word pilot for the British National Corpus\(^3\) (BNC) for which the sense inventory is picked up as well from Hector.

**HKUST-Chinese** has approximately 38 725 sentences again from the HowNet knowledge base (Dong, 1998).

\(^3\)http://www.natcorp.ox.ac.uk/
Swedish corpus - 179 151 instances with the Gothenburg lexical database as inventory and the SUC Corpus as its corpus.

Image captions - 2 304 words from image captions of an image collection and WordNet as source inventory.

All this shows how immense the efforts to solve this exceptionally important problem (the knowledge acquisition bottleneck) are and how despite this considerable attempt the biggest obstacle for supervised WSD is still on the way. It will surely employ further on a great deal of commission but until then we could continue discussing the process of WSD with a supervised system and the next step which should be taken from deciding on a sense inventory and having semantically annotated corpus is the data preprocessing.

2.3.3 Data Preprocessing

Having chosen the corpora and extracted the needed parts of it, however, does not already mean that we are ready to go ahead and train a classifier. There is another intermediate step which is not necessarily difficult but in most cases quite complex. It consists of data preprocessing and preparation for the extraction of the training and test examples. The former is believed to have a significant impact on the performance of supervised WSD systems. It conditions the quality of the discovered patterns and makes it possible to find useful information from initial data. The most basic preparation of the data is the filtering of any unwanted special or meta characters, which are normally existent in large corpora or the removal of exceptional punctuation marks. Such "noise" in the data does not only lead to decrease in accuracy, but depending on the further processing and the software that is used it might even lead to other problems (e.g. part-of-speech (POS) tagger (usually used further in the process) not handling special punctuation characters).

The second part in this process is POS tagging. Depending on the decision that is made about the information which needs to be contained in the examples, multi-words (a multi-word expression is a lexeme build up of a sequence of at least two lexemes that has properties that are not predictable from the properties of the individual lexemes), lexical dependencies and n-gram (sub-sequence of n items, in our case words, from a given sequence) detection might be necessary as well.

When the above mentioned basic preprocessing and POS tagging are done, the data can be finally used for extraction of examples needed for training and testing of the classifier (also called word-expert). This step is further described in the following section.

2.3.4 Feature Vectors

The examples in the training set are constructed as feature vectors (FV). Feature vectors are fixed-length n-dimensional vectors of features (where n is the number of selected attributes/features)
that represent the relevant information about the example they refer to. Those attributes are predefined and picked up uniformly for each instance so that each vector has the same number and type of features. There are three types of features from which the selection can be made: local features, global features and syntactic dependencies. Local features describe the narrow context around the target word - bigrams, trigrams (other n-grams) that are generally constituted by lemmas, POS tags, word-forms together with their positions with respect to the target word. Global features on the other hand consider not only the local context around the word that is being disambiguated, but the whole context (whole sentences, paragraphs, documents) aiming to describe best the semantic domain in which the target word is found. The third type of features that can be extracted are the syntactic dependencies. They are normally extracted at a sentence level using heuristic patterns and regular expressions in order to gain a better representation for the syntactic cues and argument-head relations such as subject, object, noun-modifier, preposition etc.

Commonly, features are selected out of a predefined pool of features (PF), which consists out of the most relevant features for the approached task. We compiled such PF (see Appendix B) being mostly influenced for our choice by (Dinu and Kübler, 2007) and (Mihalcea, 2002).

Let us consider a simple example. Given the sentences in (2) we will attempt to create FVs that represent the occurrences of the noun bank in those sentences.

(2)

1. "He sat on the bank of the river and watched the currents."
2. "He operated a bank of switches."

For the sake of simplicity at this point we chose a minimum number of features from the pre-defined pool of features. The corresponding set consists only of local features \((cw_{-2}, cw_{-1}, cw_0, cw_{+1}, cw_{+2}, cp_{-2}, cp_{-1}, cp_0, cp_{+1}, cp_{+2})\). The resulting vectors thus can be seen in Table 2.4 below. Since we do not explain in detail exactly how those vectors are put together, please refer to Chapter 6 where we follow stepwise the process of FV construction.

<table>
<thead>
<tr>
<th>#</th>
<th>cw_{-2}</th>
<th>cw_{-1}</th>
<th>cw_0</th>
<th>cw_{+1}</th>
<th>cw_{+2}</th>
<th>cp_{-2}</th>
<th>cp_{-1}</th>
<th>cp_0</th>
<th>cp_{+1}</th>
<th>cp_{+2}</th>
<th>class</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>on</td>
<td>the</td>
<td>bank</td>
<td>of</td>
<td>the</td>
<td>P</td>
<td>DET</td>
<td>N</td>
<td>P</td>
<td>DET</td>
<td>bank%1:17:01::</td>
</tr>
<tr>
<td>2.</td>
<td>operated</td>
<td>a</td>
<td>bank</td>
<td>of</td>
<td>switches</td>
<td>V</td>
<td>DET</td>
<td>N</td>
<td>P</td>
<td>N</td>
<td>bank%1:14:01::</td>
</tr>
</tbody>
</table>

Table 2.4: Feature vectors for the sentences in (2).

As we have already noted, supervised learning is based on the training data that has been manually annotated beforehand while unsupervised learning does not need such predefined sense inventory. Thus, a very important part of the FV is the class label that represents the classification of this vector (this is the value of the output space that is associated with a training example) which in our case is to be found in the class column. As its value we have selected the correct sense of the noun bank in the sentence represented by the sense key that this sense
has in WordNet (Fellbaum, 1998). Refer to Section 6.4.1 for further information on the final FVs that we constructed for our system.

The test set is indeed very similar to the training set. However, since we need to evaluate the system, no class labels are included in the feature vectors. System performance is normally believed to increase with the amount of training data and thus usually the training set is a relatively larger portion of the whole data than the test set. It is possible to divide the data in 5 portions where 4 portions will be the training set and 1 portion the test set resulting in a ratio 4:1. Other often used ratios are 2:1 and 9:1. According to Palmer et al. (2007) a division of 2:1 may provide a more realistic indication of a system’s performance, since a larger test set is considered. Still, we know that labeled data is not plenty, which is why it is preferably taken for training and not for testing. By dividing the data in any particular split however, a bias is unquestionably involved. Consequently a better generalization accuracy measurement has to be used in real experiments - n-fold cross-validation or in particular 10-fold cross-validation (Weiss and Kulkowski, 1991).

In n-fold cross-validation the data is divided into n number of folds for which it is desirable that they are of equal size. Accordingly n separate experiments are performed, and in each experiment (also called fold) n-1 portions of the data is used for training and 1 for testing, in such a way that each portion is used as a test item exactly once. If n equals the sample size (the size of the data set) the process is called leave-one-out cross-validation.

### 2.3.5 Supervised WSD Algorithms

One of the main decisions which needs to be met when designing a supervised WSD system is the choice of the algorithm that is to be employed. In Table 2.5 on page 26 is a basic overview of the most often used alternatives as well as some literature where more information can be found about them. A short description of the algorithms is provided as well in order to give an outline of their usage and importance.
## Probabilistic methods

Categorize each of the new examples by using calculated probabilistic parameters. The latter convey the probability distributions of the categories and the contexts that are being described by the features in the feature vectors.

Naïve Bayes (Duda et al., 2001) is one of the simplest representatives of probabilistic methods that presupposes the conditional independence of features given the class label. The main idea is that an example is created by selecting the most probable sense for the instance and as well for each of its features independently considering their individual distributions. The algorithm uses the Bayes inversion rule (Fienberg, 2006). It is often considered that the independence assumption is a problem for Naïve Bayes and thus alternative algorithms as the decomposable model by (Bruce and Wiebe, 1994) have been developed.

Maximum entropy (Berger et al., 1996) is another quite robust probabilistic approach that combines stochastic evidence from multiple different sources without the need for any prior knowledge of the data.

## Discriminating rules

Assign a sense to an example by selecting one or more predefined rules that are satisfied by the features in the example and hence selecting the sense that the predictions of those rules yield. Examples for such methods are Decision lists and Decision trees.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Algorithms</th>
<th>Literature</th>
</tr>
</thead>
<tbody>
<tr>
<td>Probabilistic</td>
<td>Naïve Bayes</td>
<td>(Duda et al., 2001)</td>
</tr>
<tr>
<td></td>
<td>Maximum Entropy</td>
<td>(Berger et al., 1996)</td>
</tr>
<tr>
<td>Similarity-Based</td>
<td>Vector Space Model</td>
<td>(Yarowsky et al., 2001)</td>
</tr>
<tr>
<td></td>
<td>k-Nearest Neighbor</td>
<td>(Ng and Lee, 1996; Ng, 1997a)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(Daelemans et al., 1999)</td>
</tr>
<tr>
<td>Discriminating Rules</td>
<td>Decision Lists</td>
<td>(Yarowsky, 1995; Martínez et al., 2002)</td>
</tr>
<tr>
<td></td>
<td>Decision Trees</td>
<td>(Mooney, 1996)</td>
</tr>
<tr>
<td>Rule Combination</td>
<td>AdaBoost</td>
<td>(Schapire, 2003)</td>
</tr>
<tr>
<td></td>
<td>LazyBoost</td>
<td>(Escudero et al., 2000a,b, 2001)</td>
</tr>
<tr>
<td>Linear Classifier</td>
<td>Perceptron</td>
<td>(Mooney, 1996)</td>
</tr>
<tr>
<td></td>
<td>Winnow</td>
<td>(Escudero et al., 2000b)</td>
</tr>
<tr>
<td></td>
<td>Exponentiated-Gradient</td>
<td>(Bartlett et al., 2004)</td>
</tr>
<tr>
<td></td>
<td>Widrow-Hoff</td>
<td>(Abdi et al., 1996)</td>
</tr>
<tr>
<td></td>
<td>Sleeping Experts</td>
<td>(Cohen and Singer, 1999)</td>
</tr>
<tr>
<td>Kernel-Based</td>
<td>Support Vector Machines</td>
<td>(Murata et al., 2001)</td>
</tr>
<tr>
<td></td>
<td>Kernel Principal Component Analysis</td>
<td>(Boser et al., 1992; Lee and Ng, 2002)</td>
</tr>
<tr>
<td></td>
<td>Regularized Least Squares</td>
<td>(Cristianini and Shawe-Taylor, 2000)</td>
</tr>
<tr>
<td></td>
<td>Average Multiclass Perceptron</td>
<td>(Popescu, 2004)</td>
</tr>
<tr>
<td>Discourse Properties</td>
<td>Yarowsky Bootstrapping</td>
<td>(Ciaramita and Johnson, 2004)</td>
</tr>
</tbody>
</table>

Table 2.5: Supervised word sense disambiguation algorithms.
CHAPTER 2. BASIC APPROACHES TO WORD SENSE DISAMBIGUATION

Decision lists (Yarowsky, 1995; Martinez et al., 2002) as their name suggests are simple ordered lists of rules that are of the form \((\text{condition, class, weight})\). Such rules are usually more easy to understand if thought of as if-then-else rules: if the condition is satisfied then the according class is assigned. However, in the form of the rules that we provided above there is also a third parameter that is taken into account - weight. Weights are used to determine the order of the rules in the list. Higher weights position the rules higher in the list and respectively lower weights mean that the rules can be found further down in the ordered list of rules. The order in the decision lists is important during classification since the rules are tested sequentially and the first rule that "succeeds" is used to assign the sense to the example. Usually the default rule in a list is the last one that accepts all remaining cases.

Decision trees (Mooney, 1996) are basically very similar to decision lists however not this often used for word sense disambiguation. They as well use classification rules but this time the rules are not ordered in a list but as an \(n\)-ary branching tree structure that represents the training set. Hence every branch of the tree represents some rule that is used to test the conjunction of features and to provide a prediction of the class label encoded in the terminal node (also called leaf node is a node in a tree data structure that has no child nodes). Some of the problems with decision trees, which makes them not really handy for WSD, are their computational cost and the data fragmentation (breaking up the data into many pieces that are not close together) that they employ. The latter leads to immense increase in computation if larger feature spaces are used. The same result is also triggered by the use of a large number of examples, however, if fewer training instances are provided a relative decrease in the reliability of the predictions for the class label can be observed.

Rule combination for supervised word sense disambiguation means that a set of homogeneous classification rules is combined and learned only by a single algorithm.

AdaBoost (Schapire, 2003) is a very often used rule combination algorithm. It combines multiple classification rules into a single classifier. The power of AdaBoost is based on the fact that the classification rules must not necessarily be very accurate but after they are combined the resulting classifier has an arbitrarily low error rate.

Linear Classifier or also called binary classifier achieved in the last few decades considerably low results and thus the highest interest to them was in the field of Information Retrieval. Those kind of classifiers decide on the classification label based on the linear combination of the features in their FVs. They aim to group the instances with the most similar feature values. The limited amount of work on linear classifiers has resulted in several articles as for example (Mooney, 1996; Escudero et al., 2000b; Bartlett et al., 2004; Abdi et al., 1996; Cohen and Singer, 1999). In case a non-linear problem has to be decided, for which the expressivity of the linear classifiers is not enough, suggestions for the use of kernel functions (kernel methods) have been made.
Kernel-based are the methods that try to find more general types of relations (and not linear as just noted above) in the FVs. Their popularity in the past few years has notably increased which can be seen from their growing participations in recent conferences as Senseval-3 for example (see Section 4.5). Examples of applications of kernel methods in supervised approaches are the ones described by Murata et al. (2001); Boser et al. (1992); Lee and Ng (2002); Cristianini and Shawe-Taylor (2000); Carpuat et al. (2004); Wu et al. (2004); Popescu (2004); Ciaramita and Johnson (2004).

One of the most popular kernel-methods is the Support Vector Machines (SVMs) presented by Boser et al. (1992). As Márquez et al. (2007) report, SVMs are established around the principle of Structural Risk Minimization from the Statistical Learning Theory (Vapnik, 1998). In their basic form SVMs are linear classifiers that view input data as two sets of vectors in an n-dimensional space. They construct a separating hyperplane (in geometry hyperplane is a higher-dimensional abstraction of the concepts of a line in a n-dimensional space) in that space, which is used to separate two data sets. To calculate the margin between those data sets, two parallel hyperplanes are constructed - one on each side of the separating hyperplane, which are directed to the two data sets. Naturally, a good separation is considered to be achieved by the hyperplane that has the largest distance to the neighboring data sets. In cases where non-linear classifiers are desired the selected SVM can be used with a kernel-function.

Discourse properties are considered by the Yarowsky bootstrapping algorithm (Yarowsky, 1995). This algorithm is semi-supervised (see Section 2.4) which makes it hardly comparable with the other algorithms in that section but it is considered (Márquez et al., 2007) relatively important for the following work on bootstrapping for WSD. It uses either automatically or manually annotated training data that is supposed to be complete (to represent each of the senses in the set) but not necessarily big. This initially smaller set is used together with a supervised learning algorithm to annotate other examples. If for a given example the annotation is being accomplished with a higher degree of confidence it is added to the "seed" set and the process is further continued.

Similarity-based is a family of methods that are most relevant to our thesis and thus we provide some more in-depth information about them. However, our aim is still the attempt to give an overview of those methods so that a better understanding of our use of them can be accomplished. Approaches of this kind are very often used in supervised WSD because they carry out the disambiguation process in a very simple way. They classify a new example via a similarity metric that compares the latter to previously seen examples and assigns a sense to it - usually this is the MFS in a pool of most similar examples. During the years, probably because of its increased usage, the approach has gained a wide variety of names: instance-based, case-based, similarity-based, example-based, memory-based, exemplar-based, analogical. As a result of the fact that the data is stored in the memory without any restructuring or abstraction
this method is also often called lazy learning method. Memory-based learning (MBL) has a long history (Stanfill and Waltz, 1986; Cost and Salzberg, 1993; Kolodner, 1993; Aha et al., 1991; Aha, 1997; Cover and Hart, 1967; Devijver and Kittler, 1982). As Daelemans et al. (2007) report a memory-based learning system is essentially build up of two components - a memory-based learning component and a similarity-based performance component. The fundamental architecture of a system based on the similarity of the examples as given to us by Daelemans et al. (2007) is nicely visualized in Figure 2.2.

![Figure 2.2: General architecture of an memory-based learning system (Daelemans et al., 2007).](image)

The learning component (to be found in the upper part of Figure 2.2) has a very straightforward task, namely adding training examples directly into memory without any processing. The performance component uses the stored examples from the learning component to classify the new instances by choosing the label that is with the highest frequency among the most similar previously seen examples.

The similarity between the instances can be determined by various distance metrics (i.e. Information-Gain and Gain Ratio Feature Weighting, Modified Value Difference metric, Overlap and Levenshtein metrics, Cosine metric, Jeffrey Divergence Distance metric, Dot-Product Distance metric, Chi-Squared and Shared Variance Feature Weighting, Distance-Weighted Class Voting, etc.). For more comprehensive information on the distance metrics and the possible choice among them refer to (Daelemans et al., 2007).

In case there is a situation in which two or more possible answers receive the same ranking...
(the so called tie) a tie breaking resolution method is used - incrementing the set of nearest neighbors until the tie is resolved or randomly choosing between the possibilities.

Do all examples have the exact same importance, do they have equal weight? Salzberg (1990); Zhang (1992) and Aha et al. (1991) have discussed the issue of some examples being better than others trying to find the answer of this question. This is called exemplar weighing in MBL and it represents the idea that some examples are more regular, more typical or even more reliable than others. Of course, the benefits for the classification accuracy that considers the weighing of the examples can be great. Ng (1997a) reports that the accuracy achieved by the improved exemplar-based classifier is comparable to the accuracy on the same data set obtained by the Naive-Bayes algorithm which Mooney (1996) classifies as having the highest disambiguation accuracy among seven state-of-art machine learning algorithms.

2.4 Semi-Supervised Corpus-Based

By now we have described the advantages and disadvantages of supervised and unsupervised word sense disambiguation approaches and tried to give a basic overview of the work that has been done on them in the recent few years. Of course, having two completely opposing approaches naturally leads to research in the area of their convergence. It resulted in semi-supervised WSD approaches. In the discourse properties part of Section 2.3.5 we have already mentioned an approach of this family - the Yarowsky bootstrapping algorithm (Yarowsky, 1995).

Many scientific reports have confirmed that unlabeled data, used together with a small set of labeled data, usually produces considerable improvement in the performance of WSD systems. The acquisition of labeled data for any disambiguation task often requires competent and expensive human labour for manually classified training examples to be created. The cost associated with the latter process may respectively make a fully manually labeled training set infeasible, whereas acquisition of unlabeled data is relatively straightforward and connected with extremely low-cost effort. This is where semi-supervised word sense disambiguation appears to be very valuable. There are many different methods that can be used for semi-supervised learning: generative mixture models, self-training, co-training, transductive support vector machines and graph-based methods.

Since supervised WSD suffers from a lack of labeled data to train on and unsupervised WSD has huge amounts of not labeled data to work with, the midpoint is where semi-supervised methods try to automatically label the examples that are not disambiguated manually. For this purpose the semi-supervised approaches use a relatively small number of human annotated examples as seeds for the automatic annotation as in (Yarowsky, 1995). The result of this attempt yields a large labeled data collection that can be used as a training set with a normal supervised learning algorithm. Therefore we can even classify semi-supervised learning as special case of supervised learning where the extra unlabeled data is used to improve the predictive power. As Pham et al. (2005) conclude, semi-supervised word sense disambiguation algorithms are either
CHAPTER 2. BASIC APPROACHES TO WORD SENSE DISAMBIGUATION

bootstrapping algorithms (Blum and Mitchell, 1998; Abney, 2002) or transductive learning algorithms (Joachims, 1999; Blum and Chawla, 2001; Joachims, 2003). Some general research on that particular family of methods can be found as well in (Chapelle et al., 2006; Seeger, 2001; Abney, 2008).

A very valuable survey on the semi-supervised learning methods is presented by Zhu (2008). The author divides this family of learning methods in several groups most important of which are: Generative Models, Self-Training and Co-Training. We will again not discuss those methods in detail, but it is beneficial for the understanding of our work to attain an overview of the most important ones of them.

**Generative Models** (Nigam et al., 2000; Baluja, 1998; Fujino et al., 2005) are considered to be among the oldest semi-supervised learning methods. They assume a model with an identifiable mixture distribution. In case a big amount of unlabeled data is present, the mixture components can be identified. Consequently, a single labeled example per component can be used in order determination of the mixture distribution to be performed. Such mixture components are as well known as *soft clusters*.

**Self-Training** (Yarowsky, 1995; Riloff et al., 2003; Maeireizo et al., 2004; Rosenberg et al., 2005) is based on the training of a classifier with a small amount of labeled data, and its further usage for the classification of the unlabeled set of examples. The examples that are most confidently labeled are then added to the training set. The classifier is then trained again on the new set and the procedure is repeated again.

**Co-Training** (Blum and Mitchell, 1998; Mitchell, 1999) as Zhu (2008) reports is centered around several assumptions: features can be split into two sets, each sub-feature set is sufficient to train a good classifier and the two sets are conditionally independent given the class. Thus those two sub-feature sets are used to train two separate classifiers. Then both of the classifiers label the yet not annotated data and try to teach each other with the examples they most confidently labeled. The classifiers are then trained again and the same process repeats.

In Section 6.1 we provide a rough overview of our system, which we classify as well as semi-supervised word sense disambiguation.
Chapter 3

Comparability for WSD Systems

Along our survey we mentioned multiple word sense disambiguation systems. This wide range of approaches, however, leads to a great number of difficulties when they need to be compared. Still, how to compare systems that differ so much from each other? Can you say that cow milk is better than soy milk, when they are obtained in two entirely contrary ways? Or can you then compare the chocolate that is made out of cow and soy milk and say which is best? There are many similar issues when we talk about comparability of WSD systems most of which find common ground in the different ways in which they are build. In the following sections we look at some of those dissimilarities of WSD systems and as well at the ways in which they can be handled so that actual comparison between systems can be made possible. This is relatively important for our work as well, since the system that we designed needs to be compared to others of its kind.

3.1 Differences between WSD Systems

Differences in WSD systems may arise at each decision point of the construction of the system. For example, let us recall again Figure 2.1 on page 20. On each of the steps we take where a choice between different options can be made (i.e. data collection, data preprocessing, algorithm selection) we determine the setting of the system out of a huge pool of altered settings.

For instance let us consider the situation in which we have to choose the data collection that we are going to use. It makes a big difference if we pick up a corpus from a technical or highly specific domain or a corpus consisting of general texts with a wider domain coverage. Domain specific texts use only a subset of the senses of the words in them, namely the senses that are relevant to the given domain, whereas general texts cover a bigger range of the senses of the words that they consist of. Ide and Véronis (1998b) report that in the Wall Street Journal corpus (which is used extremely often in word sense disambiguation tasks) certain senses of words that are usually used for testing such as line are completely absent. Hence, the employment of different corpora in different WSD systems (especially corpora on entirely dissimilar domains
where sense inventories and considerably different levels of frequencies for a given word and/or sense are assembled) turns out to be pointless if an attempt to compare the systems’ results is planned further on in the process.

Furthermore, the number of words that is used for testing will be as well significantly important to the results a system can accomplish. As Preiss (2006) reports Bruce and Wiebe (1994) used only a single word (*interest*) for the evaluation of their system, while Stevenson (1999) evaluated on all of the labeled words in the SemCor corpus (see Section 2.3.2).

However, it is not only the number of test words that is used but as well their type - different parts of speech, degree of ambiguity, the degree of clarity between the distinctions amongst the senses of a word, the degree of sense granularity (fine sense distinctions vs. coarse sense distinctions), the frequency of their appearance in a metonymic or a metaphoric way, the inter-annotator agreement (IAA) (IAA is the percentage of times that the annotators chose the same sense for the targeted instance. In some cases the latter can have worse accuracy than randomly picking a sense - (Ahlsvede, 1995) reports variations between 63.3% and 90.2% agreement among annotators) and so on and so forth.

This more than complex constellation greatly serves to demonstrate the vivid need for what Gale et al. (1992) have proved to be an inseparable part of the evaluation of word sense disambiguation systems - upper and lower bound for evaluation. The upper bound represents the highest possible accuracy that a system could have. Since it is unrealistic to expect that an automated method is better in distinguishing between senses than a human is, the upper bound is set to be the agreement among the human annotators. The lower bound (also called baseline) on the other hand represents the performance of a very basic approach that has proved to yield relatively good results. It serves to delineate the lowest result that a system should have, since such a baseline result is already easily gained.

### 3.2 Most Frequently Used Baselines

In the following section we discuss the most often used approaches to gain a comparable baseline for a WSD system. Unfortunately it is not always the case that they provide the needed information for a good comparison between WSD systems. Thus in Chapter 4 we will present another possibility for the evaluation of word sense disambiguation approaches.

#### 3.2.1 The Lesk Algorithm

The Lesk algorithm (Lesk, 1986), which we presented in Section 2.1.1, is one of the baselines usually used in WSD systems. However, the application of the original algorithm is in most cases more than is actually needed and therefore various simplifications or other modifications of it are normally taken into account e.g. variations of the context window to which the algorithm is applied, identical word forms that occur in the same definition are discarded, stemming or case
corrections are left aside, usage of phrase filtering, etc. As reported by Mihalcea (2007) the Lesk algorithm reaches accuracy of 50-70% on a set of ambiguous word pairs, which were manually annotated (Oxford Advanced Learner’s Dictionary was used as sense inventory).

### 3.2.2 Most Frequent Sense

The MFS (Gale et al., 1992) is even more straightforward than the Lesk algorithm. According to the statistics of the frequencies of targeted words, which are normally drawn from different corpora, the instances are labeled with the most frequent of the senses of the word that is disambiguated.

Another option for gaining the frequencies of the senses in the corpus on which it has been worked on is by performing domain analysis as in (Koeling et al., 2005). Extracting the frequencies of the current corpus, however, triggers in most cases the "the chicken or the egg" problem since this already presupposes sense tagging of the text and thus an existing WSD system on the first place.

The MFS approach has a relatively good performance. It is based on (Zipf, 1945) - the Zipfian distribution (a specific sense has a dominant frequency of occurrence, whereas all remaining senses of the target word share a considerable decrease in the frequency of occurrence) and reaches accuracy of about 55.2%.

### 3.2.3 Random Choice

The random choice baseline assumes equal weights for all senses of the target word in order to choose one of them. Usually, however, the choice of the sense is not completely random but restrained by the POS tag of the targeted sense if such is provided beforehand. If not, all possibilities are considered equal.
Chapter 4

Evaluation of WSD Systems

In the previous Chapter we looked at the wide variety of factors for comparability of word sense disambiguation systems and noted that therefore in the history of WSD there have been many cases where different approaches could not be appropriately compared. The fruitful discussion that has been led on the workshop "Evaluating Automatic Semantic Taggers" (Resnik and Yarowsky, 1997a,b; Kilgarriff, 1997) has started the first open, community-based evaluation exercise, Senseval1, for WSD that can even be thought of as extremely successful and a lot more sophisticated extension than the upper and lower bound proposal by Gale et al. (1992). There have been other not so fortunate attempts as the ARPA-sponsored Message Understanding Conferences (i.e. (ARPA, 1993)), Text Retrieval Conferences (i.e. (Harman, 1993, 1995)) and many others.

Since in the last few years Senseval turned out to be a good standard for evaluating WSD systems and because we chose to use it in our approach, in the following chapter we will give a brief historical overview of the evaluation exercise and then will continue presenting the three competitions that have been conducted by now (Senseval-1 (Section 4.3), Senseval-2 (Section 4.4) and Senseval-3 (Section 4.5)). Regardless of the fact that only the last - Senseval-3 competition is relevant for our approach, we consider it important to briefly discuss the earlier ones, hence they provide the base for all later competitions and respectively the base for Senseval-3 as well. However, in case more comprehensible information is needed, refer to (Palmer et al., 2007). Nevertheless, before starting with Senseval we will look at the general issues in evaluation of WSD systems.

4.1 Fundamentals in Evaluation of WSD Systems

Types of evaluation  WSD systems are evaluated in two ways - *in vitro* and *in vivo* (Ide and Véronis, 1998a).

*In vitro* - is the type of evaluation in which the WSD task is assigned independent of any

1 http://www.senseval.org/
specific application. The evaluation is represented in terms of accuracy using a benchmark particularly constructed for the purpose.

*In vivo* - is the type of evaluation in which the WSD system are incorporated into some larger application system, such as Information Retrieval, Machine Translation, Information Extraction or Question Answering. The systems are then evaluated according to their relative contribution to the overall system performance.

Nowadays the first of those two types of evaluation, namely the *in vitro* is more often used than the *in vivo* type.

**Scoring** As (Palmer et al., 2007) report, the scoring of systems on a given example is based on the exact-match criterion. This means that if the answer of the system exactly matches the predefined correct sense label a score of 1 is assigned. If the answer does not match the given score is 0. In cases where the system provides multiple weighted answers to a single example \( w \), the score that the system acquires is computed as the probability that it assigns the correct sense tag \( c \) given the example and its context:

\[
Score = \Pr(c|w, context(w))
\]

If an example has multiple correct answers, the score that the system acquires is the sum of all probabilities that it assigns to any of the correct answers:

\[
Score = \sum_{t=1}^{C} \Pr(c_t|w, context(w))
\]

where \( t \) ranges over the \( C \) correct tags.

In cases of hierarchical sense inventory (as most of the nowadays dictionaries are) three different levels of scoring are used - *fine-grained*, *coarse-grained* and *mixed-grain*:

- **fine-grained** - only exact matching answers count as correct.

- **coarse-grained** - both system answers and predefined correct answers are matched to the top-level sense of the sense-hierarchy of the targeted word. If they match the answer is counted as correct.

- **mixed-grain** - as correct is counted the answer that is matching any of the ”parent” senses or ”children” senses in the hierarchy of the correct sense.

If there is the chance that the system does not give an answer for all examples few measures are used in order to capture the total system performance. *Coverage* of the system amounts to the percentage of instances from the test set for which the systems gives an answer. Another
measure is precision. It is the ratio of the correct answers given by the system to the number of all given answers:

\[
\text{precision} = \frac{|\text{correct answer} \cap \text{given answer}|}{|\text{given answer}|}
\]

(4.3)

Recall is computed by the ratio of the correct answers given by the system to the number of all correct answers.

\[
\text{recall} = \frac{|\text{correct answer} \cap \text{given answer}|}{|\text{correct answer}|}
\]

(4.4)

In order to capture precision and recall together, their harmonic mean, also called F-score (which represents the total system performance) is used:

\[
F = \frac{2 \cdot (\text{precision} \cdot \text{recall})}{\text{precision} + \text{recall}}
\]

(4.5)

4.2 International Evaluation Exercise Senseval

The main purpose of the open, community-based evaluation exercise - Senseval - is to assess the existing approaches for word sense disambiguation and to coordinate and convey different evaluation competitions and accompanying activities. Its aim is to measure the competence of the WSD systems with regard to various target words, various languages and various aspects of language and as well to advance the comprehension of lexical semantics and polysemy.

4.3 Senseval-1

The first of the evaluation competitions of Senseval, Senseval-1 (Kilgarriff, 1998; Kilgarriff and Palmer, 2000; Kilgarriff and Rosenzweig, 2000) was conducted in 1998 with an associated workshop held at Herstmonceux Castle, Sussex, England. As Palmer et al. (2007) report, it has been designed only for English although a parallel task (Romanseval (Segond, 2000; Calzolari and Corazzari, 2000)) has been run for Italian and French. The Hector lexicon (see Section 2.3.2 or (Atkins, 1993)) was used as sense inventory for the task as thirty-four randomly selected words under particular criteria (POS - noun, verb, adjective; number of senses, frequency) were targeted. The data was manually annotated by professional lexicographers with a quite solid IAA of about 80%.

For Senseval-1 the lexical sample task was chosen against the all-words task. In the all-words task the competing systems would need to target all the words in the set of data (i.e. all open-class words in the corpora) whereas in the lexical sample task a single sample of the words is

\(^2\)http://www.itri.brighton.ac.uk/events/senseval/ARCHIVE

\(^3\)http://aune.lpl.univ-aix.fr/projects/romanseval/
selected and corpus examples for the targeted words are extracted. Kilgarriff and Rosenzweig (2000) reason the choice of the lexical sample task for the competition with several arguments: first the fact that more efficient human tagging can be achieved for it; second, the unavailability of a free full dictionary (necessary for the all-words task); third, the impossibility to incorporate all-words task in some of the systems (the need for either sense tagged data or manual input for each dictionary entry).

In total there were 8,448 training instances distributed among 15 nouns, 13 verbs, 8 adjectives and 5 indeterminates. An overview of the figures can be seen in Table 4.3.

<table>
<thead>
<tr>
<th>Senseval -1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date</td>
</tr>
<tr>
<td>Lexical Sample Task</td>
</tr>
<tr>
<td>Corpus</td>
</tr>
<tr>
<td>Nouns</td>
</tr>
<tr>
<td>Verbs</td>
</tr>
<tr>
<td>Adjectives</td>
</tr>
<tr>
<td>Indeterminates</td>
</tr>
<tr>
<td>Training Instances</td>
</tr>
<tr>
<td>Participating Systems</td>
</tr>
</tbody>
</table>

Table 4.1: Summary of the Senseval-1 evaluation task.

4.4 Senseval-2

The results from Senseval-1 proved that an evaluation exercise of this kind is extremely helpful in training WSD systems. Thus, the decision to continue the open, community-based competition with a new task, namely Senseval-2\(^4\) was realized in 2001 and was concluded by the SENSEVAL-2: Second International Workshop on Evaluating Word Sense Disambiguation Systems, held in Toulouse, France and, as well, the workshop Word Sense Disambiguation: Recent Successes and Future Directions held in 2002, Philadelphia, USA. In this edition, it was not only the lexical sample task that the systems could participate in but the all-words task was included as well. For the sake of clarity we will look at those tasks separately further on but since our interest for the whole Senseval-2 competition is more from historical reasons we will not present any result from the tasks. Such can be found directly on the web page\(^5\) provided by Senseval.

**The All-Words Task** was developed for four languages: Czech, Dutch, English and Estonian and it is specific with the fact that the systems have to disambiguate all different ambiguous words in the text.

---

\(^4\)http://www.sle.sharp.co.uk/senseval2
\(^5\)http://193.133.140.102/senseval2/Results/all_graphs.htm
CHAPTER 4. EVALUATION OF WSD SYSTEMS

The Lexical Sample Task in Senseval-2 was conducted for several languages: Basque, Chinese, Danish, English, Italian, Japanese, Korean, Spanish and Swedish. Again as in Senseval-1 only a sample of the words in the corpora was targeted for disambiguation.

<table>
<thead>
<tr>
<th></th>
<th>Senseval-2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date</td>
<td>1998</td>
</tr>
<tr>
<td>All-words</td>
<td>Czech, Dutch, English, Estonian; Basque, Chinese, Danish, English, Italian, Japanese, Korean, Spanish, Swedish;</td>
</tr>
<tr>
<td>Lexical Sample Task</td>
<td>94</td>
</tr>
</tbody>
</table>

Table 4.2: Summary of the Senseval-2 evaluation task.

4.5 Senseval-3

Senseval-3 (Mihalcea and Edmonds, 2004) is the most recent competition in the Senseval evaluation exercise that took place in 2004 and culminated in a workshop held in Barcelona, Spain, as part of ACL 2004 (The 42nd Annual Meeting of the Association for Computational Linguistics). In order to illustrate the power of corpus-based systems in Section 2.2 and Section 2.3 we already showed the most recent results (Table 2.2 on page 17 and Table 2.3 on page 19) of the competing systems for WSD in the Senseval-3 English lexical sample task. This task is most relevant for our experiments because of the fact that we used it as part of the source corpus for them. This is one of the reasons why we describe the task more detailed than the previous ones and especially why we pay the most attention to the English lexical sample part of it.

One of our intentions with describing Senseval-1 and Senseval-2 was to show the gradual extension of the scope of the enterprise. Respectively, Senseval-3 continued this tendency and resulted in 16 individual tasks with 55 teams to participate and 160 resulting systems.

4.5.1 The All-Words Task

The all-words task in Senseval-3 was conducted only for two languages - English and Italian from which only the former has taken part in Senseval-2.

The English all-words task (Snyder and Palmer, 2004; Villarejo et al., 2004) was approached by 47 teams that targeted approximately 5000 words in corpora extracted from the Penn Treebank (two articles from the Wall Street Journal and one from the Brown Corpus) and sense inventory from WordNet. The subject of annotation were predicates together with the headwords of their arguments, adjectives and adverbs. As we have already mentioned before, supervised systems generally perform better than unsupervised ones, which was as well the case in the task as Snyder and Palmer (2004) report. The authors discuss also the fact that the best performing systems already reached accuracy of 65% - 70% on the task where the IAA is between 70% and
75%. Even for Senseval-2, the best performing system (from Southern Methodist University) reported an accuracy of 69% with an IAA of 80%. Since one cannot expect that the systems’ accuracies get better than the IAAs, we can conclude that the results achieved for this task are good and therefore already very close to the upper bound.

The Italian all-words task (Ulivieri et al., 2004) had considerably fewer teams to compete against each other, i.e. seven. The corpus for the task was from the Italian Syntactic Semantic Treebank (Montemagni et al., 2003) (with ItalWordNet as lexicon) and amounted again approximately 5000 words (mainly from newspaper articles about politics, sports, news). Targeted were nouns, verbs, adjectives and a small set of proper nouns.

### 4.5.2 The Lexical Sample Task

The *Lexical sample task* for Senseval-3 was developed for several languages: Basque (Agirre et al., 2004), Catalan (Màrquez et al., 2004a), Chinese (Niu et al., 2004b), English (Mihalcea et al., 2004a), Italian (Magnini et al., 2004), Romanian (Mihalcea et al., 2004b) and Spanish (Màrquez et al., 2004b). A Swedish task was planned but unfortunately cancelled later on. Since the systems across the different languages do not differ greatly in their nature we will describe only one of them, which is most relevant to our work, namely the English lexical sample task.

The English lexical sample task was the subject of interest for 27 teams with 47 resulting systems (e.g. Escudero et al., 2004; Decadt et al., 2004; Buscaldi et al., 2004; Cabezas et al., 2004; Carpuat et al., 2004; Ciaramita and Johnson, 2004; García-Vega et al., 2004; Grozea, 2004; Lamjiri et al., 2004) and many others). From the 60 planned words in the beginning only a total of 57 took part in the final disambiguation task (20 nouns, 32 verbs and 5 adjectives). Table 4.3 provides a more detailed insight of the sense inventory of the words in the English lexical sample task.

<table>
<thead>
<tr>
<th>Class</th>
<th>Nr of words</th>
<th>Avg senses (fine)</th>
<th>Avg senses (coarse)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nouns</td>
<td>20</td>
<td>5.8</td>
<td>4.35</td>
</tr>
<tr>
<td>Verbs</td>
<td>32</td>
<td>6.31</td>
<td>4.59</td>
</tr>
<tr>
<td>Adjectives</td>
<td>5</td>
<td>10.2</td>
<td>9.8</td>
</tr>
<tr>
<td>Total</td>
<td>57</td>
<td>6.47</td>
<td>4.96</td>
</tr>
</tbody>
</table>

Table 4.3: Summary of the sense inventory of the words present in the English lexical sample task (Mihalcea et al., 2004a).

For the collection of the corpus many different sources were used: BNC, Penn Treebank, Los Angeles Times and Open Mind Common Sense, whereas only the first was used in the final version of the data. In this particular task, the normal procedure of manual annotation was solved by the usage of the OMWE interface (Chklovski and Mihalcea, 2002), which was a very interesting and unconventional step. The main idea was that the data was not labelled by pro-
fessional lexicographers but rather by contributors on a voluntary basis. The latter could use a Web-based application which enabled easy and fast annotation of the examples. Since we used exactly this corpus as a grounding base for our data, we consider it very relevant to review the most important parts of the annotation process.

For each of the 57 words a set of sentences with their occurrences was extracted from the corpus and presented to the users of the Web-application. For each of the instances, the contributors had the chance to select one or more senses from a predefined list of all senses that the targeted word has. Of course, it was more than preferable that only one sense is chosen, however, the possibility to select more than one sense together with the options unclear and none of the above was provided. Figure 4.1 on page 41 shows an example of the Web-application.

![Example of the Web-application](image)

Figure 4.1: Screenshot from Open Mind Word Expert (Chklovski and Mihalcea, 2002).

As sense inventory a mixture of WordNet (for all the nouns and the adjectives) and Wordsmyth\(^6\) (for the verbs) was used. We already noticed in the previous sections that the MFS classifier is usually used as a baseline for system performance and for the English lexical sample task this heuristic reached 64.5% (coarse) and 55.2% (fine) accuracy. The reported overall IAA (normally used as upper bound for system performance) for the data is relatively low - approximately

\(^6\)http://www.wordsmyth.net/
67.3%. From this result we could conclude that the expected performance of the systems in that
task was in the range between the MFS and the inter-annotator agreement (IAA). However, the
best performing system was reported to have reached an accuracy of 79.3% (coarse) and 72.9%
(fine). Those considerably unexpected results could either mean that the IAA achieved by the
OMWE is not good enough to be used as an upper bound for the task or that the systems reach
considerably better performance than humans do for the given data.

By briefly describing the Senseval-3 lexical sample task and in particular the English lexical
sample task we aimed to give a broad overview of the attempts and achievements in the en-
terprise, hence this is a starting point of understanding the system that we designed, which is
fairly similar to the systems we have presented for the task. In Table 4.4 we have summed up
the most important information from the presented section. To those figures we will return later
on in our work when we will look at the structure and results of our approach.

<table>
<thead>
<tr>
<th>Language</th>
<th>IAA</th>
<th>Best Score (coarse)</th>
<th>System</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basque</td>
<td>78.2%</td>
<td>71.1%</td>
<td>Swarthmore College</td>
</tr>
<tr>
<td>Catalan</td>
<td>93.16%</td>
<td>85.2%</td>
<td>IRST</td>
</tr>
<tr>
<td>Chinese</td>
<td>-</td>
<td>60.40%</td>
<td>I²R-WSD (FEATUREA1)</td>
</tr>
<tr>
<td>English</td>
<td>67.3%</td>
<td>79.3%</td>
<td>htsa3</td>
</tr>
<tr>
<td>Italian</td>
<td>73% - 99%</td>
<td>53.1%</td>
<td>IRST-Kernels</td>
</tr>
<tr>
<td>Romanian</td>
<td>41.72% - 94.43%</td>
<td>77.1%</td>
<td>romanian-swat hk-bo</td>
</tr>
<tr>
<td>Spanish</td>
<td>83% - 90%</td>
<td>84.20%</td>
<td>IRST</td>
</tr>
</tbody>
</table>

Table 4.4: Summary of the Senseval-3 Lexical Sample Task.

4.5.3 Other Tasks

Along with the all-words task and the lexical sample task there were a few other competitions
in Senseval-3 (Automatic subcategorization acquisition (Preiss and Korhonen, 2004), Multili-
ingual lexical sample/Translation (Chklovski et al., 2004), Word-Sense Disambiguation of WordNet
Glosses (Litkowski, 2004b), Automatic Labeling of Semantic Roles (Litkowski, 2004a), Identifi-
cation of Logic Forms in English (Rus, 2004), Identification of Semantic Roles in Swedish - can-
celled). However, those tasks are not that relevant for our work and for the sake shortness we
will not pay further attention to them. Thus, if any detailed information is needed, refer to the
relevant literature for each task.

4.6 Semeval-1

Since the results from the Senseval competitions were so good and there was a growing in-
terest in the task of classifying semantic relations between pairs of words a new exercise was
designed as a follower of Senseval - Semeval. The change of the name of the task was the result of the attempt to extend its spectrum to all aspects of computational semantic analysis of language. Consequently, the scope of the first task of Semeval - Semeval-1 (Agirre et al., 2007) has been constrained to a particular application of semantic relation classification, relational search. Semeval-1 included 18 different tasks targeting the evaluation of systems for the semantic analysis of text. The approached relations were between nominals (e.g. nouns and base noun phrases, excluding named entities). The used data consisted of manually annotated sentences. Semeval-1 took place in 2007, followed by a workshop held in conjunction with ACL in Prague - (Agirre et al., 2007).

4.7 Summary

After we have discussed the problem of comparability of word sense disambiguation systems in Chapter 3, in the current chapter we gave an overview of the ways in which those systems could be uniformly evaluated and as well the variety and coverage of the developed approaches.

The evaluation exercises conducted in the Senseval enterprise definitely show that the recent WSD systems are able to achieve considerably good accuracy levels that for some tasks become even comparable with human performance. The rising with each Senseval task divergence proved as well that the system performance remains as well relatively consistent over a variety of word types, frequencies and sense distributions.

As Palmer et al. (2007) discuss there are still many open problems connected with the evaluation of WSD systems. One such problem is the choice of sense inventory, which we already saw in the results above lead to noticeably inconsistent performance of humans and automatic systems. A very important open question is the impact of more training data for high polysemy verbs.
Chapter 5

TiMBL: Tilburg Memory-Based Learner

In Section 2.3.5 of our work we reviewed the supervised methods for word sense disambiguation and payed most attention to the similarity-based methods also called memory-based methods. Since we have chosen to use a method of this family we needed as well a software that can facilitate the usage of such methods. Our choice is the Tilburg Memory-Based Learner (TiMBL).

5.1 Overview

TiMBL\(^1\) was the outcome of combining ideas from various different MBL approaches. It is a fast and discrete decision-tree-based implementation of the \(k\)-nearest neighbor classification algorithm described in more detail in (Daelemans et al., 2007). As a result it has become one of the very important and useful natural language processing tools for multiple alternative domains. This comes from the fact that TiMBL, following the principles of machine-based learning is created around the belief that intelligent behavior could be achieved by analogical reasoning and not by the use of abstract mental rules. This is how the computation of behavior from already seen representations of earlier experience to new situations, based on the similarity of the old and the new situation, is of a great significance. The latter describes the heart of TiMBL - the possibility to learn classification tasks from already seen examples.

The software was jointly developed by the Induction of Linguistic Knowledge (ILK) Research Group of the Department of Communication and Information Sciences at the Tilburg University, The Netherlands and the CNTS - Language Technology Group by the Department of Linguistics at the University of Antwerp, Belgium. The complete C++ source code was released under the GNU General Public License\(^2\) (GNU GPL) as published by the Free Software Foundation\(^3\). Its

---

\(^1\)http://ilk.uvt.nl/timbl/
\(^2\)http://www.gnu.org/copyleft/gpl.html
\(^3\)http://www.fsf.org/
installation is considerably straightforward on the majority of UNIX-based systems.

Originally TiMBL was designed to be the solution for the linguistic classification task, however, it can be exploited for any alternative categorization task with appropriate (symbolic or numeric) features and discrete (non-continuous) classes for which training data is available. The latter again leads us to the already discussed topic on the acute shortage of labeled data.

5.2 Application

As we have mentioned above the training data for WSD, namely the data that TiMBL uses in the process of learning is represented by feature vectors of which the exact structure is shown in Section 2.3.4. The format of the feature files is flexible, since TiMBL is able to guess the type of format in most of the cases. However, we will stick to the most often used format - feature vectors are features separated by spaces on a single line.

As an example let us consider the situation in which we have a training set (delivered to TiMBL as the file data.train) and a test set (data.test). After running the tool as follows:

> Timbl -f data.train -t data.test

TiMBL returns a new file data.test.IB1.O.gr.k1.out, which consists basically of the data in our test file data.test. However, the system adds a new feature to each FV, which represents the new class that it has predicted for the vector. The experiment is conducted with the default parameters for the systems and the results are sent to standard output (or if needed are written in a separate data file). For a more detailed information on the format and information of the results, refer to (Daelemans et al., 2007).

The name of the output file data.test.IB1.O.gr.k1.out consists of the most important information for the conducted experiment. The first two parts represent the name of the test file that was used for the analysis (data.test) and together with .out it is referred to the output file of the experiment; IB1 represents the machine-based learning algorithm that was employed - the k-NN algorithm in this particular case; O stands for the similarity computed with weighted overlap; gr means that the relevance weights were computed with gain ratio and finally k1 represents the number of most similar patterns in the memory on which the output label was based. If those default settings are the ones one needs for the planned experiment, there is not much more to do. However, when we talked about supervised WSD methods we mentioned multiple algorithms that could be employed for the purpose and TiMBL supports a good selection of them. Another extremely wide range of possibilities is connected with the distance metrics that can be used with TiMBL in order to determine the similarity between the different instances. All different options can be specified directly on the command line before running an experiment with TiMBL.

> Timbl -k 3 -f data.train -t data.test
This command for instance will run the latter experiment. However this time a different number of nearest neighbors will be used for extrapolation. Normally the default value is 1, thus if anything else is needed it must be specified explicitly.

A very important for us option, which we use further in our work is the \texttt{+v n} (verbosity) option. It allows us to output the nearest neighbors on which decisions are based.

Daelemans et al. (1999) comprehensively describe all options and their possible value ranges that can be chosen.
Chapter 6

Automatic Extraction of Examples for WSD

The following Chapter is devoted particularly to our work and the description of the designed by us system. We will describe stepwise the process of word sense disambiguation that we employ starting with a short overview of the system (Section 6.1) and the data collection that we use - Section 6.2. A short discussion of how the data collection is preprocessed can be found in Section 6.3, followed by its division into training and test set in Section 6.4. We will look at the specific WSD algorithm that we selected (Section 6.5) as well as the parameter optimization which we considered (Section 6.6). In order to make it completely transparent how we acquired the results reported in Section 6.8 we will first present (Section 6.7) the scoring software that was provided by Senseval-3, which we employed for the scoring of our system. The final section of the chapter (Section 6.9) is an evaluation of the presented system.

6.1 Overview of the System

The idea of how our system is constructed is pretty straightforward. As a semi-supervised one it is basically put together similarly to the one visualized in Figure 2.1 on page 20. The difference between a supervised WSD system and ours is delineated by the automatically labeled data that we add to our final collection of examples. It characterizes our approach as a semi-supervised WSD. This distinctness is depicted in Figure 6.1 on page 48 in the Data Collection component. We chose to exhibit this figure here, since it represents to a great extent the overview of our system. Later on in the section we will discuss in more detail each of its separate parts in order to give a deeper insight of our endeavors. How those several components function together is represented in Figure 6.1 on page 48.

A very important part of the experimental process is the collection of data. We used the manually annotated data, but since our aim was to extend it we gathered examples from several online dictionaries and various corpora. The latter collection was to be automatically annotated.
and together with the manually annotated one the final data collection was constructed (to be seen in the upper left corner of Figure 6.1). The next step was to preprocess the data and turn it into a training and test set needed for the training and testing procedures. For training, a specific WSD algorithm had to be selected together with the parameters that best fit the given task. After the word-expert is trained it is tested on the prepared test set and an evaluation of the results is attempted. At that point, if the evaluation is satisfying, the final word-expert/classifier is accepted. In case the results need to be improved a new start of the process at any of its states can be attempted.

![Figure 6.1: Our semi-supervised WSD system.](image)

### 6.2 Data Collection

Here we will describe in more detail the data we had collected for our experiments. We will pay attention mainly to the sources we used to gather the data, to the sense inventory according to which the senses were chosen and to the actual annotation procedure.

#### 6.2.1 Sense Inventory

The English lexical sample task (see Section 4.5.2), which served as the source for our manually annotated data used WordNet and Wordsmyth as sense inventories. The former was concerned with the possible senses for the nouns and the adjectives in the examples and the latter with
CHAPTER 6. AUTOMATIC EXTRACTION OF EXAMPLES FOR WSD

those for the verbs. A proper mapping between the two is provided from the English lexical sample task in Senseval-3. We kept this setup as it is. However for the new, automatically extracted unlabeled data we used only WordNet for all words.

6.2.2 Source Corpora

Our final data collection as can be seen in Figure 6.1 on page 48 consists of manually annotated data and unlabeled data, which we automatically annotate. In the following section we will discuss in more detail both of those sets.

The Manually Annotated Data was gathered from the Senseval-3 English lexical sample task which we have already discussed in Section 4.5.2.

At that point we know that the data is available from the Senseval-3 competition but we still have not looked at its format. Similar to the one used in Senseval-2 the data is marked up in the Extensible Markup Language\(^1\) (XML) format. However, as the authors remark, it can’t be parsed as valid XML could be, since the characters that need special handling in XML are not escaped. Nevertheless, an attempt to fix this disadvantage for future competitions has already been started. Let us consider a simple example of how an instance is constructed in the Senseval lexical sample task format, which can be seen in (3) below:

\[(3)\]

\[
<\text{instance id}="\text{activate.v.bnc.00024693}" \text{docs}rsrc="\text{BNC}"
<\text{answer instance}="\text{activate.v.bnc.00024693}" \text{sense}id="38201"/>
<\text{context}>
\text{Do you know what it is , and where I can get one ? We suspect you had seen the Terrex Autospade , which is made by Wolf Tools . It is quite a hefty spade , with bicycle \text{-} type handlebars and a sprung lever at the rear , which you step on to \text{activate} it . Used correctly , you should n’t have to bend your back during general digging , although it wo n’t lift out the soil and put in a barrow if you need to move it ! If gardening tends to give you backache , remember to take plenty of rest periods during the day , and never try to lift more than you can easily cope with .}
</\text{context}>
</\text{instance}>

First, each of the separate examples is wrapped in an \(<\text{instance}>\) tag which carries important information - it has an identification attribute (id) the value of which uniquely identifies

\(^1\)http://www.w3.org/XML/
each instance. The value (in our case activate.v.bnc.00024693) is build up of four different parts separated by dots - the target word itself, its POS tag, the source corpus and a unique number. There is as well a second attribute (docsrsr) that shows from which corpus this particular example was extracted (again in this case - BNC).

Second, very important for the training set and completely excluded from the test one, an <answer> tag is added. It, too, has two attributes: instance (used to link the answer to a particular example) and senseid (used to give the correct identification number of the answer according to the Senseval standard). If it is the case that more than one correct answer per example can be assigned, each of them is specified in a separate tag e.g.:

<answer instance="activate.v.bnc.00044852" senseid="38201"/>
<answer instance="activate.v.bnc.00044852" senseid="38202"/>

Third, a special tag for marking the whole example is used - <context>.

The last and probably most important part of an example instance is the <head> tag. It is used to mark the target word - activate - in its context. It is interesting to note that this tag can be used more than once per example. If this happens more than one FV can be extracted per example.

In the final collection each of the sets of examples for the separate words is separated by the <lexelt> tag holding a single attribute (item). Its value is the targeted word and its POS tag separated by a dot.

**The Unlabeled Data** was collected from several different sources - online dictionaries, WordNet and various corpora. The extraction of such diverse examples provided us as well with the chance to analyze better their quantity and quality and therefore contributed greatly to our evaluation of automatically extracted examples for word sense disambiguation.

When we discuss the lexicons of our choice it is important to note that because of the fact that normally dictionaries are not complete in respect to the information they contain about a given word, we used more than one source with the expectation that a larger coverage of distinct senses can be achieved.

*Dictionary.com Unabridged (v 1.1)* is one of the dictionaries that we used to which an online interface is provided by Dictionary.com² (a multi-source dictionary search service produced by Dictionary.com, LLC³). The latter provides access as well to multiple different dictionaries, however, we made use of just one of them (for a full list of the provided dictionaries, refer to the following page: http://dictionary.reference.com/help/about.html).

The online version of *Cambridge Advanced Learner's Dictionary⁴* is another source we have utilized. The Cambridge dictionary, as others of its kind, is created with the use of the Cam-
bridge International Corpus which consists of approximately 1 billion words of relatively modern spoken and written English from around the world.

As source for examples that we also employed the American Heritage Dictionary of the English Language which is accessible online under (http://education.yahoo.com/reference/dictionary/).

Another lexicon from which we extracted examples is The Longman Dictionary of Contemporary English Online\(^5\). It is basically an online version of the CD-ROM of the Longman Dictionary of Contemporary English, Updated Edition.

Although it is not exactly a dictionary WordNet 3.0\(^6\) was as well an important lexicon for our system, since it did not serve us only as a source for examples. Its further usages though will be discussed later on.

### Lexicons

<table>
<thead>
<tr>
<th>Sources</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dictionary.com Unabridged (v 1.1)</td>
<td>114</td>
</tr>
<tr>
<td>Cambridge Advanced Learner’s Dictionary</td>
<td>405</td>
</tr>
<tr>
<td>American Heritage Dictionary of the English Language</td>
<td>194</td>
</tr>
<tr>
<td>The Longman Dictionary of Contemporary English Online</td>
<td>709</td>
</tr>
<tr>
<td>WordNet 3.0</td>
<td>300</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>1 722</strong></td>
</tr>
</tbody>
</table>

Table 6.1: The collection and size of lexicon examples used in our system.

Discussing the origin of the examples does not give us a complete idea of their kind. To make that part more clear we will make a short illustration. Let us again take the word *activate* and Cambridge Advanced Learner’s Dictionary as an example. The search from the online interface of the dictionary returns an entry a screenshot of which is to be found in Figure 6.2.

![activate screenshot](http://www.ldoceonline.com/)

Figure 6.2: A screenshot of the result from the online interface of the Cambridge Advanced Learner’s Dictionary when searched for the word *activate*.

The fact that we are interested in the target word as a verb leaves us with the first part of the entry which displays two separate senses of the verb. Only one, however, gives an illustration of the usage of the word in that particular sense - the first one. In consideration of the fact

---

\(^5\) http://www.ldoceonline.com/
\(^6\) http://wordnetweb.princeton.edu/perl/webwn
that such archetypes are usually very typical and often met usages of the word we select exactly them as instances for our automatically extracted corpus of examples. Even though that at the time when we obtain the example we look at particular sense of the word (in this case to cause something to start), we do not further keep track of this information. This might seem to be counter intuitive, as exactly this knowledge is what our system so "actively" seeks to discover. Though, as a result of the fact that we use more than one dictionary for this purpose and that we finally aim to use a uniform sense inventory, keeping this knowledge will lead to the need of mapping between the used lexicons. Such mapping does not exist by now for all of the lexicons of our choice and there is a long discussion concerning all the problems that come with it. In the best case there will be a corresponding definition of each sense in a dictionary to each sense in another lexicon (known as one-to-one mapping). Unfortunately, most often are the cases where there is a many-to-one, one-to-many or even one-to-zero mapping. This means that either many of the senses can map to a single sense in a different dictionary, one can map to many of them, or even one cannot be mapped to any sense in another dictionary. This does not necessarily imply that the dictionaries are of a bad quality - they were just made with different intentions and for different audiences. Ide and Véronis (1990) discuss this issue in more detail and even propose a solution to the problem. At that point, however, since we do not keep track of the provided sense information and thus are not influenced by the described problem, we will continue our discussion by presenting the corpora sources which we used for extracting additional examples for our data set.

Additionally, we have used two different corpora for building up our sources and those are the BNC (see also Section 2.3.2) and the ukWaC. The British National Corpus is a 100 million word accumulation of both spoken and written language. It has been gathered from a big variety of sources and was designed to give a good representation of current British English, both spoken and written. The BNC corpus provided us with a total of 15 472 examples, which we included in our data collection. The second source we used was the ukWaC corpus (Ferraresi et al., 2008). The corpus was automatically constructed by Adriano Ferraresi by crawling a part of the English Internet domain (more precisely - the .uk domain). As Ferraresi et al. (2008) mention, the corpus consists of more than 2 billion tokens and thus it is one of the largest freely available linguistic resources for English. Form the ukWaC corpus we gathered 158 072 examples. In Table 6.2 we give a summary of the examples that we collect from the two different corpora.

<table>
<thead>
<tr>
<th>Corpora</th>
<th>Sources</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>BNC</td>
<td>15 472</td>
<td></td>
</tr>
<tr>
<td>ukWaC</td>
<td>158 072</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>173 544</td>
<td></td>
</tr>
</tbody>
</table>

Table 6.2: The collection and size of corpora examples used in our system.
6.2.3 Automatic Annotation

Having automatically extracted the unlabeled examples in the form of possible occurrences of the targeted word we still have few steps to go through until we are ready for the actual disambiguation process. One such step is the automatic annotation of the data. In that step we do not only need to care about the fact that each instance acquires a sense tag but as well that the final result has the same predefined format as the already provided manually annotated data.

The automatic annotation in our system is a very interesting process. In order to perform WSD by the use of more examples we first need to annotate those examples which itself implies WSD. This chicken and egg problem, however, was in a way resolved by the fact that we used another method for the automatic annotation of examples: the generalized framework for WSD - WordNet::SenseRelate::TargetWord\(^7\) developed by Ted Pedersen, Siddharth Patwardhan and Satanjeev Banerjee - (Patwardhan et al., 2005).

WordNet::SenseRelate::TargetWord employs a knowledge-based method (see Section 2.1), since it uses an existing semantic network (in this case WordNet) to measure the semantic similarity of words, on the basis of which word sense disambiguation is accomplished.

WordNet::SenseRelate::TargetWord is a freely available Perl\(^8\) package that performs WSD. It is distributed under the GNU GPL. The package disambiguates a target word in a given context. The process of disambiguation is based on the computation of a measure of semantic similarity and relatedness based on WordNet and is further implemented in the package WordNet::Similarity\(^9\). This measure is applied to the target word and its neighbors. As a result the most similar sense is given. One of the reasons to use this framework was because it uses WordNet as sense inventory, which means that the senses given as the result of the disambiguation is one of the WordNet senses of the targeted word. This is exactly what we require in order to get a data set most similar to the already manually annotated one from Senseval-3.

To get a better idea of how this exactly works, let us follow again an example. Recall the entry that the Cambridge Advanced Learners Dictionary returned when we searched for the verb activate - *The alarm is activated by the lightest pressure*. (see again Figure 6.2 on page 51). If we use WordNet::SenseRelate::TargetWord to get the WordNet sense of activate in this sentence the module gives us as an answer activate\#v\#1. This means that activate as a verb is used in this sentence with its first sense in WordNet. Now, let’s have a look at the WordNet entry of the verb activate that can be seen in Figure 6.3 on page 54.

---

\(^7\)http://search.cpan.org/~sid/WordNet-SenseRelate-TargetWord-0.09/lib/WordNet/SenseRelate/TargetWord.pm

\(^8\)http://www.perl.org/about.html

\(^9\)http://search.cpan.org/dist/WordNet-Similarity/lib/WordNet/Similarity.pm
CHAPTER 6. AUTOMATIC EXTRACTION OF EXAMPLES FOR WSD

Verb

- **S: (v) trip#4, actuate#1, trigger#1, activate#1, set off#1, spark off#1, spark#1, trigger off#1, touch off#1** (put in motion or move to act) "trigger a reaction", "actuate the circuits"
- **S: (v) activate#2** (make active or more active) "activate an old file"
- **S: (v) activate#3** (make more adsorptive) "activate a metal"
- **S: (v) activate#4, aerate#2** (aerate (sewage) so as to favor the growth of organisms that decompose organic matter)
- **S: (v) activate#5** (make (substances) radioactive)

Figure 6.3: A screenshot of the result from the online interface of WordNet when searched for the word *activate*.

What we find under the first sense is the definition *put in motion or move to act*, which means that *WordNet::SenseRelate::TargetWord* provided us with the correct answer. So now we have the example and the WordNet sense of the targeted word but still our instance does not look like the one we saw in (3) on page 49. Thus, there is yet a small step we need to do to get our example ready - marking it up in the XML format that Senseval-3 uses. In (4) the final instance is shown.

(4)

```xml
<instance id="activate.v.cald.12345678" docsrc="CALD">
<answer instance="activate.v.cald.12345678" senseid="38201"/>
<context>
The alarm is <head>activated</head> by the lightest pressure.
</context>
</instance>
```

An observant eye will note that the *senseid* in (4) and the one (*activate#v#1*) we got from the Perl module *WordNet::SenseRelate::TargetWord* are not the same. This is because of the fact that Senseval-3 provides the sense inventory for the words in a separate XML formatted file. The entry for activate in this file looks like shown in (5). Thus we included in our system a mapping between the senses that we get from *WordNet::SenseRelate::TargetWord* and the ones that are in the sense inventory file provided by Senseval-3.

(5)

```xml
<lexelt item="activate.v">
<sense id="38201" source="ws" synset="activate actuate energize start stimulate" gloss="to initiate action in; make active."/>
```
Patwardhan et al. (2007) report that WordNet::SenseRelate::TargetWord achieves 54% of accuracy. Since there is no better way to check if the tendency in correctness of the annotations will remain the same for our examples, we picked up a small part of the instances and annotated them manually. We used the examples gathered from the online dictionaries for this purpose and extracted 1000 random instances from them. The manual annotation gave us an upper bound for the performance of our system (or the so called gold standard) which was achieved with an IAA of about 65.5%. WordNet::SenseRelate::TargetWord labeled correctly 647 out of the 1000 instances which results to an accuracy of 64.7% on our data.

Now, we have our completed, automatically annotated examples and together with the manually annotated ones from Senseval-3 English Lexical Sample we build our final data collection as was shown in Figure 6.1 on page 48. On the latter one can also see that the next step we undertake in our system is the needed preprocessing of the data before it is turned into training and test sets, which TiMBL can use.

### 6.3 Data Preprocessing

Data preprocessing is normally considered extremely important for WSD systems of this kind. In fact it is often the case in which badly preprocessed data has an effect on the final accuracy of the disambiguation system. Since our data set now consists basically of automatically collected and annotated examples the possibility that any sort of noise (undesired data) exists in it is extremely high.

#### 6.3.1 Basic Preprocessing

The basic preprocessing step in our system includes several very fundamental operations that prepare our data to be compatible with the manually annotated one like tokenization (breaking the examples up into their constituent tokens), punctuation handling and special and metacharacters characters handling (which were often found in the automatically collected data). Since we detect our target word in the automatic annotation step, this operation is not further needed here. Another possible manipulation is the use of stoplist (a list of words that are either not
considered as valid tokens or are not desired further in the examples). We however did not make use of a stoplist in our system. So if we go back to our example from (4) the only change that will take place is the separation of the dot at the end of the sentence (6):

(6)

```
<instance id="activate.v.cald.12345678" docsrc="CALD">
<answer instance="activate.v.cald.12345678" senseid="38201"/>
<context>
The alarm is <head>activated</head> by the lightest pressure .
</context>
</instance>
```

6.3.2 Part-of-Speech Tagging

The last step before we can finally extract our feature vectors for the creation of training and test sets is the POS tagging of the examples. For this purpose we used the part-of-speech tagger\(^{10}\) for English developed at the University of Tokyo (Department of Computer Science, Tsujii laboratory) by Yoshimasa Tsuruoka and Jun'ichi Tsujii (Tsuruoka and Tsujii, 2005). One of the reasons to choose exactly this tagger was its high tagging speed and at the same time state-of-art accuracy (2400 tokens/sec with 97.10% accuracy). The tagger is based on a bidirectional inference algorithm for sequence labeling problems (POS tagging, text chunking (shallow parsing), named entity recognition) as described in (Tsuruoka and Tsujii, 2005)

So finally our toy example looks like in (7):

(7)

```
<instance id="activate.v.cald.12345678" docsrc="CALD">
<answer instance="activate.v.cald.12345678" senseid="38201"/>
<context>
The/DT alarm/NN is/VBZ <head>activated/VBN</head> by/IN
the/DT lightest/JJS pressure/NN ./.
</context>
```

Since the Senseval-3 XML markup does not compile as it is, and since this is not of a significant importance for our work, we did not change the output of the tokenizer into a valid XML markup any further and worked with it as it is shown in (7). However, in case such markup is needed, it should be the same as the one used in the Senseval competition and should look like in (8) on page 57.

\(^{10}\)http://www-tsujii.is.s.u-tokyo.ac.jp/~tsuruoka/postagger/
(8)

<instance id="activate.v.cald.12345678" docsrc="CALD">
<answer instance="activate.v.cald.12345678" senseid="38201">
<context>
The <p="DT"/> alarm <p="NN"/> is <p="VBZ"/> <head>activated <p="VBN"/></head> by <p="IN"/> the <p="DT"/> lightest <p="JJS"/> pressure <p="NN"/> . <p="."/>
<context>

So, having our POS annotated examples, there is only one thing to do before we acquire our training and test sets and this is the extraction of the FVs from the prepared instances.

### 6.4 Training and Test Sets

#### 6.4.1 Feature Vectors

The context features that we finally chose to use for our FVs are selected out of the pool of features (see Appendix B) that we considered sensible for the task. Similar to the vectors shown in Table 2.4 on page 24 our feature vectors are build up from the collection of features shown in Table 6.3 on page 58.

There are few things that can be noted here. First, our choice of the set of features ensures that the vectors are descriptive enough of the occurrence of the target word in a given context. This set of selected features lead to good results in Romanian (Dinu and Kübler, 2007) and as well to the training of good and precise word-experts in our system. Another interesting point to be noted are the features like VA and PB in our case - in other words the features that do not get covered by the context of the word. Such features get "zero" or "place holder" values so that their position is kept in the vector but no actual value is stated.

Finally, our FV looks like this: The alarm is activated by the lightest DT NN VBZ VBN IN DT JJS pressure alarm - is by - 38201, and is ready to be included in our training set.
CHAPTER 6. AUTOMATIC EXTRACTION OF EXAMPLES FOR WSD

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
<th>In Our Toy Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>CT_{-3}</td>
<td>token at position -3 from the TW</td>
<td>The</td>
</tr>
<tr>
<td>CT_{-2}</td>
<td>token at position -2 from the TW</td>
<td>alarm</td>
</tr>
<tr>
<td>CT_{-1}</td>
<td>token at position -1 from the TW</td>
<td>is</td>
</tr>
<tr>
<td>CT_{0}</td>
<td>the target word</td>
<td>activated</td>
</tr>
<tr>
<td>CT_{1}</td>
<td>token at position 1 from the TW</td>
<td>by</td>
</tr>
<tr>
<td>CT_{2}</td>
<td>token at position 2 from the TW</td>
<td>the</td>
</tr>
<tr>
<td>CT_{3}</td>
<td>token at position 3 from the TW</td>
<td>lightest</td>
</tr>
<tr>
<td>CP_{-3}</td>
<td>the POS tag of the token at position -3 from the TW</td>
<td>DT</td>
</tr>
<tr>
<td>CP_{-2}</td>
<td>the POS tag of the token at position -2 from the TW</td>
<td>NN</td>
</tr>
<tr>
<td>CP_{-1}</td>
<td>the POS tag of the token at position -1 from the TW</td>
<td>VBZ</td>
</tr>
<tr>
<td>CP_{0}</td>
<td>the POS tag of the target word</td>
<td>VBN</td>
</tr>
<tr>
<td>CP_{1}</td>
<td>the POS tag of the token at position 1 from the TW</td>
<td>IN</td>
</tr>
<tr>
<td>CP_{2}</td>
<td>the POS tag of the token at position 2 from the TW</td>
<td>DT</td>
</tr>
<tr>
<td>CP_{3}</td>
<td>the POS tag of the token at position 3 from the TW</td>
<td>JJJS</td>
</tr>
<tr>
<td>NA</td>
<td>the first noun after the TW</td>
<td>pressure</td>
</tr>
<tr>
<td>NB</td>
<td>the first noun before the TW</td>
<td>alarm</td>
</tr>
<tr>
<td>VA</td>
<td>the first verb after the TW</td>
<td>-</td>
</tr>
<tr>
<td>VB</td>
<td>the first verb before the TW</td>
<td>is</td>
</tr>
<tr>
<td>PA</td>
<td>the first preposition after the TW</td>
<td>by</td>
</tr>
<tr>
<td>PB</td>
<td>the first preposition before the TW</td>
<td>-</td>
</tr>
<tr>
<td><strong>Answer</strong></td>
<td>the answer (only in the training features)</td>
<td>38201</td>
</tr>
</tbody>
</table>

Table 6.3: Features included in the feature vectors of our system and their corresponding values from our toy example.

6.4.2 Training Set

The training set we have is a combination of manually and automatically annotated data from the sources we point out in Section 2.3.2.

The manually annotated part of the corpus is the original Senseval-3 training set\(^\text{11}\). The total number of instances in the training set is 7,860 and a more detailed distribution of the examples can be seen again in Appendix C at the end of the thesis.

6.4.3 Test Set

The test set\(^\text{12}\) that we use is the one provided by Senseval-3 English lexical sample task. It has a total of 3,944 manually annotated examples. A comprehensive description of the size of examples per word is given in Appendix C. We kept the complete size of the test set and the only

\(^{11}\)http://www.cse.unl.edu/~rada/senseval/senseval3/data/EnglishLS/EnglishLS.train.tar.gz

\(^{12}\)http://www.cse.unl.edu/~rada/senseval/senseval3/data/EnglishLS/EnglishLS.test.tar.gz
changes to it (e.g. tokenization, punctuation and metacharacters handling) were made in our
data preprocessing step, so that the extraction of FVs for testing was made possible.

A summary of the final data collection is represented in Table 6.4 below:

<table>
<thead>
<tr>
<th>Annotation</th>
<th>Source</th>
<th>Train</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Automatic</td>
<td>Dictionaries</td>
<td>1722</td>
<td></td>
</tr>
<tr>
<td></td>
<td>BNC</td>
<td>15472</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ukWaC</td>
<td>158072</td>
<td></td>
</tr>
<tr>
<td>Manual</td>
<td>Senseval-3 (ELST)</td>
<td>7860</td>
<td>3944</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td>183126</td>
<td>3944</td>
</tr>
</tbody>
</table>

Table 6.4: The total collection of examples used in our system.

### 6.5 Algorithm Selection

In Section 2.3.5 we gave an overview of the different methods which can be used in a supervised
WSD system. There we also mentioned that in our system we use a similarity-based method.
More precisely our choice for an algorithm as a representative of MBL is the $k$-Nearest Neighbor
($k$NN) (Ng and Lee, 1996). We already noted (see Section 2.3.5) how the algorithms of this family
classify new examples based on the similarity of the given instances. Additionally, we discussed
that the set of the $k$ nearest examples out of all stored instances is extracted by measuring the
distance between the target example and all training examples. What we did not mention by
now is how the latter distance is represented. The calculation of the distance is done according
to a given distance metric which in our case is the overlap metric (a basic distance metric that
works for patterns with symbolic features). If we recall again, the examples are represented as
FVs in our system (see Section 6.4.1). They are composed of several features. The overlap metric
(see Equations 6.1 and 6.2) measures the distance between the vectors $X$ and $Y$ ($\Delta(X, Y)$) by the
sum of the distances between the $n$ features (in our case $n = 20$) used as building parts of the
vectors. $\delta(x_i, y_i)$ is the distance per feature.

$$\Delta(X, Y) = \sum_{i=1}^{n} \delta(x_i, y_i)$$ \hspace{1cm} (6.1)

where:

$$\delta(x_i, y_i) = \begin{cases} 
\text{abs} \left( \frac{x_i - y_i}{\max_i - \min_i} \right) & \text{if numeric, else} \\
0 & \text{if } x_i = y_i \\
1 & \text{if } x_i \neq y_i \end{cases}$$ \hspace{1cm} (6.2)

IB1 (Aha et al., 1991) is the result of employing the $k$NN together with the overlap distance
metric. However, this combination used in TiMBL is a bit different than the originally suggested algorithm by Aha et al. (1991), since it refers to the $k$ nearest distances and not to the $k$ nearest examples. Daelemans et al. (2007) also describe further differences, plus the exact usage of the IB1 algorithm in TiMBL.

### 6.6 Parameter Optimizations

Parameter optimization for machine-learning of word sense disambiguation is considered to be extremely important to the performance of a WSD system (Hoste et al., 2002). The authors argue and even demonstrate that optimization per word-expert results to a significant improvement in the generalization accuracies of the final WSD system. This is why we considered this step as important and approached it in our work as well.

#### 6.6.1 General Parameter Optimizations

One of the disadvantages of parameter optimization is that in big systems like ours, the computational effort in the system rises proportionally to the number of optimized parameters. Because of this fact we will not be able to perform an exhaustive parameter optimization for our system, but rather concentrate on a single one and chose a very limited number of options.

In section 6.5 we focused on the algorithm that we selected for conducting our experiments - the $k$NN. Referring to its description, we can conclude that the most straightforward parameters that can be optimized for this algorithm are the distance metric with which it is used and the values for $k$ representing the distance to the nearest neighbors. Apart from those other possibilities for parameter optimization as feature weighing and class weighing could be considered as well.

Because of the limited time we have we will concentrate mostly on the $k$ parameter focusing on at most three possible values - $k = 1$, $k = 3$, $k = 5$. We intentionally select odd numbers as values for $k$ in order to avoid ties.

What is important to note here is that in the cases where the parameter $k$ is optimized only the best performance of the three options is taken for each of the separate word-experts. For example if the word `note` reaches an accuracy when $k = 1$ - 54.5%, when $k = 3$ - 60% and when $k = 5$ - 64.2%, only the last score will be included as final in the results. Whenever we present results with not optimized parameters only the default value for $k$, which for TiMBL is 1, should be considered.

#### 6.6.2 Automatic Feature Selection

In the process of disambiguation a memory-based system like ours considers each training example in order to give the best possible label for the example that is being disambiguated. Of
course, this is good especially for fully supervised approaches, since then the training data is considerably less and exceptional cases can be represented only with a few instances. A negative feature, however, is the fact that each instance is taken into account with all its features present in the FV. Since memory-based learners are extremely sensitive to irrelevant features this leads to a significant decrease in accuracy whenever such features are present.

For this purpose we use a forward-backward algorithm similar to (Dinu and Kübler, 2007) or (Mihalcea, 2002) which improves the performance of a word-expert by selecting a subset of features from the provided FV for the targeted word. This subset has less but more relevant for the disambiguation process features.

The exact forward (9) and backward (10) algorithms that we used, following Dinu and Kübler (2007) are as follows:

(9)

function ForwardAutomaticFeatureSelection
1: generate a pool of features PF = \{F_i\}
2: initialize the set of selected features with the empty set SF = \{\emptyset\}
3: extract training and testing corpora for the given target ambiguous word
4: repeat
5: \hspace{1em} for each feature \(F_i \in PF\) do
6: \hspace{2em} run a disambiguation experiment on the training set; each example in the training set contains the features in SF and the feature \(F_i\)
7: \hspace{2em} determine the feature \(F_i\) leading to the best accuracy
8: \hspace{2em} remove \(F_i\) from PF and add it to SF
9: \hspace{1em} end for
10: until no improvements are obtained

(10)

function BackwardAutomaticFeatureSelection
1: generate a pool of features PF = \{F_i\}
2: extract training and testing corpora for the given target ambiguous word
3: repeat
4: \hspace{1em} for each PF \cap F_i do
5: \hspace{2em} run a disambiguation experiment on the training set
6: \hspace{2em} determine the feature \(F_i\) leading to the worst accuracy
7: \hspace{2em} remove \(F_i\) from PF
8: \hspace{1em} end for
9: until improvements are obtained or the accuracy remains stable
6.7 Scoring

Having trained a classifier and tested it on the provided examples, our system returns answers for each of the instances in the test set. Those answers, however, need to be checked and for this purpose Senseval-3 provides a special scoring software (scorer\textsuperscript{213}), developed in C. We scored our system for precision and recall against two of the three provided scoring schemes - fine-grained (the answers of the system need to match exactly according to the correct answers from the scorer) and coarse-grained (the answers are compared to the coarse-grained senses) (see Section 4.1).

The scoring software expects a specific format for the answers, which does not correspond to the output file from the memory-based learner. More precisely the scorer expects that each line contains the id for the lexical item (this is the value of the attribute \texttt{item} in the \texttt{<lexelt>} tag), the id for the instance (which is the value of the \texttt{id} attribute in the \texttt{<instance>} tag), all possible answers of the system (in the same vector) and if needed a comment. All separate parts of the answer vector need to be separated by spaces.

Scorer2 ignores all answers that have already been scored (in case more than one answer vector is provided for an instance). A very interesting feature of the software is the fact that it allows the use of weights (probabilities) for the answers.

Again for clarification, let us consider our toy example and how exactly the correct answer for it should look like. First we need the reference id for the lexical item, which in our case is \texttt{activate.v}. Second, the reference id for the instance - \texttt{activate.v.cald.12345678} and then the answer - 38201, which all together looks like:

\begin{equation}
\text{activate.v activate.v.cald.12345678 38201}
\end{equation}

6.8 Experimental Results

In order to get a good overview of the strengths and disadvantages of our system as well as a good assessment of the automatically added data to it we conducted several different experiments. We started with training word-experts only on the manually annotated instances (see Section 6.8.1), which shows how good the system is designed and gives us a basis to compare it to other already existing ones that were reported in the Senseval-3 competition. However, since our main aim is to examine the instances that we automatically gathered, we trained word-experts only on them as well (Section 6.8.2), or used them to extend the already existing Senseval-3 corpus (Section 6.8.3).

\textsuperscript{13}http://www.cse.unl.edu/~rada/senseval/senseval3/scoring/scoring.tar.gz
6.8.1 Supervised WSD

Before testing the automatically gathered data, we first have to be sure of how good our system is at all and will its performance be sufficient for the examination of other data. This is why, our first experiment was based on the data (both training and test data) that was already manually annotated and provided by the Senseval-3 competition.

As we described earlier in our thesis, we conduct several different experiments and optimizations until the final word-expert is trained. We included the forward and backward algorithm feature selection - similar to Dinu and Kübler (2007) and (Mihalcea, 2002) combined with a simplistic parameter optimization (both described in Section 6.6) which improved the systems results considerably.

Since it is not the aim of our thesis, we will not discuss in depth the improvements of the accuracy due to the parameter optimization. However, for the current and the following experiments we will always provide a short summary of the accuracy with both optimized (Table 6.6) and not optimized (Table 6.5) $k$. This will serve as a demonstration of how important parameter optimization can be for a WSD system and how such a simplistic optimization already results to good improvements for our work.

<table>
<thead>
<tr>
<th>feature selection</th>
<th>coarse P</th>
<th>coarse R</th>
<th>fine P</th>
<th>fine R</th>
</tr>
</thead>
<tbody>
<tr>
<td>all features</td>
<td>65.9</td>
<td>65.9</td>
<td>61.0</td>
<td>61.0</td>
</tr>
<tr>
<td>forward</td>
<td>76.1</td>
<td>76.1</td>
<td>72.4</td>
<td>72.4</td>
</tr>
<tr>
<td>backward</td>
<td>73.0</td>
<td>73.0</td>
<td>68.9</td>
<td>68.9</td>
</tr>
<tr>
<td>MFS</td>
<td>64.5</td>
<td>64.5</td>
<td>55.2</td>
<td>55.2</td>
</tr>
<tr>
<td><strong>best performance</strong></td>
<td><strong>77.6</strong></td>
<td><strong>77.6</strong></td>
<td><strong>74.0</strong></td>
<td><strong>74.0</strong></td>
</tr>
</tbody>
</table>

Table 6.5: System performance on manually annotated data without optimization of the $k$ parameter.

<table>
<thead>
<tr>
<th>feature selection</th>
<th>coarse P</th>
<th>coarse R</th>
<th>fine P</th>
<th>fine R</th>
</tr>
</thead>
<tbody>
<tr>
<td>all features</td>
<td>70.7</td>
<td>70.7</td>
<td>65.0</td>
<td>65.0</td>
</tr>
<tr>
<td>forward</td>
<td>78.7</td>
<td>78.7</td>
<td>74.3</td>
<td>74.3</td>
</tr>
<tr>
<td>backward</td>
<td>77.8</td>
<td>77.8</td>
<td>73.3</td>
<td>73.3</td>
</tr>
<tr>
<td>MFS</td>
<td>64.5</td>
<td>64.5</td>
<td>55.2</td>
<td>55.2</td>
</tr>
<tr>
<td><strong>best performance</strong></td>
<td><strong>79.3</strong></td>
<td><strong>79.3</strong></td>
<td><strong>75.1</strong></td>
<td><strong>75.1</strong></td>
</tr>
</tbody>
</table>

Table 6.6: System performance on manually annotated data with optimization of the $k$ parameter.

With the manually annotated data and an optimization of the $k$ parameter our system reaches results as in Table 6.6. As can be noted, parameter optimization increases the results with
approximately 2% for the given data set. For the results of individual words, refer to Table 7.1.

For a WSD system of such a simplistic kind as ours, the results that can be seen in Table 6.6 are considerably good. We reach an accuracy approximately 20% higher than the MFS classifier on that data which as Mihalcea et al. (2004a) report scores 64.5% (coarse) and 55.2% (fine) accuracy. In comparison with other systems from the Senseval-3 - Lexical Sample Task, the system we designed reaches a competitive coarse-grained result with the best reported systems in the competition. Moreover, our fine-grained accuracy is significantly better than all participating systems (see Table 6.7 below). For a full Table of all supervised WSD in Senseval-3 refer to Table 2.3 on page 19.

<table>
<thead>
<tr>
<th>System/Team</th>
<th>coarse</th>
<th>fine</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>P</td>
<td>R</td>
</tr>
<tr>
<td>htsa3/U.Bucharest(Grozea)</td>
<td>79.3</td>
<td>79.3</td>
</tr>
<tr>
<td>IRST - Kernels/TTC-IRST(Strapparava)</td>
<td>79.5</td>
<td>79.5</td>
</tr>
<tr>
<td>nusels/Nat.U.Singapore(Lee)</td>
<td>78.8</td>
<td>78.8</td>
</tr>
<tr>
<td>our system</td>
<td>79.3</td>
<td>79.3</td>
</tr>
</tbody>
</table>

Table 6.7: Comparison with the three best supervised systems in the Senseval-3 lexical sample task (Mihalcea et al., 2004a).

Those results show that our system reaches competitive results to the state-of-art WSD systems. It is interesting to note here that we compare a MBL system (ours) to a Naïve Bayes system (htsa3) (recall Section 2.3.5 where we mentioned that Mooney (1996) classifies Naïve Bayes as performing best among seven state-of-art machine learning algorithms) and two systems that use an SVM classifier - IRST and nusels. Those results give us a good basis for using our system in further experiments with data that has been automatically gathered and annotated.

### 6.8.2 Unsupervised WSD

As a completely unsupervised system we rely only on the data we gather automatically from online dictionaries and corpora (see Section 6.2.2). This means that our training set has not been influenced by any manually annotated data. As test set we used again the same Senseval-3 test set provided by the Lexical Sample Task in the competition. This gives us again the possibility to compare our results to already existing state-of-art systems that participated in the enterprise.

Without an optimization of the $k$ parameter the unsupervised approach shows rather poor results that are shown in Table 6.8 below. What can be seen in Table 6.9 however is that even with optimized parameters and the extremely large influence of the forward and backward feature selection algorithms the results are still not considerably good. Of course it will be very illogical to expect that the automatically gathered and labeled data will perform equal to the manually prepared one, but at least a comparison to the MFS classifier would be good to be in favor of the unsupervised approach. However, as we can see in Table 6.8 our unsupervised memory-based
learning approach is far below the accuracy of the MFS classifier. Even an optimized result (see Table 6.9) performs poorer than the given heuristic.

<table>
<thead>
<tr>
<th>feature selection</th>
<th>coarse P</th>
<th>coarse R</th>
<th>fine P</th>
<th>fine R</th>
</tr>
</thead>
<tbody>
<tr>
<td>all features</td>
<td>37.3</td>
<td>37.3</td>
<td>29.9</td>
<td>29.9</td>
</tr>
<tr>
<td>forward</td>
<td>50.2</td>
<td>50.2</td>
<td>44.1</td>
<td>44.1</td>
</tr>
<tr>
<td>backward</td>
<td>46.0</td>
<td>46.0</td>
<td>39.3</td>
<td>39.3</td>
</tr>
<tr>
<td>MFS</td>
<td>64.5</td>
<td>64.5</td>
<td>55.2</td>
<td>55.2</td>
</tr>
<tr>
<td><strong>best performance</strong></td>
<td><strong>51.6</strong></td>
<td><strong>51.6</strong></td>
<td><strong>43.1</strong></td>
<td><strong>43.1</strong></td>
</tr>
</tbody>
</table>

Table 6.8: System performance on automatically annotated data without optimization of the $k$ parameter.

<table>
<thead>
<tr>
<th>feature selection</th>
<th>coarse P</th>
<th>coarse R</th>
<th>fine P</th>
<th>fine R</th>
</tr>
</thead>
<tbody>
<tr>
<td>all features</td>
<td>45.4</td>
<td>45.4</td>
<td>35.5</td>
<td>35.5</td>
</tr>
<tr>
<td>forward</td>
<td>54.7</td>
<td>54.7</td>
<td>46.2</td>
<td>46.2</td>
</tr>
<tr>
<td>backward</td>
<td>53.2</td>
<td>53.2</td>
<td>44.2</td>
<td>44.2</td>
</tr>
<tr>
<td>MFS</td>
<td>64.5</td>
<td>64.5</td>
<td>55.2</td>
<td>55.2</td>
</tr>
<tr>
<td><strong>best performance</strong></td>
<td><strong>56.2</strong></td>
<td><strong>56.2</strong></td>
<td><strong>47.5</strong></td>
<td><strong>47.5</strong></td>
</tr>
</tbody>
</table>

Table 6.9: System performance on automatically annotated data with optimization of the $k$ parameter.

The results in Table 6.9 point out that our system achieves 56.2% (coarse) and 47.5% (fine) accuracy by using the automatically gathered data as training set. In comparison with the participants in Senseval-3, however, our very simplistic system reaches again competitive results.

<table>
<thead>
<tr>
<th>System/Team</th>
<th>coarse P</th>
<th>coarse R</th>
<th>fine P</th>
<th>fine R</th>
</tr>
</thead>
<tbody>
<tr>
<td>wsdiit/IITBombay(Ramakrishnan et. al.)</td>
<td>73.9</td>
<td>74.1</td>
<td>66.1</td>
<td>65.7</td>
</tr>
<tr>
<td>Cymfony/(Niu)</td>
<td>66.4</td>
<td>66.4</td>
<td>56.3</td>
<td>56.3</td>
</tr>
<tr>
<td>Prob0/Cambridge U. (Preiss)</td>
<td>63.6</td>
<td>63.6</td>
<td>54.7</td>
<td>54.7</td>
</tr>
<tr>
<td>clr04-Is/CL Research (Litkowski)</td>
<td>55.5</td>
<td>55.5</td>
<td>45.0</td>
<td>45.0</td>
</tr>
<tr>
<td>our system</td>
<td>56.2</td>
<td>56.2</td>
<td>47.5</td>
<td>47.5</td>
</tr>
</tbody>
</table>

Table 6.10: Comparison with the four best unsupervised systems in the Senseval-3 lexical sample task (Mihalcea et al., 2004a).

From Table 6.10 we can see that for both coarse and fine-grained accuracy the system we present reaches higher results than a system relying on definition properties - syntactic, semantic, subcategorization patterns as well as other lexical information (clr04-Is). The latter means that constructing a system like clr04-Is implies a lot of effort and computation, whereas our system is connected only with gathering the data and its automatic annotation, regardless any other
information that can be extracted from the examples. This shows that automatic extraction of examples for WSD can be considered a considerably effective and easy way for the construction of corpora for unsupervised word sense disambiguation approaches, which are competitive with the presented in Senseval-3 participating systems.

Generally, we cannot claim that this fully unsupervised approach reaches good results, but what is interesting in its performance is the fact that for several words it does get considerably close to a supervised approach (e.g. the words activate, argument, different, express) and in several examples the unsupervised outperformed the supervised one - difficulty, encounter, important. The exact results can be looked up in Table 7.1 and Table 7.2.

Another very important issue, which needs to be noted here is that despite of the huge number of examples in some cases the performance of the word-expert is relatively poor (e.g. ask, lose, play, treat) or even equal to 0 - solid. In the interesting case of solid the problem is that the automatically gathered sense inventory for the word does include proportionately too few instances of the senses that the word is tested on.

6.8.3 Semi-supervised WSD

In Section 2 we discussed the basic approaches to WSD and noted that except from the already defined knowledge and corpus-based (both supervised and unsupervised) approaches there are as well approaches that combine effort from different areas. In the previous section we looked at an application of our system as an unsupervised one. The reported results, however, even though competitive with other systems of this kind did not achieve results better than simple heuristics for the given data. Thus we continued our attempt to obtain the most out of the automatically prepared data and so conducted several semi-supervised experiments. All of the pursuits employ different subsets of the final data collection we have and thus demonstrate different usages, strengths and weaknesses of the mixture of automatically and manually annotated data.

Filtering based on the distribution of senses - In this experiment we used a very similar setting to our unsupervised approach. As a training set we used only the automatically annotated data and as a test set the provided by Senseval-3 Lexical Sample Task. However, the training set was filtered, such that not all instances were used as in the previous experiment.

With filtering here we mean that we excluded all examples that break the distribution of senses in the given by Senseval-3 Lexical Sample Task training set. To clarify we will use a simple example. Let us consider one of the words in the predefined lexical sample - the verb suspend. If in the original manually annotated training set this word appears with sense distributions as in (12) - to be read as: suspend appears in the manually annotated training set with the sense 4155301 - 19 times, with the sense 4155304 - 12 times, etc.

(12)
Respectively, we remember this distribution and filter our automatically prepared training set in such a way that the senses for the word `suspend' have the same proportional distribution as in (12). A possible result of the filtering would be for instance a set where we have enough examples to finally extract a training set for `suspend' that is double the size of the original one as is shown in (13) below:

(13)

The results we get from this approach are to be found in Table 6.11 and Table 6.12. An exhaustive table (Table 7.3) displaying all separate results per word-expert can be found further on page 99.

<table>
<thead>
<tr>
<th>feature selection</th>
<th>coarse</th>
<th>fine</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>P</td>
<td>R</td>
</tr>
<tr>
<td>all features</td>
<td>45.1</td>
<td>45.1</td>
</tr>
<tr>
<td>forward</td>
<td>62.4</td>
<td>62.4</td>
</tr>
<tr>
<td>backward</td>
<td>55.2</td>
<td>55.2</td>
</tr>
<tr>
<td>MFS</td>
<td>64.5</td>
<td>64.5</td>
</tr>
<tr>
<td><strong>best performance</strong></td>
<td>64.4</td>
<td>64.4</td>
</tr>
</tbody>
</table>

Table 6.11: System performance on automatically annotated data without optimization of the $k$ parameter and filtered for preservation of the distribution of the senses as in the original manually annotated training set.
CHAPTER 6. AUTOMATIC EXTRACTION OF EXAMPLES FOR WSD

<table>
<thead>
<tr>
<th>feature selection</th>
<th>coarse P</th>
<th>coarse R</th>
<th>fine P</th>
<th>fine R</th>
</tr>
</thead>
<tbody>
<tr>
<td>all features</td>
<td>54.7</td>
<td>54.7</td>
<td>47.0</td>
<td>47.0</td>
</tr>
<tr>
<td>forward</td>
<td>65.8</td>
<td>65.8</td>
<td>59.2</td>
<td>59.2</td>
</tr>
<tr>
<td>backward</td>
<td>63.3</td>
<td>63.3</td>
<td>56.8</td>
<td>56.8</td>
</tr>
<tr>
<td>MFS</td>
<td>64.5</td>
<td>64.5</td>
<td>55.2</td>
<td>55.2</td>
</tr>
<tr>
<td>best performance</td>
<td>66.7</td>
<td>66.7</td>
<td>60.2</td>
<td>60.2</td>
</tr>
</tbody>
</table>

Table 6.12: System performance on automatically annotated data with optimization of the $k$ parameter and filtered for preservation of the distribution of the senses as in the original manually annotated training set.

Filtering the training set in order to preserve the distribution as it is defined to be in the Senseval-3 Lexical Sample Task training set helps to a great extend. It improved the performance of our system with approximately 10%, which already is a result better than the MFS heuristic achieves on the data set. Recall the performance of the system on not filtered data (Table 6.9 on page 65) reaching 56.2% (coarse) and 47.5% (fine) accuracy against the filtered one (Table 6.12) reaching 66.7% (coarse) and 60.2% (fine) accuracy. This shows that memory-based learners are extremely sensitive to the clustering of senses in the training set against the clustering of senses in the test set. Moreover, a description of the distributions such as the one that we automatically extracted from the given training set implies considerably less effort than preparing the training set itself, which means that such relatively effective semi-supervised methods as the one we just described can be realized in an exceptionally "cheap" way employing significantly less human effort.

Filtering based on the distance of examples - Another very interesting usage of the automatically acquired data is connected with its utmost filtering and further addition to the manually prepared training set. With utmost filtering here we mean the following process: First, we let TiMBL output each of the instances from our automatically annotated training set together with a number representing the distance of the given instance to the most similar example in the manually annotated training set. Second, from the outputted from TiMBL set, we extract only those instances that do not differ too much from the manually annotated training set. The resulting collection of examples (consisting of 141 instances) we add to the given by Senseval-3 Lexical Sample Task training set. As test set we use again the original manually annotated test set. The following experiment achieves results that can be seen in Table 6.13 on page 69 and its optimized version - Table 6.14 on page 69. If needed, refer to Table 7.4 on page 100 for an exhaustive listing of the results per word-expert.

At first glance the improvement in the results (Table 6.14) are not big. However, we consider this experiment as exceedingly important and very successful, hence the changes in the separate word-experts are a lot more revealing than the final result. This is so, since the 141 new examples that we distribute amongst all of the words never exceed 6 instances for a different
word. Be that as it may, looking closely at Tables 7.1 and 7.4 we can note a variation of up to 4% as a result of those very few added examples. This means that even with very small number of instances, which are carefully selected out of a huge pool of examples we gain an exceptionally big variation and in some cases separate word-experts improve with up to 4% of accuracy.

<table>
<thead>
<tr>
<th>feature selection</th>
<th>coarse P</th>
<th>coarse R</th>
<th>fine P</th>
<th>fine R</th>
</tr>
</thead>
<tbody>
<tr>
<td>all features</td>
<td>67.0</td>
<td>67.0</td>
<td>62.3</td>
<td>62.3</td>
</tr>
<tr>
<td>forward</td>
<td>76.9</td>
<td>76.9</td>
<td>72.4</td>
<td>72.4</td>
</tr>
<tr>
<td>backward</td>
<td>73.0</td>
<td>73.0</td>
<td>68.9</td>
<td>68.9</td>
</tr>
<tr>
<td>MFS</td>
<td>64.5</td>
<td>64.5</td>
<td>55.2</td>
<td>55.2</td>
</tr>
<tr>
<td>best performance</td>
<td>78.5</td>
<td>78.5</td>
<td>73.9</td>
<td>73.9</td>
</tr>
</tbody>
</table>

Table 6.13: System performance on automatically annotated data without optimization of the $k$ parameter and distance and distribution filtering mixed with the manually annotated training set.

<table>
<thead>
<tr>
<th>feature selection</th>
<th>coarse P</th>
<th>coarse R</th>
<th>fine P</th>
<th>fine R</th>
</tr>
</thead>
<tbody>
<tr>
<td>all features</td>
<td>70.5</td>
<td>70.5</td>
<td>64.9</td>
<td>64.9</td>
</tr>
<tr>
<td>forward</td>
<td>78.7</td>
<td>78.7</td>
<td>74.0</td>
<td>74.0</td>
</tr>
<tr>
<td>backward</td>
<td>77.6</td>
<td>77.6</td>
<td>73.0</td>
<td>73.0</td>
</tr>
<tr>
<td>MFS</td>
<td>64.5</td>
<td>64.5</td>
<td>55.2</td>
<td>55.2</td>
</tr>
<tr>
<td>best performance</td>
<td>79.4</td>
<td>79.4</td>
<td>75.0</td>
<td>75.0</td>
</tr>
</tbody>
</table>

Table 6.14: System performance on automatically annotated data with optimization of the $k$ parameter and distance and distribution filtering mixed with the manually annotated training set.

Of course, extending this training set further on manually will result to a lot of very expensive human work. Our approach, however, can be used for this purpose. As we saw, from our original automatically collected training set of about 170 000 instance we managed to extract only 141 examples that are at the shortest similarity distance which bring improvement of the accuracy. Moreover, those 141 examples, even if so similar to the ones we already have always bring new information along. Consequently, this new information makes the classifier less sensitive to new examples. As a result we can conclude that adding automatically acquired examples to the manually prepared data set helps and that our approach reached such good results via an easy and very computationally "cheap" method for gaining new instances for training of WSD systems.

### 6.8.4 Discussion

The experiments we conducted and described in the previous few sections show that our system performs competitive to other state-of-art systems. The data that we used, however, has several
advantages but also some disadvantages, which can be clearly seen in the results of the experiments and which we will shortly discuss in this section. Since we used automatically prepared data together with the manually annotated one, we had the chance to observe the following issues:

**The manually annotated data - disadvantages:**

- The preparation of the data implies an extremely laborious, costly and expensive/intensive process of human annotation, which can hardly be accomplished in a very little time.
- The data is of a significantly small amount, which is the effect of its costly preparation.
- As a result of the manual annotation, the nature of the training data is optimally adapted to the nature of the test data.
  - proportional distribution of occurring senses
  - very similar length of the context
  - cover only a subset of the actual senses of the word
- The above mentioned issue leaves hardly any possibility for improvement, except by a carefully and precisely filtered external data.
- The preparation of such data implies also predefined semantic resources, which:
  - are not available in all languages
  - differ considerably in the quality
  - provide granularity of senses often different than the one needed for the task

**The manually annotated data - advantages:**

- As a result of its careful preparation the manually annotated data leads to exceptionally good system performance, outperforming unsupervised and semi-supervised methods.
- Employs only well selected senses that are predefined in the training data.
- Has an extremely homogeneous nature, which ensures good performance of the system.

**The automatically annotated data - disadvantages:**

- Used without any additional processing could lead to poor performance of the WSD system.
- The derived senses for the target word are specific for the corpus they are extracted from.
- Is never as homogeneous as manually prepared data and thus can lead to poorer results.
The automatically annotated data - advantages:

- Used selectively according to various criteria (which we discussed in our work) can be extremely valuable and employed for improvement of supervised WSD systems.
- Resources can be created with considerably small effort.
- Possibility to extract data for all languages that can provide corpora.
- The granularity of senses can be controlled by the choice of corpora.
- Can provide a big amount of examples.

All those advantages and disadvantages lead to the conclusion, that automatically vs. manually annotated data compete mostly in respect to the invested effort in creation of the data vs. the performance of the final system. However, in our work we showed that automatically annotated data can be used with several different purposes, which can achieve good results and thus be considered important.

In order to visualize better the difference between automatically annotated data and the manually prepared one we looked at the system performance on three different randomly chosen words (one noun, one verb and one adjective) from the lexical sample. We started with a training set consisting of only 10 randomly chosen instances and gradually added sets of new 10 examples and observed the results. The curves can be seen in Figure 7.1, Figure 7.2 and Figure 7.3 on page 93 and 95. What can be seen on the graphs is that gradual addition of instances generally lead to an increase in accuracy. Even though that the curves for the automatically annotated data are so far below the manually annotated one, which can be as well seen in the poor performance of our unsupervised experiment (see Table 6.9) we already showed, that this performance can be easily improved (refer to Table 6.12). Moreover, the constant increase of corpora ensures the fact that more instances can be easily extracted. But, of course, how many are enough will still stay an open question in the field. However, if we recall the interesting and extreme case of solid or as well other words as lose, talk and treat we see that the big number of examples does not always lead to good results. This is so, because as we noted the quality of those examples is, too, exceptionally important.
CHAPTER 6. AUTOMATIC EXTRACTION OF EXAMPLES FOR WSD

6.9 Evaluation

The following section is a summary of the results that our system achieved on the multiple experiments, which we conducted (see Section 6.8). Additionally we report the total system performance according to the measures discussed in Section 4.1. To begin with let us consider the four different experiments, which we completed. The difference between them was in the altered training set consisting of (the following sets are referred to further in the section with their numbers below):

1. Only Automatically Annotated Data
2. Filtered Automatically Annotated
3. Only Manually Annotated
4. Manually Annotated Data plus Filtered Automatically Annotated

From Table 6.15 we can see that our system achieves best results with training set 4.

<table>
<thead>
<tr>
<th>Set</th>
<th>all features</th>
<th>feature selection</th>
<th>best perf.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>coarse</td>
<td>fine</td>
<td>coarse</td>
</tr>
<tr>
<td>1</td>
<td>45.4</td>
<td>45.4</td>
<td>35.5</td>
</tr>
<tr>
<td>2</td>
<td>54.7</td>
<td>54.7</td>
<td>47.0</td>
</tr>
<tr>
<td>3</td>
<td>70.7</td>
<td>70.7</td>
<td>65.0</td>
</tr>
<tr>
<td>4</td>
<td>70.5</td>
<td>70.5</td>
<td>64.9</td>
</tr>
</tbody>
</table>

Table 6.15: Summary of the results for all experiments.

We always showed the accuracy of the system in terms of precision and recall, however in Section 4.1 we mentioned as well their harmonic average (used to represent the total performance of a system) - the F-score. The F-score will also allow us to compare the figures with the upper and lower bound for the experiment. Thus, in Table 6.16 we show the computed F-scores for the four experiments we conducted compared to the upper (IAA) and lower (MFS heuristic) bound for the English lexical sample task in the Senseval-3 competition as reported by (Mihalcea et al., 2004a).
Table 6.16: Comparison of our system’s scores with the upper and lower bound for the experiment and also with the best performing system in the Senseval-3 English lexical sample task.

<table>
<thead>
<tr>
<th></th>
<th>coarse</th>
<th>fine</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>56.2</td>
<td>47.5</td>
</tr>
<tr>
<td>MFS</td>
<td>64.5</td>
<td>55.2</td>
</tr>
<tr>
<td>2</td>
<td>66.7</td>
<td>60.2</td>
</tr>
<tr>
<td>IAA</td>
<td>67.3</td>
<td>-</td>
</tr>
<tr>
<td>htsa3</td>
<td>79.3</td>
<td>72.9</td>
</tr>
<tr>
<td>3</td>
<td>79.3</td>
<td>75.1</td>
</tr>
<tr>
<td>4</td>
<td>79.4</td>
<td>75.0</td>
</tr>
</tbody>
</table>

Table 6.16 shows that our system performs best on the given task by attempting a semi-supervised approach. The latter employs the given by the Senseval-3 English lexical sample task training set that we extended with 141 automatically annotated instances. It outperforms both the upper and the lower bound for the given task and as well the best performing supervised system in the Senseval-3 English lexical sample task.
Chapter 7

Conclusion, Future and Related Work

7.1 Related Work

There have been several approaches that relate to our work in the sense that they approach the automatic acquisition of sense-tagged corpora and its further usage in word sense disambiguation. Those attempts are described in more detail in (Gonzalo and Verdejo, 2007). The authors classify the strategies in five different types:

1. Acquisition by direct web searching
2. Bootstrapping from seed examples
3. Acquisition via web directories
4. Acquisition via cross-language evidence
5. Web-based cooperative annotation

We will not aim to discuss in depth any of those approaches but rather note some of the conclusions made by Gonzalo and Verdejo (2007) about their general nature, which relates to the observed by us importance of the automatically acquired sense-tagged data.

One such remark is the fact that similar to our experience, the quality of the automatically gathered data equals, or even outperforms the performance of the human-tagged examples.

Another relevant issue that the authors mention is the correlation between the correctness of the annotation of the examples and the final performance of the system. Of course, it is the case that if the instances are correctly labeled normally the results will be better, however, there are cases in which the performance of the supervised system is very close to the performance of the unsupervised one. This means that not always the selected examples are useful for training a classifier.
The last related to our work remark from the above observed by Gonzalo and Verdejo (2007) approaches for automatic acquisition of examples for WSD is the fact that the performance of unsupervised strategies is far poorer than the performance of supervised ones. Similar to our unsupervised experiment, the authors mention, that in most of the cases the approaches of this family do not even reach the MFS baseline.
CHAPTER 7. CONCLUSION, FUTURE AND RELATED WORK

7.2 Future Work

An interesting question in our work, which we assume can lead to very good performance is the choice of words for which examples are extracted. Consider again our experiment based on the addition of the closest instances from the automatically prepared training set to the manually annotated one. As we reported, for a big subset of the words there is accuracy variance of the separate word-experts of about 4%. To figure out the tendency with which the results rise or fall would be a valuable information that can be used in order to add selectively the closest instances.

Another issue worth further investigation is what exactly is the effect of the proportional distribution of senses on the final results. It will be interesting to see if the performance of the supervised systems would stay the same if the training and test sets provided by Senseval-3 did not keep such close proportions of the senses.

Parameter optimization is also an extremely important part of a WSD system, which can lead to a considerable increase of its performance and respectively of the performance of the separate word-experts. In our work we managed to show only a very limited parameter optimization concentrated only on three different values for a single parameter. The exceptionally good results of the attempt prove that further investigation of the parameter optimization is essential and will most certainly lead to improvement of the performance of the WSD system.

7.3 Conclusion

This thesis describes the design and performance of a simplistic memory-based word sense disambiguation system, which makes use of automatic feature selection and minimal parameter optimization. We show that the system performs competitive to other state-of-art systems and use it further for evaluation of automatically acquired data for word sense disambiguation.

In our work we conducted several experiments employing the automatically extracted data as training instances. The results of our approach prove that automatic extraction of examples for word sense disambiguation can help to a great extend for the improvement of manually annotated training data. We showed that by selecting only the most similar automatically annotated instances and adding them to the manually annotated ones the performance of the separate word-experts can rise up to 4%.

Nowadays, the increasing number and size of freely available corpora extremely accomodates the arising need for resources, which are required for approaches similar to ours. Once provided such corpora, however, we demonstrated that the potential of those approaches for improvement of the performance of word sense disambiguation systems by automatically extracted data is immense and determines their further exploration as more than worthwhile.
Bibliography


BIBLIOGRAPHY


Daelemans, W., J. Zavrel, A. van den Bosch and K. can der Sloot (2007), TiMLB: Tilburg Memory-Based Learner, Tech. rep., Induction of Linguistic Knowledge Research Group, Department of Communication and Information Sciences, Tilburg University; CNTS-Language Technology Group, Department of Linguistics, University of Antwerp.


Mihalcea, R. and T. Chklovski (2003), Open mind word expert: Creating large annotated data collections with web users’ help, in *Proceedings of 4th International Workshop on Linguistically Interpreted Corpora (LINC-03) held in conjunction with EACL*.


Rosenberg, C., M. Hebert and H. Schneiderman (2005), Semi-supervised selftraining of object detection models., in *Seventh IEEE Workshop on Applications of Computer Vision*.


## A List of Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AW</td>
<td>Ambiguous Word</td>
</tr>
<tr>
<td>BNC</td>
<td>British National Corpus</td>
</tr>
<tr>
<td>FV</td>
<td>Feature Vector</td>
</tr>
<tr>
<td>GNU GPL</td>
<td>GNU General Public License</td>
</tr>
<tr>
<td>IAA</td>
<td>Inter-Annotator Agreement</td>
</tr>
<tr>
<td>kNN</td>
<td>$k$-Nearest Neighbor</td>
</tr>
<tr>
<td>MBL</td>
<td>Memory-Based Learning</td>
</tr>
<tr>
<td>MFS</td>
<td>Most Frequent Sense</td>
</tr>
<tr>
<td>OMWE</td>
<td>Open Mind Word Expert</td>
</tr>
<tr>
<td>P</td>
<td>Precision</td>
</tr>
<tr>
<td>PF</td>
<td>Pool of Features</td>
</tr>
<tr>
<td>POS</td>
<td>Part-of-Speech</td>
</tr>
<tr>
<td>R</td>
<td>Recall</td>
</tr>
<tr>
<td>SVM</td>
<td>Support Vector Machine</td>
</tr>
<tr>
<td>TiMBL</td>
<td>Tilburg Memory-Based Learner</td>
</tr>
<tr>
<td>WSD</td>
<td>Word Sense Disambiguation</td>
</tr>
<tr>
<td>XML</td>
<td>Extensible Markup Language</td>
</tr>
</tbody>
</table>
B Pool of Features

**CW** Current word (0-param). The ambiguous word (AW) itself.
- Notation: CW
- References: (Mihalcea, 2002), (Dinu and Kübler, 2007), (Usabaev, 2008)

**CW** Current word (1-param). The word in the surrounding of the AW - at position \( k \).
- Notation: CW[=K], optimization=[-3...3]
- References: (Dinu and Kübler, 2007), (Usabaev, 2008)

**CP** Current part-of-speech (0-param). The part-of-speech of the AW.
- Notation: CP
- References: (Mihalcea, 2002)

**CP** Current part-of-speech (1-param). The part-of-speech of the word in the surrounding of the AW - at position \( k \).
- Notation: CP
- References: (Dinu and Kübler, 2007), (Usabaev, 2008)

**CF** Contextual features (1-param). The words and parts of speech of K words surrounding the AW.
- Notation: CF[=K], default=3
- References: (Mihalcea, 2002)

**COL** Collocations (1-param) Collocations (Ng and Lee, 1996) formed with maximum K words surrounding the AW.
- Notation: COL[=K], default=3
- References: (Mihalcea, 2002)

**HNP** Head of noun phrase (0-param). The head of the noun phrase to which the AW belongs, if any.
- Notation: HNP
- References: (Mihalcea, 2002)

**SK** Sense specific keywords (2-param). Maximum MX keywords occurring at least MN times (Ng and Lee, 1996) are determined for each sense of the ambiguous word. The value of this feature is either 0 or 1, depending if the current example contains one of the determined keywords or not.
- Notation: SK[=MN, MX], default=5, 5
- References: (Mihalcea, 2002)

**B** Bigrams (2-param). Maximum MX bigrams occurring at least MN times are determined for all training examples. The value of this feature is either 0 or 1, depending if the current example contains one of the determined bigrams or not.
- Notation: B[=MN, MX], default=5,20
- References: (Mihalcea, 2002)

**VB** Verb before (0-param). The first verb found before the AW.
- Notation: VB
- References: (Mihalcea, 2002), citep18, (Usabaev, 2008)

**VA** Verb after (0-param). The first verb found after the AW.
- Notation: VA
- References: (Mihalcea, 2002), (Dinu and Kübler, 2007), (Usabaev, 2008)

**NB** Noun before (0-param). The first noun found before the AW.
- Notation: NB
- References: (Mihalcea, 2002), (Dinu and Kübler, 2007), (Usabaev, 2008)

**NA** Noun after (0-param). The first noun found after the AW.
- Notation: NA
- References: (Mihalcea, 2002), (Dinu and Kübler, 2007), (Usabaev, 2008)
NEB  Named Entity before (0-param). The first
Named Entity found before the AW.
- Notation: NEB
- References: (Mihalcea, 2002)

NEA  Named Entity after (0-param). The first
Named Entity found after the AW.
- Notation: NEA
- References: (Mihalcea, 2002)

PB   Preposition before (0-param). The first prepo-
sition found before the AW.
- Notation: PB
- References: (Mihalcea, 2002), (Dinu and
Kübler, 2007), (Usabaev, 2008)

PA   Preposition after (0-param). The first prepo-
sition found after the AW.
- Notation: PA
- References: (Mihalcea, 2002), (Dinu and
Kübler, 2007), (Usabaev, 2008)

PRB  Pronoun before (0-param). The first pro-
noun found before the AW.
- Notation: PRB
- References: (Mihalcea, 2002)

PRA  Pronoun after (0-param). The first pro-
noun found after the AW.
- Notation: PRA
- References: (Mihalcea, 2002)

DT   Determiner (0-param). The determiner, if
any, found before the AW.
- Notation: DT
- References: (Mihalcea, 2002)

PPT  Parse path (1-param) Maximum K parse
components found on the path to the top
of the parse tree (sentence top).
- Notation: PPT[=K], default=10 - For in-
stance, the value of this feature for the
word school, given a parse tree (S (NP (JJ
big) (NN house))), is NN, NP, S
- References: (Mihalcea, 2002) - described
but not used

SPC  Same parse phrase components (1-param).
Maximum K parse components found in
the same phrase as the AW.
- Notation: SPC[=K], default=3 - For the
example in the PPT feature, this feature
would be set to JJ, NN.
- References: (Mihalcea, 2002)
## C Training and Test Set (Senseval-3 English Lexical Sample Task)

<table>
<thead>
<tr>
<th>Lexical unit</th>
<th>Training size</th>
<th>Test size</th>
<th>Lexical unit</th>
<th>Training size</th>
<th>Test size</th>
</tr>
</thead>
<tbody>
<tr>
<td>activate.v</td>
<td>228</td>
<td>114</td>
<td>miss.v</td>
<td>58</td>
<td>30</td>
</tr>
<tr>
<td>add.v</td>
<td>263</td>
<td>132</td>
<td>note.v</td>
<td>132</td>
<td>67</td>
</tr>
<tr>
<td>appear.v</td>
<td>265</td>
<td>133</td>
<td>operate.v</td>
<td>35</td>
<td>18</td>
</tr>
<tr>
<td>argument.n</td>
<td>221</td>
<td>111</td>
<td>organization.n</td>
<td>112</td>
<td>56</td>
</tr>
<tr>
<td>arm.n</td>
<td>266</td>
<td>133</td>
<td>paper.n</td>
<td>232</td>
<td>117</td>
</tr>
<tr>
<td>ask.v</td>
<td>261</td>
<td>131</td>
<td>party.n</td>
<td>230</td>
<td>116</td>
</tr>
<tr>
<td>atmosphere.n</td>
<td>161</td>
<td>81</td>
<td>performance.n</td>
<td>172</td>
<td>87</td>
</tr>
<tr>
<td>audience.n</td>
<td>200</td>
<td>100</td>
<td>plan.n</td>
<td>166</td>
<td>84</td>
</tr>
<tr>
<td>bank.n</td>
<td>262</td>
<td>132</td>
<td>play.v</td>
<td>104</td>
<td>52</td>
</tr>
<tr>
<td>begin.v</td>
<td>181</td>
<td>79</td>
<td>produce.v</td>
<td>186</td>
<td>94</td>
</tr>
<tr>
<td>climb.v</td>
<td>133</td>
<td>67</td>
<td>provide.v</td>
<td>136</td>
<td>69</td>
</tr>
<tr>
<td>decide.v</td>
<td>122</td>
<td>62</td>
<td>receive.v</td>
<td>52</td>
<td>27</td>
</tr>
<tr>
<td>degree.n</td>
<td>256</td>
<td>128</td>
<td>remain.v</td>
<td>139</td>
<td>70</td>
</tr>
<tr>
<td>difference.n</td>
<td>226</td>
<td>114</td>
<td>rule.v</td>
<td>59</td>
<td>30</td>
</tr>
<tr>
<td>different.a</td>
<td>98</td>
<td>50</td>
<td>shelter.n</td>
<td>196</td>
<td>98</td>
</tr>
<tr>
<td>difficulty.n</td>
<td>46</td>
<td>23</td>
<td>simple.a</td>
<td>36</td>
<td>18</td>
</tr>
<tr>
<td>disc.n</td>
<td>200</td>
<td>100</td>
<td>smell.v</td>
<td>108</td>
<td>55</td>
</tr>
<tr>
<td>eat.v</td>
<td>181</td>
<td>87</td>
<td>solid.a</td>
<td>58</td>
<td>29</td>
</tr>
<tr>
<td>encounter.v</td>
<td>130</td>
<td>65</td>
<td>sort.n</td>
<td>190</td>
<td>96</td>
</tr>
<tr>
<td>expect.v</td>
<td>156</td>
<td>78</td>
<td>source.n</td>
<td>64</td>
<td>32</td>
</tr>
<tr>
<td>express.v</td>
<td>110</td>
<td>55</td>
<td>suspend.v</td>
<td>128</td>
<td>64</td>
</tr>
<tr>
<td>hear.v</td>
<td>63</td>
<td>32</td>
<td>talk.v</td>
<td>146</td>
<td>73</td>
</tr>
<tr>
<td>hot.a</td>
<td>86</td>
<td>43</td>
<td>treat.v</td>
<td>112</td>
<td>57</td>
</tr>
<tr>
<td>image.n</td>
<td>146</td>
<td>74</td>
<td>use.v</td>
<td>26</td>
<td>14</td>
</tr>
<tr>
<td>important.a</td>
<td>36</td>
<td>19</td>
<td>wash.v</td>
<td>66</td>
<td>34</td>
</tr>
<tr>
<td>interest.n</td>
<td>185</td>
<td>93</td>
<td>watch.v</td>
<td>100</td>
<td>51</td>
</tr>
<tr>
<td>judgment.n</td>
<td>62</td>
<td>32</td>
<td>win.v</td>
<td>78</td>
<td>39</td>
</tr>
<tr>
<td>lose.v</td>
<td>71</td>
<td>36</td>
<td>write.v</td>
<td>44</td>
<td>23</td>
</tr>
<tr>
<td>mean.v</td>
<td>80</td>
<td>40</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| Total        | 7860         | 3944      |
D Graphs

Figure 7.1: Accuracy change of the adjective *hot*.
Figure 7.2: Accuracy change of the noun *arm*. 
Figure 7.3: Accuracy change of the verb *appear*. 
E Tables

<table>
<thead>
<tr>
<th>word</th>
<th>all features</th>
<th>coarse</th>
<th>fine</th>
<th>forward</th>
<th>coarse</th>
<th>fine</th>
<th>backward</th>
<th>coarse</th>
<th>fine</th>
<th>best performance</th>
<th>coarse</th>
<th>fine</th>
<th>test</th>
<th>train</th>
</tr>
</thead>
<tbody>
<tr>
<td>activate</td>
<td>82.5</td>
<td>82.5</td>
<td>82.5</td>
<td>82.9</td>
<td>82.9</td>
<td>82.9</td>
<td>82.9</td>
<td>82.8</td>
<td>83</td>
<td>83.8</td>
<td>83.8</td>
<td>83.8</td>
<td>114</td>
<td>228</td>
</tr>
<tr>
<td>add</td>
<td>84.1</td>
<td>84.1</td>
<td>84.1</td>
<td>84.1</td>
<td>84.1</td>
<td>84.1</td>
<td>84.1</td>
<td>84.8</td>
<td>84.8</td>
<td>84.8</td>
<td>84.8</td>
<td>84.8</td>
<td>132</td>
<td>263</td>
</tr>
<tr>
<td>appear</td>
<td>71.9</td>
<td>71.8</td>
<td>71.8</td>
<td>71.8</td>
<td>71.8</td>
<td>71.8</td>
<td>71.8</td>
<td>71.7</td>
<td>71.7</td>
<td>71.7</td>
<td>71.7</td>
<td>71.7</td>
<td>133</td>
<td>269</td>
</tr>
<tr>
<td>argument</td>
<td>58.1</td>
<td>58.1</td>
<td>52.7</td>
<td>52.7</td>
<td>52.7</td>
<td>52.7</td>
<td>52.7</td>
<td>57.7</td>
<td>60.4</td>
<td>60.4</td>
<td>55.9</td>
<td>55.9</td>
<td>61.3</td>
<td>221</td>
</tr>
<tr>
<td>arm</td>
<td>87.2</td>
<td>87.2</td>
<td>87.2</td>
<td>87.2</td>
<td>87.2</td>
<td>87.2</td>
<td>87.2</td>
<td>89.5</td>
<td>89.5</td>
<td>89.5</td>
<td>89.5</td>
<td>89.5</td>
<td>90.2</td>
<td>203</td>
</tr>
<tr>
<td>ask</td>
<td>55.1</td>
<td>55.1</td>
<td>53.1</td>
<td>53.1</td>
<td>53.1</td>
<td>53.1</td>
<td>53.1</td>
<td>59.9</td>
<td>59.9</td>
<td>59.9</td>
<td>59.9</td>
<td>59.9</td>
<td>64.1</td>
<td>131</td>
</tr>
<tr>
<td>atmosphere</td>
<td>64.2</td>
<td>64.2</td>
<td>64.2</td>
<td>64.2</td>
<td>64.2</td>
<td>64.2</td>
<td>64.2</td>
<td>67.9</td>
<td>67.9</td>
<td>67.9</td>
<td>67.9</td>
<td>67.9</td>
<td>71</td>
<td>81</td>
</tr>
<tr>
<td>audience</td>
<td>97</td>
<td>97</td>
<td>71.5</td>
<td>71.5</td>
<td>71.5</td>
<td>71.5</td>
<td>71.5</td>
<td>79.6</td>
<td>79.6</td>
<td>79.6</td>
<td>79.6</td>
<td>79.6</td>
<td>97</td>
<td>61</td>
</tr>
<tr>
<td>bank</td>
<td>80.8</td>
<td>80.8</td>
<td>72</td>
<td>72</td>
<td>72</td>
<td>72</td>
<td>72</td>
<td>84.1</td>
<td>84.1</td>
<td>84.1</td>
<td>78.4</td>
<td>78.4</td>
<td>88.6</td>
<td>132</td>
</tr>
<tr>
<td>begin</td>
<td>62</td>
<td>62</td>
<td>62</td>
<td>70.9</td>
<td>70.9</td>
<td>70.9</td>
<td>70.9</td>
<td>72.8</td>
<td>72.8</td>
<td>72.8</td>
<td>72.8</td>
<td>72.8</td>
<td>72.8</td>
<td>81</td>
</tr>
<tr>
<td>climb</td>
<td>65.7</td>
<td>65.7</td>
<td>65.7</td>
<td>65.7</td>
<td>65.7</td>
<td>65.7</td>
<td>65.7</td>
<td>77.6</td>
<td>77.6</td>
<td>77.6</td>
<td>77.6</td>
<td>77.6</td>
<td>77.6</td>
<td>181</td>
</tr>
<tr>
<td>decide</td>
<td>82.5</td>
<td>82.5</td>
<td>82.5</td>
<td>82.5</td>
<td>82.5</td>
<td>82.5</td>
<td>82.5</td>
<td>86.5</td>
<td>86.5</td>
<td>86.5</td>
<td>86.5</td>
<td>86.5</td>
<td>86.5</td>
<td>122</td>
</tr>
<tr>
<td>degree</td>
<td>81.6</td>
<td>81.6</td>
<td>71.5</td>
<td>71.5</td>
<td>71.5</td>
<td>71.5</td>
<td>71.5</td>
<td>86.7</td>
<td>86.7</td>
<td>86.7</td>
<td>78.7</td>
<td>78.7</td>
<td>86.7</td>
<td>256</td>
</tr>
<tr>
<td>difference</td>
<td>65.8</td>
<td>65.8</td>
<td>54.4</td>
<td>54.4</td>
<td>54.4</td>
<td>54.4</td>
<td>54.4</td>
<td>66.2</td>
<td>66.2</td>
<td>66.2</td>
<td>66.2</td>
<td>66.2</td>
<td>66.2</td>
<td>226</td>
</tr>
<tr>
<td>different</td>
<td>70</td>
<td>70</td>
<td>54</td>
<td>54</td>
<td>54</td>
<td>54</td>
<td>54</td>
<td>72</td>
<td>72</td>
<td>72</td>
<td>72</td>
<td>72</td>
<td>72</td>
<td>98</td>
</tr>
<tr>
<td>difficulty</td>
<td>44.9</td>
<td>44.9</td>
<td>56.5</td>
<td>56.5</td>
<td>56.5</td>
<td>56.5</td>
<td>56.5</td>
<td>87</td>
<td>87</td>
<td>87</td>
<td>87</td>
<td>87</td>
<td>87</td>
<td>33</td>
</tr>
<tr>
<td>disc</td>
<td>46</td>
<td>46</td>
<td>46</td>
<td>52.5</td>
<td>52.5</td>
<td>52.5</td>
<td>52.5</td>
<td>56</td>
<td>56</td>
<td>56</td>
<td>56</td>
<td>56</td>
<td>56</td>
<td>100</td>
</tr>
<tr>
<td>eat</td>
<td>88.5</td>
<td>88.5</td>
<td>88.5</td>
<td>88.5</td>
<td>88.5</td>
<td>88.5</td>
<td>88.5</td>
<td>89.7</td>
<td>89.7</td>
<td>89.7</td>
<td>89.7</td>
<td>89.7</td>
<td>89.7</td>
<td>181</td>
</tr>
<tr>
<td>encounter</td>
<td>96.9</td>
<td>96.9</td>
<td>81.9</td>
<td>81.9</td>
<td>81.9</td>
<td>81.9</td>
<td>81.9</td>
<td>99.6</td>
<td>99.6</td>
<td>99.6</td>
<td>99.6</td>
<td>99.6</td>
<td>99.6</td>
<td>160</td>
</tr>
<tr>
<td>expect</td>
<td>80.8</td>
<td>80.8</td>
<td>80.8</td>
<td>80.8</td>
<td>80.8</td>
<td>80.8</td>
<td>80.8</td>
<td>85.9</td>
<td>85.9</td>
<td>85.9</td>
<td>85.9</td>
<td>85.9</td>
<td>85.9</td>
<td>156</td>
</tr>
<tr>
<td>express</td>
<td>81.8</td>
<td>81.8</td>
<td>89.1</td>
<td>89.1</td>
<td>89.1</td>
<td>89.1</td>
<td>89.1</td>
<td>89.1</td>
<td>89.1</td>
<td>89.1</td>
<td>89.1</td>
<td>89.1</td>
<td>89.1</td>
<td>150</td>
</tr>
<tr>
<td>bear</td>
<td>68.6</td>
<td>68.6</td>
<td>59.4</td>
<td>59.4</td>
<td>59.4</td>
<td>59.4</td>
<td>59.4</td>
<td>78.1</td>
<td>78.1</td>
<td>78.1</td>
<td>78.1</td>
<td>78.1</td>
<td>78.1</td>
<td>32</td>
</tr>
<tr>
<td>hot</td>
<td>79.1</td>
<td>79.1</td>
<td>79.1</td>
<td>79.1</td>
<td>79.1</td>
<td>79.1</td>
<td>79.1</td>
<td>81.4</td>
<td>81.4</td>
<td>81.4</td>
<td>81.4</td>
<td>81.4</td>
<td>81.4</td>
<td>43</td>
</tr>
<tr>
<td>image</td>
<td>55.4</td>
<td>55.4</td>
<td>55.4</td>
<td>55.4</td>
<td>55.4</td>
<td>55.4</td>
<td>55.4</td>
<td>66.2</td>
<td>66.2</td>
<td>66.2</td>
<td>66.2</td>
<td>66.2</td>
<td>66.2</td>
<td>74</td>
</tr>
<tr>
<td>important</td>
<td>63.2</td>
<td>63.2</td>
<td>42.1</td>
<td>42.1</td>
<td>42.1</td>
<td>42.1</td>
<td>42.1</td>
<td>65.8</td>
<td>65.8</td>
<td>65.8</td>
<td>65.8</td>
<td>65.8</td>
<td>65.8</td>
<td>19</td>
</tr>
<tr>
<td>interest</td>
<td>69.9</td>
<td>69.9</td>
<td>69.9</td>
<td>69.9</td>
<td>69.9</td>
<td>69.9</td>
<td>69.9</td>
<td>74.2</td>
<td>74.2</td>
<td>74.2</td>
<td>74.2</td>
<td>74.2</td>
<td>74.2</td>
<td>135</td>
</tr>
<tr>
<td>judgment</td>
<td>34.4</td>
<td>34.4</td>
<td>28.1</td>
<td>28.1</td>
<td>28.1</td>
<td>28.1</td>
<td>28.1</td>
<td>50.2</td>
<td>50.2</td>
<td>50.2</td>
<td>50.2</td>
<td>50.2</td>
<td>50.2</td>
<td>122</td>
</tr>
<tr>
<td>lose</td>
<td>44.4</td>
<td>44.4</td>
<td>44.4</td>
<td>44.4</td>
<td>44.4</td>
<td>44.4</td>
<td>44.4</td>
<td>61.1</td>
<td>61.1</td>
<td>61.1</td>
<td>61.1</td>
<td>61.1</td>
<td>61.1</td>
<td>36</td>
</tr>
<tr>
<td>mean</td>
<td>67.5</td>
<td>67.5</td>
<td>67.5</td>
<td>67.5</td>
<td>67.5</td>
<td>67.5</td>
<td>67.5</td>
<td>78.5</td>
<td>78.5</td>
<td>78.5</td>
<td>78.5</td>
<td>78.5</td>
<td>78.5</td>
<td>40</td>
</tr>
</tbody>
</table>

Table 7.1: The performance of our system on the Senseval-3 Lexical Sample Task data.
### BIBLIOGRAPHY

The performance of our system trained on the automatically gathered examples and tested on the Senseval-3 Lexical Sample Task test set.

<table>
<thead>
<tr>
<th>word</th>
<th>all features forward</th>
<th>backward</th>
<th>best performance</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>coarse</td>
<td>fine</td>
<td>coarse</td>
</tr>
<tr>
<td>activate</td>
<td>49.1</td>
<td>49.1</td>
<td>49.1</td>
</tr>
<tr>
<td>add</td>
<td>34.8</td>
<td>34.8</td>
<td>34.8</td>
</tr>
<tr>
<td>appear</td>
<td>45.1</td>
<td>45.1</td>
<td>45.1</td>
</tr>
<tr>
<td>arm</td>
<td>47.7</td>
<td>47.7</td>
<td>47.7</td>
</tr>
<tr>
<td>ask</td>
<td>48.5</td>
<td>48.5</td>
<td>48.5</td>
</tr>
<tr>
<td>atmosphere</td>
<td>34.8</td>
<td>34.8</td>
<td>34.8</td>
</tr>
<tr>
<td>audience</td>
<td>36.5</td>
<td>36.5</td>
<td>36.5</td>
</tr>
<tr>
<td>bank</td>
<td>50.5</td>
<td>50.5</td>
<td>50.5</td>
</tr>
<tr>
<td>begin</td>
<td>50.3</td>
<td>50.3</td>
<td>50.3</td>
</tr>
<tr>
<td>climb</td>
<td>52.2</td>
<td>52.2</td>
<td>52.2</td>
</tr>
<tr>
<td>decide</td>
<td>58.1</td>
<td>58.1</td>
<td>58.1</td>
</tr>
<tr>
<td>degree</td>
<td>54.7</td>
<td>54.7</td>
<td>54.7</td>
</tr>
<tr>
<td>difference</td>
<td>27.2</td>
<td>27.2</td>
<td>27.2</td>
</tr>
<tr>
<td>different</td>
<td>70</td>
<td>70</td>
<td>70</td>
</tr>
<tr>
<td>difficulty</td>
<td>93.1</td>
<td>93.1</td>
<td>93.1</td>
</tr>
<tr>
<td>disc</td>
<td>43</td>
<td>43</td>
<td>43</td>
</tr>
<tr>
<td>eat</td>
<td>31</td>
<td>31</td>
<td>31</td>
</tr>
<tr>
<td>encounter</td>
<td>96.9</td>
<td>96.9</td>
<td>96.9</td>
</tr>
<tr>
<td>expect</td>
<td>86.7</td>
<td>86.7</td>
<td>86.7</td>
</tr>
<tr>
<td>express</td>
<td>78.2</td>
<td>78.2</td>
<td>78.2</td>
</tr>
<tr>
<td>hear</td>
<td>31.2</td>
<td>31.2</td>
<td>31.2</td>
</tr>
<tr>
<td>hot</td>
<td>46.8</td>
<td>46.8</td>
<td>46.8</td>
</tr>
<tr>
<td>image</td>
<td>24.3</td>
<td>24.3</td>
<td>24.3</td>
</tr>
<tr>
<td>important</td>
<td>57.9</td>
<td>57.9</td>
<td>57.9</td>
</tr>
<tr>
<td>interest</td>
<td>14</td>
<td>14</td>
<td>14</td>
</tr>
<tr>
<td>judgment</td>
<td>21.9</td>
<td>21.9</td>
<td>21.9</td>
</tr>
<tr>
<td>lose</td>
<td>11.1</td>
<td>11.1</td>
<td>11.1</td>
</tr>
<tr>
<td>mean</td>
<td>17.5</td>
<td>17.5</td>
<td>17.5</td>
</tr>
</tbody>
</table>

Table 7.2: The performance of our system trained on the automatically gathered examples and tested on the Senseval-3 Lexical Sample Task test set.
Table 7.3: System performance on automatically annotated data with optimization of the $k$ parameter and filtered for preservation of the distribution of the senses as in the original manually annotated training set.
Table 7.4: System performance on automatically annotated data with optimization of the $k$ parameter and distance filtering with the manually annotated training set.