Thesis Dissertation:

Machine Learning Techniques for Word Sense Disambiguation

Gerard Escudero Bakx

Thesis advisors:

Lluís Màrquez Villodre and German Rigau Claramunt

For the obtention of the PhD Degree at the
Universitat Politècnica de Catalunya

Barcelona, May 22, 2006

Abstract

In the Natural Language Processing (NLP) community, Word Sense Disambiguation (WSD) has been described as the task which selects the appropriate meaning (sense) to a given word in a text or discourse where this meaning is distinguishable from other senses potentially attributable to that word. These senses could be seen as the target labels of a classification problem. That is, Machine Learning (ML) seems to be a possible way to tackle this problem.

This work studies the possible application of the algorithms and techniques of the Machine Learning field in order to handle the WSD task.

The first issue treated has been the adaptation of alternative ML algorithms to deal with word senses as classes. Then, a comparison of these methods is performed under the same conditions. The evaluation measures applied to compare the performances of these methods are the typical precision and recall, but also agreement rates and kappa statistics.

The second topic explored is the cross-corpora application of supervised Machine Learning systems for WSD to test the generalisation ability across corpora and domains. The results obtained are very disappointing, seriously questioning the possibility of constructing a general enough training corpus (labelled or unlabelled), and the way its examples should be used to develop a general purpose Word Sense Tagger.

The use of unlabelled data to train classifiers for Word Sense Disambiguation is a very challenging line of research in order to develop a really robust, complete and accurate Word Sense Tagger. Due to this fact, the next topic treated in this work is the application of two bootstrapping approaches on WSD: the Transductive Support Vector Machines and the Greedy Agreement bootstrapping algorithm by Steven Abney.

During the development of this research we have been interested in the construction and evaluation of several WSD systems. We have participated in the last two editions of the English Lexical Sample task of Senseval evaluation exercises. The Lexical Sample tasks are mainly oriented to evaluate Supervised Machine Learning systems. Our systems achieved

a very good performance in both editions of this competition. That is, our systems are among the state-of-the-art on WSD. As a complementary work of this participation, some other issues have been explored: 1) a comparative study of the features used to represent examples for WSD; 2) the comparison of different feature selection procedures; and 3) an study of the results of the best systems of Senseval-2 that shows the real behaviour of these systems with respect to the data.

Summarising, this work has: 1) studied the application of Machine Learning to Word Sense Disambiguation; and 2) described our participation on the English Lexical Sample task of both Senseval-2 and Senseval-3 international evaluation exercises. This work has clarified several open questions which we think will help to understand this complex problem.

Acknowledgements

We want to specially thank to Victoria Arranz for her help with English writing; to David Martínez for their useful comments and discussions; to Adrià Gispert for its perl scripts; to Yarowsky's group for the feature extractor of syntactical patterns; to the referees of international conferences and of the previous version of this document for their helpful comments; to Victoria Arranz, Jordi Atserias, Laura Benítez and Montse Civit for their friendship; and to Montse Beserán for its infinity patience and love.

We want to also thank to Lluís Màrquez and German Rigau for all these years of dedication; to Horacio Rodríguez for his humanity; to Eneko Agirre for his comments; and to all the members of the TALP Research Centre of the Software Department of the Technical University of Catalonia for working together.

This research has been partially funded by the European Commission via EuroWordNet (LE4003), NAMIC (IST-1999-12392) and MEANING (IST-2001-34460) projects; by the Spanish Research Department via ITEM (TIC96-1243-C03-03), BASURDE (TIC98-0423-C06) and HERMES (TIC2000-0335-C03-02); and by the Catalan Research Department VIA CIRIT's consolidated research group 1999SGR-150, CREL's Catalan WordNet project and CIRIT's grant 1999FI-00773.

Contents

Abstract **i**

Acknowledgements **iii**

1 Introduction **1**

1.1 Usefulness of WSD 2

1.2 Thesis Contributions 3

1.3 Thesis Layout 4

2 State-of-the-Art **7**

2.1 The Word Sense Disambiguation Task 8

2.1.1 Knowledge-based Methods 9

2.1.2 Corpus-based Approach 12

2.1.3 Corpus for training Supervised Methods 12

2.1.4 Porting across corpora 15

2.1.5 Feature selection and parameter optimisation 16

2.2 Supervised Corpus-based Word Sense Disambiguation 18

2.2.1 Word Sense Disambiguation Data 18

2.2.2 Representation of Examples 21

2.2.3 Main Approaches to Supervised WSD 22

2.3 Current Open Issues 28

2.3.1 Bootstrapping Approaches 28
2.4 Evaluation 30
2.4.1 Pre-Senseval evaluations 30
2.4.2 Senseval Evaluation 32
2.5 Further Readings 38

3 A Comparison of Supervised ML Algorithms for WSD 39

3.1 Machine Learning for Classification 40
3.1.1 Naive Bayes 41
3.1.2 Exemplar-based learning 42
3.1.3 Decision Lists 44
3.1.4 AdaBoost 44
3.1.5 Support Vector Machines 48
3.2 Setting 51
3.2.1 Corpora 51
3.2.2 Basic Features 51
3.3 Adapting Naive-Bayes and Exemplar-Based Algorithms 53
3.3.1 Comments about Related Work 53
3.3.2 Description of the Experiments 54
3.3.3 Conclusions 59
3.4 Applying the AdaBoost Algorithm 59
3.4.1 Description of the Experiments 60
3.4.2 Conclusions 65
3.5 Comparison of Machine Learning Methods 65
3.5.1 Setting of Evaluation 66
3.5.2 Discussion 67
3.6 Conclusions 70

4 Domain Dependence **73**

4.1 Introduction . . . 73

4.2 Setting . . . 74

4.3 First Experiment: Across Corpora evaluation . . . 75

4.4 Second Experiment: tuning to new domains . . . 77

4.5 Third Experiment: training data quality . . . 77

4.6 Conclusions . . . 81

5 Bootstrapping **83**

5.1 Transductive SVMs . . . 83

5.1.1 Setting . . . 84

5.1.2 Experiments with TSVM . . . 86

5.2 Greedy Agreement Bootstrapping Algorithm . . . 91

5.2.1 Setting . . . 92

5.2.2 Experimental Evaluation . . . 94

5.3 Conclusions and Current/Further Work . . . 101

6 Evaluation in Senseval Exercises **103**

6.1 Senseval Corpora . . . 103

6.2 TALP System at Senseval-2 . . . 104

6.2.1 Feature Set . . . 105

6.2.2 Preprocessing and Hierarchical Decomposition . . . 108

6.2.3 Evaluation . . . 109

6.3 From Senseval-2 to Senseval-3 . . . 111

6.3.1 System Decomposition . . . 112

6.3.2 The TalpSVM System . . . 116

6.3.3 Global Results . . . 121

6.4 Senseval-3 . . . 121

6.4.1 Learning Framework 123
6.4.2 Features 124
6.4.3 Experimental Setting 124
6.4.4 Evaluation 126
6.4.5 Extending the Feature Selection process 127
6.5 Conclusions 129

7 Conclusions **131**
7.1 Summary 131
7.2 Contributions 133
7.3 Publications 134
7.4 Further Work 135

Bibliography **137**

A Statistical Measures **153**
A.1 Evaluation Measures 153
A.2 Significance Tests 154
A.3 Agreement and Kappa statistic 154

B Example of the Feature Extractor **155**

C Resources and Web References **159**

D List of Acronyms **161**

Chapter 1

Introduction

Since the appearance of the first computers, in the earlier 50's, humans have been thinking in Natural Language Understanding (NLU). Since then, lots of talking computers appeared in fiction novels and films. Humans usually see computers as intelligent devices. However, the pioneers of Natural Language Processing (NLP) underestimated the complexity of the task. Unfortunately, natural language systems seems to require extensive knowledge about the world which in turn it is not easy to acquire. NLP community has been researching on NLU for 50 years and satisfactory results have been obtained only for very restricted domains[1].

It is commonly assumed that NLU is related with semantics. NLP community have applied different ways to tackle semantics, such as logical formalisms or frames to represent the necessary knowledge. The current usual way to tackle NLP is as a process chain, in which each step of the chain is devoted to one Natural Language task; usually trying to solve a particular Natural Language ambiguity problem. Natural language presents many types of ambiguity, ranging from morphological ambiguity to pragmatic ambiguity passing through syntactic or semantic ambiguities. Thus, most efforts in Natural Language Processing are devoted to solve different types of ambiguities, such as: part-of-speech (POS) tagging –dealing with morphosyntactic categories– or parsing –dealing with syntax.

The work presented here focus on *lexical* or *word sense* ambiguity. That is, the ambiguity related to the polisemy of words (isolated words are ambiguous). The determination of the meaning of each word of a context seems to be a necessity in order to understand the language. *Lexical ambiguity* is a long-standing problem in NLP, which appears on early references (Kaplan, 1950; Yngve, 1955) on Machine Translation[2].

[1] See Winograd SHRDLU program (Winograd, 1972).

[2] *Wikipedia* points Word Sense Disambiguation as one of the big difficulties for NLP.

Lexical Ambiguity Resolution or Word Sense Disambiguation (WSD) is the problem of assigning the appropriate meaning (sense) to a given word in a text or discourse where this meaning is distinguishable from other senses potentially attributable to that word (Ide and Véronis, 1998). Thus, a WSD or Word Sense Tagging system must be able to assign the correct sense of a given word, for instance *age*, depending on the context in which the word occurs.

Table 1.1 shows, as an example, two of the senses of noun *age*, their translation into Catalan and two glosses describing their meaning[3]. Sense 1 of word *age* is translated into Catalan as *edat* and sense 2 to the noun *era*, which indicates a real word sense distinction. An NLP system trying to capture the intended meaning of the word *age* in the two sentences in table 1.2 would performing word sense disambiguation[4].

English	**Catalan**	**gloss**
age 1	*edat*	the length of time something (or someone) has existed
age 2	*era*	a historic period

Table 1.1: Definitions of two of the senses of word *age* from WordNet 1.5.

sense	**gloss**
age 1	He was mad about stars at the **age** of nine .
age 2	About 20,000 years ago the last ice **age** ended .

Table 1.2: Examples from DSO corpus.

1.1 Usefulness of WSD

WSD task is a potential *intermediate task* (Wilks and Stevenson, 1996) for many other NLP systems, including mono and multilingual Information Retrieval, Information Extraction, Machine Translation or Natural Language Understanding.

Resolving the sense ambiguity of words is obviously essential for many Natural Language Understanding applications (Ide and Véronis, 1998). However, current NLU applications are mostly domain specific (Kilgarriff, 1997; Viegas et al., 1999). Obviously, when restricting the application to a domain we are, in fact, using the "one sense per discourse" hypothesis to eliminate most of the semantic ambiguity of polysemous words (Gale et al., 1992b).

[3]The English information of this table has been extracted from WordNet 1.5 (Miller et al., 1990).
[4]These sentences have been extracted from the DSO corpus (Ng and Lee, 1996).

Furthermore, the need of enhanced WSD capabilities appears in many applications whose aim is not language understanding. Among others, we could mention:

- *Machine Translation*: This is the field in which the first attempts to perform WSD where carried out (Weaver, 1955; Yngve, 1955; Bar-Hillel, 1960). There is no doubt that some kind of WSD is essential for the proper translation of polysemous words.
- *Information Retrieval*: in order to discard occurrences of words in documents appearing with inappropriate senses (Salton, 1968; Salton and McGill, 1983; Krovetz and Croft, 1992; Voorhees, 1993; Schütze and Pedersen, 1995).
- *Semantic Parsing*: Alshawi and Carter (1994) suggest the utility of WSD in restricting the space of competing parses, especially, for the dependencies, such as prepositional phrases.
- *Speech Synthesis and Recognition*: WSD could be useful for the correct phonetisation of words in Speech Synthesis (Sproat et al., 1992; Yarowsky, 1997), and for word segmentation and homophone discrimination in Speech Recognition (Connine, 1990; Seneff, 1992).
- *Acquisition of Lexical Knowledge*: many approaches designed to automatically acquire large-scale NLP resources such as, selectional restrictions (Ribas, 1995), subcategorisation verbal patterns (Briscoe and Carroll, 1997), translation links (Atserias et al., 1997) have obtained limited success because of the use of limited WSD approaches.
- *Lexicography*: Kilgarriff (1997) suggests that lexicographers can be not only suppliers of NLP resources, but also customers of WSD systems.

Unfortunately, the low accuracy results obtained by current state-of-the-art WSD systems seem to be below practical requirements. Carpuat and Wu (2005) studied the usefulness of WSD to Statistical Machine Translation and, with a few exceptions, no improvement was achieved in their experiments. They also questioned the current models of Statistical MT.

1.2 Thesis Contributions

The main contribution of this thesis is to explore the possible application of algorithms and techniques of the Machine Learning field to the Word Sense Disambiguation. We have summarised below the list of contributions we consider most important from this work:

Comparative study: First, we have performed a comparison of the most used algorithms in the research literature. During this process, we have clarified some confusing information on the literature and developed some improvements on the adaptation of some algorithms (in terms of representation of the information and efficiency). We have empirically demonstrated that the best systems on Word Sense Disambiguation data are those based on margin-maximisation. We have also seen that accuracy, agreement, and Kappa statistics show interesting dimensions when evaluating the systems.

Generalisation across corpora: The experiments of chapter 4 empirically show the dependency of Machine Learning on the training corpora. This issue has to be solved in order to develop a real and useful Word Sense Tagger. Until knowing the way of building full general corpora or flexible training procedures able to adapt to changing objective functions, the only Word Sense Taggers we can build are those based on restricted domains.

Bootstrapping: The experiments of chapter 5 explore two Bootstrapping techniques: Transductive Support Vector Machines and the greedy agreement Abney's algorithm. The first approach does not proved to work properly. However, the Abney's algorithm showed better performance. Initial experiments seem to indicate a promising line of research.

International Evaluation Exercises on WSD: We also have participated on the English Lexical Sample task of both Senseval-2 and Senseval-3 competitions with two different systems obtaining very good results. On Senseval-2 our system got the fifth position at only 4.8% of accuracy from the winner, and on Senseval-3 our system achieved the seventh position at only 1.6% of accuracy. In both cases, our system proved to be competitive with state of the art WSD systems. A complementary work has been developed: an study of the sources of the differences among the top performing systems at Senseval-2.

1.3 Thesis Layout

This section is devoted to overview the contents of the rest of the thesis:

- A survey of the State of the Art on WSD is presented in chapter 2.
- Chapter 3 contains a comparison of the most used Machine Learning algorithms on the Word Sense Disambiguation community. It also reviews other previous

comparison works, and explains how the algorithms should be adapted to deal with the word senses.

- In chapter 4 the cross-corpora application of Machine Learning techniques on Word Sense Disambiguation is empirically studied.
- Chapter 5 analyses the usefulness of two bootstrapping techniques on Word Sense Disambiguation.
- Chapter 6 describes our participation in the English Lexical Sample task of both Senseval-2 and Senseval-3 evaluation exercises. It also shows an study of the best performing systems in Senseval-2.
- Finally, the conclusions of our research are presented in chapter 7.

There are four appendixes in this document. The first one shows the formulae of the statistical measures applied in all the documents for evaluation purposes. The second appendix describes in detail (with an example) the output of the feature extractor resulting of this work. The third appendix contains a list of web references of all resources related WSD. And the final one contains a list of the acronyms used throughout all the text.

Chapter 2

State-of-the-Art

This chapter is devoted to present the state-of-the-art of WSD. Several approaches have been proposed for assigning the correct sense to a word in context[1], some of them achieving remarkable high accuracy figures. Initially, these methods were usually tested only on a small set of words with few and clear sense distinctions, e.g. Yarowsky (1995b) reports 96% precision for twelve words with only two clear sense distinctions each. Despite the wide range of approaches investigated and the large effort devoted to tackle this problem, it is a fact that to date, no large-scale broad-coverage and highly accurate WSD system has been built. It still remains an open problem if we look at the main conclusions of the *ACL SIGLEX Workshop: Tagging Text with Lexical Semantics: Why, What and How?* "WSD is perhaps the great open problem at the lexical level of NLP" (Resnik and Yarowsky, 1997) or to the results of the Senseval (Kilgarriff and Rosenzweig, 2000), Senseval-2 (Kilgarriff, 2001) and Senseval-3 (Mihalcea et al., 2004) evaluation exercises for WSD; in which none of the systems presented in these conferences achieved 80% accuracy on both English Lexical Sample and All Words tasks. Actually, in the last two competitions the best systems achieved accuracy values near 65%.

WSD has been described as an *AI-complete* problem in the literature, that is, "its solution requires a solution to all the general *Artificial Intelligence* (AI) (problems of representing and reasoning about arbitrary real-world knowledge" (Kilgarriff, 1997) or "a problem which can be solved only by first resolving all the difficult problems in AI, such as the representation of common sense and encyclopedic knowledge" (Ide and Véronis, 1998). This fact evinces the hardness of the task and, the large amount of resources needed to tackle the problem.

[1]The most simple approach ignore context and selects for each ambiguous word its most frequent sense (Gale et al., 1992a; Miller et al., 1994).

This chapter has been organised as follows: section 2.1 is devoted to describe the task at hand and the different ways to address this problem. Then, we focus on the corpus-based approach, which is the framework of this thesis. In section 2.2 we extend the corpus-based explanation by showing some WSD data, the main corpus-based approaches and the way WSD data is represented. Section 2.3 is devoted to present a set of promising research lines in the field, and section 2.4 to several evaluation issues. Finally, section 2.5 contains references for extending several issues of the state-of-the-art.

2.1 The Word Sense Disambiguation Task

WSD typically involves two main tasks. On the one hand (1) determining the different possible senses (or meanings) of each word, and, on the other hand (2) tagging each word of a text with its appropriate sense with high accuracy and efficiency.

The former task, that is, the precise definition of a sense, is still under discussion and remains as an open problem within the NLP community. Section 2.2.1 enumerates the most important sense repositories used by the Computational Linguistics Community. At the moment, the most used Sense Repository is WordNet. However, many problems arise mainly related to the granularity of the senses. Contrary to the intuition that the agreement between human annotators should be very high in the WSD task (using a particular sense repository), some papers report surprisingly low figures. For instance, (Ng et al., 1999) reports an agreement rate of 56.7% and a Kappa value of 0.317 when comparing the annotation of a subset of the DSO corpus performed by two independent research groups[2]. Similarly, Véronis (1998) reports values of Kappa near to zero when annotating some special words for the Romanseval corpus[3]. These reports evince the fragility of existing sense repositories. Humans can not agree in the meaning of a word due to the subtle and fine distinctions of the sense repositories, lack of clear guidelines, and tight time constraints and limited experience of the human annotators. Moreover, Senseval-3 has shown the dependence of the sense repositories on the accuracy of the systems. The English Lexical Sample task used as sense repository WordNet and the best system achieved an accuracy of 72.9% (Mihalcea et al., 2004). On the other side, for Catalan and Spanish Lexical Sample tasks, the organisers developed a sense repository

[2]The Kappa statistic k (Cohen, 1960) (see appendix A) is a good measure for inter-annotator agreement because reduces the effect of chance agreement. A Kappa value of 1 indicates perfect agreement, while 0.8 is considered as indicating good agreement (Carletta, 1996).

[3]Romanseval is, like Senseval for English, a specific competition between WSD systems for Romance languages at the moment of first Senseval event. In Senseval-2 and 3 events, Romanseval tasks have become part of Senseval.

based on coarser grained senses and the best systems achieved respectively 85.8% and 84.2% (Màrquez et al., 2004b,a). However, the discussion on the most appropriate sense repository for WSD is beyond the scope of the work presented here.

The second task, the tagging of each word with a sense, involves the development of a system capable of tagging polysemic words in running text with sense labels[4]. The WSD community accepts a classification of these systems in two main general categories: *knowledge-based* and *corpus-based* methods. All methods build a representation of the examples to be tagged using some previous information. The difference between them is the source of this information. *Knowledge-based* methods obtain the information from external knowledge sources such as Machine Readable Dictionaries (MRDs) or lexico-semantic ontologies. This knowledge exists previous to the disambiguation process, and usually have been manually generated (probably not taking into account its intended use when developing them). On the contrary, in corpus-based methods the information is gathered from contexts of previously annotated instances (examples) of the word. These methods extract the knowledge from examples applying Statistical or Machine Learning methods. When these examples are previously hand-tagged we talk about *supervised learning*, while if the examples do not come with the sense label we talk about *unsupervised learning*. *Supervised Learning* involves a large amount of semantically annotated training examples labelled with their correct senses (Ng and Lee, 1996). On the contrary, Knowledge-based systems do not require the development of training corpora (Rigau et al., 1997)[5]. Our work focus on the supervised Corpus-based framework, although we briefly discus knowledge-based methods in the following section.

2.1.1 Knowledge-based Methods

These methods mainly try to avoid the need of large amounts of training materials required in supervised methods. Knowledge-based methods can be classified in function of the type of resources they use: 1) Machine-Readable Dictionaries; 2) Thesauri; or 3) Computational Lexicons or Lexical Knowledge Bases.

- **Machine-Readable Dictionaries (MRDs)** provide a ready-made source of information about word senses and knowledge about the world, which could be very useful for WSD and NLU. Since the work by Lesk (1986) many researchers have used MRDs as structured source of lexical knowledge for WSD systems. However,

[4]Obviously, for monosemous words, content words having a unique sense in the sense repository, the task is trivial.

[5]This system uses Machine-Readable Dictionaries available as external knowledge resources.

MRDs contain inconsistencies and are created for human use, and not for machine exploitation. There is a lot of knowledge in a dictionary only really useful when performing a complete WSD process on the whole definitions (Richardson, 1997; Rigau, 1998; Harabagiu and Moldovan, 1998). See also the results of the Senseval-3 task devoted to disambiguate WordNet glosses (Castillo et al., 2004).

WSD techniques using MRDs can be classified according to: the lexical resource used (mono or bilingual MRDs); the MRD information exploited by the method (words in definitions, semantic codes, etc); and the similarity measure used to relate words from context and MRD senses.

Lesk (1986) created a method for guessing the correct word sense counting word overlaps between the definitions of the word and the definitions of the context words, and selecting the sense with the greatest number of overlapping words. Although this method is very sensitive to the presence or absence of the words in the definition, it has served as the basis for most of the subsequent MRD-based disambiguation systems. Among others, Guthrie et al. (1991) proposed the use of the subject semantic codes of the *Longman Dictionary of Contemporary English* (LDOCE) to improve the results. Cowie et al. (1992) improved Lesk's method by using the Simulated Annealing algorithm. Wilks et al. (1993) and Rigau (1998) used the co-occurrence data extracted from LDOCE and *Diccionario General de la Lengua Española* (DGILE), respectively, to construct word-context vectors.

- **Thesauri** provide information about relationships among words, specially synonymy. Like MRDs, a thesaurus is a resource created for humans and, therefore, is not a source of perfect information about word relations. However, thesauri provide a rich network of word associations and a set of semantic categories potentially valuable for large-scale language processing. Roget's International Thesaurus is the most used thesaurus for WSD. It classifies 60,071 words into 1,000 semantic categories (Yarowsky, 1992; Grefenstette, 1993; Resnik, 1995).

  Yarowsky (1992) used each occurrence of the same word under different categories of a thesaurus as representations of the different senses of that word. The resulting classes are used to disambiguate new occurrences of a polysemous word. Yarowsky notes that his method concentrates on extracting topical information.

- In the late 80s and throughout 90s, a large effort have been carried out on developing manually large-scale **knowledge bases**: WordNet (Miller et al., 1990), CyC (Lenat and Ramanathan, 1990), ACQUILEX (Briscoe, 1991), and Mikrokosmos (Viegas et al., 1999) are examples of such resources. Currently, WordNet is the best-known and the most used resource for WSD in English. In WordNet the concepts are

defined as synonymy sets called *synsets* linked one to each other through semantic relations (hyperonymy, hyponymy, meronymy, antonymy, and so on). Each sense of a word is linked to a *synset*.

Considering WordNet as a reference, new WordNets of other languages were started within the EuroWordNet framework (Dutch, Italian, Spanish, French, and German)[6]. In this framework all these WordNets are interconnected to the English WordNet by means of an interlingual index.

Taking the WordNet structure as source, some methods using semantic distance metrics have been developed. Most of these metrics consider only nouns. Sussna's metric (Sussna, 1993) consists of computing the distance as a function of the length of the shortest path between nodes (word senses). It is interesting because he used many relations and not only the hyperonymy-hyponymy relation. *Conceptual Density*, a more complex semantic distance measure between words is defined in (Agirre and Rigau, 1995) and tested on the *Brown Corpus*, as a proposal for WSD in (Agirre and Rigau, 1996). Viegas et al. (1999) illustrate the Mikrokosmos approach to WSD applying an ontological graph search mechanism, *Onto-Search*, to check constraints.

Magnini et al. (2001) have studied the influence of the domains in WSD, using WordNet Domains (Magnini and Cavaglia, 2000). The main aim of the work is to reduce the polisemy of words by selecting first the domain of the text. They obtained a high precision but low recall at Senseval-2 English Lexical Sample Task (see section 2.4). At Senseval-3, domains were used as a component of their complete system including among others a corpus-based module obtaining very good results.

Knowledge-based systems are gaining importance in the All Words tasks of last Senseval events to tag words with a low number of training examples. Most systems combine both knowledge-based and corpus-based approaches when all-words data have to be processed. They learn those words for which labelled examples are available; and, apply an unsupervised approach for those words without enough training examples.

The information used as input for corpus-based methods is being called knowledge sources or sources of information by the community (Lee et al., 2004). Some authors argue that knowledge bases can be seen as sources of information for representing example features (see as examples section 6.4 and appendix B).

[6]Currently, several national and international projects are funding the construction and improvement of WordNets (See Global WordNet Association web page for a complete list of developed WordNets).

2.1.2 Corpus-based Approach

Corpus-based approaches are those that build a classification model from examples. These methods involve two phases: *learning* and *classification*. The learning phase consists of learning a sense classification model from the training examples. The classification process consists of the application of this model to new examples in order to assign the output senses. Most of the algorithms and techniques to build models from examples come from the Machine Learning area of AI. Section 2.2.3 enumerates some of these algorithms that have been applied to WSD; and section 2.2.1 enumerates the most important corpora used by the WSD community.

One of the first and most important issues to take into account is the representation of the examples by means of features/attributes. That is, which information could and should be provided to the learning component from the examples. The representation of examples highly affects the accuracy of the systems. It seems to be as or more important than the learning method used by the system. Section 2.2.2 is devoted to discuss the most common example representation appearing in the literature.

2.1.3 Corpus for training Supervised Methods

Right-sized training sets for WSD

So far, corpus-based approaches have obtained the best absolute results. However, it is well known that supervised methods suffer from the lack of widely available semantically tagged corpora, from which to construct really broad coverage WSD systems. This is known as the "knowledge acquisition bottleneck" (Gale et al., 1993). Ng (1997b) estimated that to obtain a high accuracy domain-independent system at least 3,200 words should be tagged with about 1,000 occurrences each. The necessary effort for constructing such a training corpus is estimated to be 16 person-years, according to the experience of the authors on the building of the DSO corpus (see section 2.2.1).

Unfortunately, many people think that Ng's estimate might fall short, as the annotated corpus produced in this way is not guaranteed to enable high accuracy WSD. In fact, recent studies using DSO have shown that: 1) The performance for state of the art supervised WSD systems continues to be in the 60%-70% for this corpus (see chapter 4), and 2) Some highly polysemous words get very low performance (20-40% accuracy).

There have been some works exploring the learning curves of each different word to investigate the amount of training data required. In (Ng, 1997b), the exemplar-based learning LEXAS supervised system was trained for a set of 137 words with at least 500

examples, and for a set of 43 words with at least 1,300 examples. In both situations, the accuracy of the system in the learning curve was still rising with the whole training data. In an independent work (Agirre and Martínez, 2000), the learning curves of two small sets of words (containing nouns, verbs, adjectives, and adverbs) were studied using different corpora (SemCor and DSO). Words of different types were selected, taking into account their characteristics: high/low polysemy, high/low frequency, and high/low skew of the most frequent sense in SemCor. The results reported, using Decision Lists as the learning algorithm, showed that SemCor data is not enough, but that in the DSO corpus the result seemed to stabilise for nouns and verbs before using all the training material. The word set tested in DSO had in average 927 examples per noun, and 1,370 examples per verb.

Selection of training examples

Another important issue is that of the selection and quality of the examples. In most of the semantically tagged corpora available it is difficult to find a minimum number of occurrences per each sense of a word. In order to overcome this particular problem, also known as "knowledge acquisition bottleneck", four main lines of research are currently being pursued: 1) Automatic acquisition of training examples; 2) Active learning; 3) Learning from labelled and unlabelled examples; and 4) Combining training examples from different words.

- In *automatic acquisition of training examples*, an external lexical source, for instance WordNet, or a seed sense-tagged corpus is used to obtain new examples from an untagged very large corpus (or the web).

  Leacock et al. (1998) used a pioneering knowledge-based technique to automatically extract training examples for each sense from the Internet. WordNet is used to locate monosemous words semantically related to those word senses to be disambiguated (monosemous relatives).

  Following this approach, Mihalcea and Moldovan (1999b) used more information from WordNet (e.g., monosemous synonyms and glosses) to construct queries, which were later fed into the Altavista web search engine. Four procedures were used sequentially, in a decreasing order of precision, but with increasing levels of retrieved examples. Results were evaluated by hand, finding out that over 91% of the examples were correctly retrieved among a set of 1,080 instances of 120 word senses. However, the number of examples acquired did not have to correlate with the frequency of senses. Agirre and Martínez (2004c) trained a WSD system with this technique showing its utility.

In Mihalcea (2002a), a sense tagged corpus (GenCor) is generated using a set of seeds, consisting of sense tagged examples from four sources: SemCor (see section 2.2.1), WordNet examples created with the method described in (Mihalcea and Moldovan, 1999b), and hand-tagged examples from other sources (e.g., Senseval-2 corpus). A corpus with about 160,000 examples was generated from these seeds. A comparison of the results obtained by their WSD system, when training with the generated corpus or with the hand-tagged data provided in Senseval-2, was reported. She concluded that the precision achieved using the generated corpus is comparable, and sometimes better, than learning from hand tagged examples. She also showed that the addition of both corpora further improved the results. Their method has been tested in the Senseval-2 framework with remarkable results.

- *Active learning* is used to choose informative examples for hand tagging, in order to reduce the acquisition cost. Argamon-Engelson and Dagan (1999) describe two main types of active learning: *membership queries* and *selective sampling*. In the first approach, the learner constructs examples and asks a teacher to label them. This approach would be difficult to apply to WSD. Instead, in selective sampling the learner selects the most informative examples from unlabelled data. The informativeness of the examples can be measured using the amount of uncertainty in their classification, given the current training data. Lewis and Gale (1994) use a single learning model and select those examples for which the classifier is most uncertain (*uncertainty sampling*). Argamon-Engelson and Dagan (1999) propose another method, called *committee-based sampling*, which randomly derives several classification models from the training set, and the degree of disagreement between them is used to measure the informativeness of the examples. For building a verb database, Fujii et al. (1998) applied selective sampling to the disambiguation of verb senses, in base to their case fillers. The disambiguation method was based on nearest neighbour classification, and the selection of examples in the notion of *training utility*, which has two criteria: *number of neighbours* in unsupervised data (i.e., examples with many neighbours will be more informative in next iterations), and *dissimilarity* of the example with other supervised examples (to avoid redundancy). A comparison of their method with uncertainty and committee-based sampling was reported, obtaining significantly better results on the learning curve.

  Open Mind Word Expert (Chklovski and Mihalcea, 2002), is a project to collect word sense tagged examples from web users. They select the examples to be tagged applying a selective sampling method. Two different classifiers are independently applied on untagged data: an instance-based classifier that uses active feature selection, and a constraint-based tagger. Both systems have a low inter-annotation

agreement (54.96%), high accuracy when they agree (82.5%), and low accuracy when they disagree (52.4% and 30.09%, respectively). This makes the disagreement cases the hardest to annotate, being the ones that are presented to the user.

- Some methods have been devised for *learning from labelled and unlabelled data*, which are also referred to as bootstrapping methods (Abney, 2002). Among them, we can highlight *co-training* (Blum and Mitchell, 1998) and their derivates (Collins and Singer, 1999; Abney, 2002). These techniques seem to be very appropriate for WSD and other NLP tasks, because of the wide availability of untagged data, and the scarcity of tagged data. In a well-known work in the WSD field, (Yarowsky, 1995b; Abney, 2004) exploited some discourse properties (see section 2.2.3) in an iterative bootstrapping process, to induce a classifier based on Decision Lists. With a minimum set of seed (annotated) examples, the system obtained comparable results to those obtained by supervised methods in a limited set of binary sense distinctions. Lately, this kind of methods are gaining importance. Section 2.3.1 is devoted to explain them in more detail.

- Recent works build classifiers for semantic classes, instead of word classifiers. Kohomban and Lee (2005) build semantic classifiers by merging training examples from words in the same semantic class obtaining good results in the application of their methods on Senseval-3 data.

2.1.4 Porting across corpora

Porting the WSD systems to new corpora of different genre/domains also presents important challenges. Some studies show that, in practice, the assumptions for supervised learning do not hold when using different corpora, even when there are many training examples available. There is a dramatic degradation of performance when training and testing on different corpora. Sense uses depend very much of the domains. That is why corpora should be large and diverse, covering many domains and topics in order to provide enough examples for each of the senses.

Chapter 4 is devoted to the study of the performance of four ML algorithms (Naive Bayes, Exemplar-based learning, Decision Lists and AdaBoost) when tested on a different corpus than the one used for training, and explores their ability to be adapted to new domains. We used the DSO corpus for the experiments. This corpus contains sentences from the Wall Street Journal corpus (WSJ, financial domain) and from the Brown Corpus (BC, balanced). We carried out three experiments to test the portability of the algorithms. For the first and second experiments, we collected the sentence examples from WSJ and

BC forcing the same number of examples per sense in both sets. The results obtained when training and testing across corpora were disappointing for all ML algorithms tested, since significant decreases in performance were observed in all cases (in some of them the cross-corpus accuracy was even lower than the "most frequent sense" baseline). The incremental addition of a percentage of supervised training examples from the target corpus did not help very much to raise accuracy of the systems. In the best case, the accuracy achieved was only slightly better than the one obtained by training on the small supervised part of the target corpus, making no use of the whole set of examples from the source corpus. The third experiment showed that WSJ and BC have very different sense distributions and that relevant features acquired by the ML algorithms are not portable across corpora, since in some cases they correlated with different senses in different corpora.

In (Martínez and Agirre, 2000), the main reason for the low performance in cross-corpora tagging was also attributed to the change in domain and genre. Again, they used the DSO corpus and a disjoint selection of the sentences from the WSJ and BC parts. In BC, texts are classified according to some predefined categories (Reportage, Religions, Science Fiction, etc.). This fact allowed them to test the effect of the domain and genre on cross-corpora sense tagging. Reported experiments, training on WSJ and testing on BC and vice versa, showed that the performance dropped significantly from the results on each corpus separately. This happened mainly because there were few common collocations (features were mainly based on collocations), and also because some collocations received systematically different tags in each corpus –a similar observation to that of (Escudero et al., 2000b). Subsequent experiments were conducted taking into account the category of the documents of the BC, showing that results were better when two independent corpora shared genre/topic than when using the same corpus with different genre/topic. The main conclusion is that the "one sense per collocation" constraint does hold across corpora, but that collocations vary from one corpus to another, following genre and topic variations. They argued that a system trained on a specific genre/topic would have difficulties to adapt to new genres/topics. Besides, methods that try to extend automatically the amount of examples for training should also take into account genre and topic variations.

2.1.5 Feature selection and parameter optimisation

Another current trend in WSD and in Machine Learning in general is the *automatic selection of features* (Hoste et al., 2002b; Daelemans and Hoste, 2002; Decadt et al., 2004). The previous feature selection should be needed for: 1) computational issues; 2) some learning algorithms are very sensitive to non relevant or redundant features; and 3) the addition of new attributes not necessary improves the performance.

Some recent works have focused on defining separate feature sets for each word, claiming that different features help to disambiguate different words. For instance, Mihalcea (2002b) applied exemplar-based algorithm to WSD, an algorithm very sensitive to irrelevant features. In order to overcome this problem she used a *forward selection* iterative process to select the optimal features for each word. She ran cross-validation on the training set, adding the best feature to the optimal set at each iteration, until no improvement was observed. The final system achieved very competitive results in the Senseval-2 competition (see section 2.4.2).

Interesting research has been conducted on parameter optimisation of machine learning algorithms for Word Sense Disambiguation. Hoste et al. (2002b) observed that although there exists some comparisons between Machine Learning algorithms trying to determine the best method for WSD, there are large variations on performance depending on three factors: algorithm parameters, input representation, and interaction between both. These observations question the validity of the results of the comparisons. They claim that changing any of these dimensions produces large fluctuations in accuracy, and that an exhaustive optimisation of parameters is required in order to obtain reliable results. They argue that there is little understanding of the interaction between these three influential factors, and while no fundamental data-independent explanation is found, data-dependent cross-validation can provide useful clues for WSD. In their experiments, they show that memory-based WSD benefits from an optimised architecture, consisting of information sources and algorithmic parameters. The optimisation is carried out using cross-validation on the learning data for each word. In order to tackle this optimisation problem, due to the computational cost of the search, they use Genetic Algorithms (Daelemans and Hoste, 2002; Decadt et al., 2004). They obtained good results in both Senseval-3 English All-words and in English Lexical Sample tasks.

Martínez et al. (2002) make use of feature selection for high precision disambiguation at the cost of coverage. By using cross-validation on the training corpus, a set of individual features with a discriminative power above a certain threshold was extracted for each word. The threshold parameter allows to adjust the desired precision of the final system. This method was used to train decision lists, obtaining 86% precision for 26% coverage, or 95% precision for 8% coverage on the Senseval-2 data. However, the study do not provided an analysis of the senses for which the systems performed correctly. One potential use of a high precision system is the acquisition of almost error-free new examples in a bootstrapping framework.

Finally, Escudero et al. (2004) achieves also good results on Senseval-3 English Lexical Sample task by performing per-word feature selection taking as input an initial feature set obtained by a per-POS feature selection over the Senseval-2 English Lexical Sample

corpus (see section 6.4 for a detailed information).

2.2 Supervised Corpus-based Word Sense Disambiguation

2.2.1 Word Sense Disambiguation Data

Main Sense Repositories

Initially, Machine Readable Dictionaries (MRDs) were used as main repositories of word sense distinctions to annotate word examples with senses. For instance, LDOCE, Logman Dictionary of Contemporary English (Procter, 1973) was frequently used as a research lexicon (Wilks et al., 1993) and for tagging word sense usages (Bruce and Wiebe, 1999).

In the first Senseval edition, in 1998, the English lexical-sample task used the HECTOR dictionary to label each sense instance. The Oxford University Press and DEC dictionary research project jointly produced this dictionary. However, WordNet (Miller et al., 1990; Fellbaum, 1998) and EuroWordNet (Vossen, 1998) are nowadays becoming the common knowledge sources for sense discriminations.

WordNet is a Lexical Knowledge Base of English. It was developed by the Cognitive Science Laboratory at Princeton University under the direction of Professor George A. Miller. The version 1.7.1 (year 2003) contains information of more than 129,000 words which are grouped in more than 99,000 synsets (concepts or synonym sets). Synsets are structured in a semantic network with multiple relations, being the most important the hyponymy relation (class/subclass). WordNet includes most of the characteristics of a MRD, since it contains definitions of terms for individual senses like in a dictionary. It defines sets of synonymous words that represent a unique lexical concept, and organises them in a conceptual hierarchy similar to a thesaurus. WordNet includes also other types of lexical and semantic relations (meronymy, antonymy, etc.) that provide the largest and richest freely available lexical resource. WordNet was designed to be used by programs; therefore, it does not have many of the associated problems of MRDs (Rigau, 1998).

Many corpora have been annotated using WordNet and EuroWordNet. Since version 1.4 up to 1.6., Princeton provides also SemCor (Miller et al., 1993)[7] (see next section for description of main corpora for WSD). DSO is annotated using a slightly modified version of WordNet 1.5, the same version used for the "*line*, *hard* and *serve*" corpora. The Open Mind Word Expert initiative uses WordNet 1.7. The English tasks of Senseval-2

[7]Rada Mihalcea automatically created SemCor 1.7a from SemCor 1.6 by mapping WordNet 1.6 into WordNet 1.7 senses.

were annotated using a preliminary version of WordNet 1.7 and most of the Senseval-2 non-English task were labelled using EuroWordNet. Although using different WordNet versions can be seen as a problem for the standardisation of these valuable lexical resources, successful methods have been achieved for providing compatibility across the European wordnets and the different versions of Princeton (Daudé et al., 1999, 2000, 2001).

Main Corpora Used

Supervised Machine Learning algorithms use semantically annotated corpora to induce classification models for deciding which is the appropriate word sense for each particular context. The compilation of corpora for training and testing such systems require a large human effort since all the words in these annotated corpora have to be manually tagged by lexicographers with semantic classes taken from a particular lexical semantic resource -most commonly WordNet (Miller et al., 1990; Fellbaum, 1998).

Supervised methods suffer from the lack of widely available semantically tagged corpora, from which to construct really broad coverage systems. And the lack of annotated corpora is even worst for languages other than English. This extremely high overhead for supervision (all words, all languages) explain why the first attempts of using statistical techniques for WSD were designed trying to avoid the manual annotation of a training corpus. This was achieved by using pseudo-words (Gale et al., 1992a), aligned bilingual corpus (Gale et al., 1993) or by working with the related problem of word form restoration (Yarowsky, 1994).

Then, the first systems used bilingual corpus aligned at a word level. These methods rely on the fact that different senses from a word in a given language are translated using different words in another language. For example, the Spanish word "*partido*" translates to "match" in English within the SPORT sense and to "party" within the POLITICAL sense. Therefore, if a corpus is available with a word-to-word alignment, when a translation of a word like "*partido*" is made, its English sense is automatically determined as "match" or "party". Gale et al. (1993) used an aligned French and English corpus for applying statistical WSD methods with a precision of 92%. Working with aligned corpora has the obvious limitation that the learned models are able to distinguish only those senses that are translated into different words in the other language.

The pseudo-words technique is very similar to the previous one. In this method, artificial ambiguities are introduced in untagged corpora. Given a set of words, for instance {"match", "party"}, a pseudo−word corpus can be created collecting all the examples for both words maintaining as labels the original words (which act as senses). This technique is also useful for acquiring training corpora for the accent restoration problem. In this

case, the ambiguity corresponds to the same word having or not accent, like the Spanish words {"*cantara*", "*cantará*"}.

The DSO corpus (Ng and Lee, 1996) is the first medium-big size corpus that was *really* semantically annotated. It contains 192,800 occurrences of 121 nouns and 70 verbs, corresponding to a subset of the most frequent and ambiguous English words. These examples, consisting of the full sentence in which the ambiguous word appears, are tagged with a set of labels corresponding, with minor changes to the senses of WordNet 1.5 –some details in (Ng et al., 1999). Ng and colleagues from the University of Singapore compiled this corpus in 1996 and, since then, it has been widely used. It is currently available from the Linguistic Data Consortium. The DSO corpus contains sentences from two different corpora, namely Wall Street Journal (WSJ) and Brown Corpus (BC). The former focused on financial domain and the second being a general corpus.

Apart from the DSO corpus, there is another major sense-tagged corpora available for English, SemCor (Miller et al., 1993), which stands for Semantic Concordance. It is available from the WordNet web site. Texts that were used to create SemCor were extracted from the Brown corpus (80%) and a novel, *The Red Badge of Courage* (20%), and then manually linked to senses from the WordNet lexicon. The Brown Corpus is a collection of 500 documents, which are classified into fifteen categories. The SemCor corpus makes use of 352 out of the 500 Brown Corpus documents. In 166 of these documents only verbs are annotated (totalising 41,525 occurrences). In the remaining 186 documents all open-class words (nouns, verbs, adjective and adverbs) are linked to WordNet (for a total of 193,139 occurrences). For an extended description of the Brown Corpus see (Francis and Kucera, 1982).

Several authors have also provided the research community with the corpora developed for their experiments. This is the case of the "*line, hard* and *serve*" corpora with more than 4,000 examples per word (Leacock et al., 1998). In this cases, the sense repository was WordNet 1.5 and the text examples were selected from the Wall Street Journal, the American Printing House for the Blind, and the San Jose Mercury newspaper. Another sense tagged corpus is the "*interest*" corpus with 2,369 examples coming from the Wall Street Journal and using the LDOCE sense distinctions.

Furthermore, new initiatives like the Open Mind Word Expert (Chklovski and Mihalcea, 2002) appear to be very promising. This system makes use of the Web technology to help volunteers to manually annotate sense examples. The system includes an active learning component that automatically selects for human tagging those examples that were most difficult to classify by the automatic tagging systems. The corpus is growing daily and, nowadays, contains more than 70,000 instances of 230 words using WordNet 1.7 for sense distinctions. In order to ensure the quality of the acquired examples, the

system requires redundant tagging. The examples are extracted from three sources: Penn Treebank corpus, Los Angeles Times collection (as provided for the TREC conferences), and Open Mind Common Sense. While the two first sources are well known, the Open Mind Common Sense provides sentences that are not usually found in current corpora. They consist mainly in explanations and assertions similar to glosses of a dictionary, but phrased in less formal language, and with many examples per sense. The authors of the project suggest that these sentences could be a good source of keywords to be used for disambiguation. They also propose a new task for Senseval-4 based on this resource.

Finally, resulting from Senseval competitions small sets of tagged corpora have been developed for several languages including: Basque, Catalan, Chinese, Czech, English, Estonian, French, Italian, Japanese, Portuguese, Korean, Romanian, Spanish, and Swedish. Most of these resources and/or corpora are available at the Senseval Web page[8].

2.2.2 Representation of Examples

Before applying any ML algorithm, all the sense examples of a particular word have to be codified in a way that the learning algorithm can handle them. The most usual way of codifying training examples is as feature vectors. In this way, they can be seen as points in an n dimensional feature space, where n is the total amount of features used.

Features try to capture information and knowledge about the *context* and the target words to be disambiguated. However, they necessarily codify only a simplification (or generalisation) of the word sense examples.

In principle this preprocessing step, in which each example is converted into a feature vector, can be seen as an independent process with respect the ML algorithm to be used. However, there are strong implications between the kind and codification of the features and the appropriateness to each learning algorithm (e.g., exemplar-based learning is very sensitive to irrelevant features, decision tree induction does not handle properly attributes with many values, etc.). In 3.3 it is discussed how the feature representation affects both to the efficiency and accuracy of two learning systems for WSD. See also (Agirre and Martínez, 2001) for a survey on the types of knowledge sources that could be relevant for codifying training examples. Also, the WSD problem (as well as other NLP tasks) have some properties, which become very important when considering the application of ML techniques: large number of features (thousands); both, the learning instances and the target concept to be learned, reside very sparsely in the feature space[9]; presence of many

[8] `http://www.senseval.org`

[9] In (Agirre et al., 2005), we find an example of recent work dealing with the sparseness of data by means of combining classifiers with different feature spaces.

irrelevant and highly dependant features; and presence of many noisy examples.

The feature sets most commonly used in the WSD literature can be grouped as follows:

Local features, representing the local context of a word usage. The local context features comprise bigrams and trigrams of word forms, lemmas, POS tags, and their positions with respect the target word. Sometimes, local features include also a bag-of-words (or lemmas) in a small window around the target word (the position of these words is not taken into account). These features are able to capture knowledge about collocations, argument-head relations and limited syntactic cues.

Topic features, representing more general contexts (wide windows of words, other sentences, paragraphs, documents), usually as a bag-of-words.

Syntactic dependencies, at a sentence level, have also been used trying to better model the syntactic behaviour and argument-head relations.

Some authors propose to add some other kind of features. In chapter 6, as an example, it has been added features generated from domain labels extracted from different knowledge sources (SUMO, WordNet Domains, WordNet Semantic Files, and EuroWordNet Top Ontology) from the Multilingual Central Repository (Atserias et al., 2004) of the Meaning Project[10].

Stevenson and Wilks (2001) propose also the combination of different linguistic knowledge sources. They integrate the answers of three partial taggers based on different knowledge sources in a feature-vector representation for each sense. The vector is completed with information about the sense (including rank in the lexicon), and simple collocations extracted from the context. The partial taggers apply the following knowledge: (i) Dictionary definition overlap, optimised for all-words by means of simulated annealing; (ii) Selectional preferences based on syntactic dependencies and LDOCE codes; and (iii) Subject codes from LDOCE using the algorithm by Yarowsky (1992).

2.2.3 Main Approaches to Supervised WSD

We can classify the supervised methods according to the induction principle they use for acquiring the classification models or rules from examples. The following classification does not aim to be exhaustive or unique. Of course, the combination of many paradigms is another possibility that has been applied recently.

[10] http://www.lsi.upc.es/∼meaning

Methods Based on Probabilistic Models

Statistical methods usually estimate a set of probabilistic parameters that express the conditional probability of each category given in a particular context (described as features). Then, these parameters can be combined in order to assign the set of categories that maximises its probability on new examples.

The Naive Bayes algorithm (Duda and Hart, 1973) is the simplest algorithm of this type, which uses the Bayes rule and assumes the conditional independence of features given the class label. It has been applied to many investigations in WSD with considerable success (Gale et al., 1992b; Leacock et al., 1993; Pedersen and Bruce, 1997; Escudero et al., 2000d; Yuret, 2004). Its main problem is the independence assumption. Bruce and Wiebe (1999) present a more complex model known as "decomposable model" which considers different characteristics dependent to each other. The main drawback of this approach is the enormous amount of parameters to be estimated, because they are proportional to the number of different combinations of the interdependent characteristics. Therefore, this technique requires a great quantity of training examples so as to appropriately estimate all the parameters. In order to solve this problem, Pedersen and Bruce (1997) propose an automatic method for identifying the optimal model (high performance and low effort in parameter estimation), by means of the iterative modification of the complexity degree of the model. Despite its simplicity, Naive Bayes is claimed to obtain state-of-the-art accuracy on supervised WSD in many papers (Mooney, 1996; Ng, 1997a; Leacock et al., 1998; Yuret, 2004).

The Maximum Entropy approach (Berger et al., 1996) provides a flexible way to combine statistical evidences from many sources. The estimation of probabilities assumes no prior knowledge of data and it has proven to be very robust. It has been applied to many NLP problems and it also appears as a competitive alternative in WSD (Suárez and Palomar, 2002; Suárez, 2004).

Methods based on the similarity of the examples

The methods in this family perform disambiguation by taking into account a similarity metric. This can be done by comparing new examples to a set of prototypes (one for each word sense) and assigning the sense of the most similar prototype, or by searching into a base of annotated examples which are the most similar.

There are many forms to calculate the similarity between two examples. Assuming the Vector Space Model (VSM), one of the simplest similarity measures is to consider the angle that both example vectors form. Schütze (1992) applied this model codifying each word

of the context with a word vector representing the frequency of its collocations. In this way, each target word is represented with a vector, calculated as the sum of the vectors of the words that are related to the words appearing in the context. Leacock et al. (1993) compared VSM, neural nets, and Naive Bayes methods, and drew the conclusion that the two first methods slightly surpass the last one in WSD. Yarowsky et al. (2001) built a system consisted of the combination of up to six supervised classifiers, which obtained very good results in Senseval-2. One of the included systems was VSM. For training it, they applied a rich set of features (including syntactic information), and weighting of feature types (Agirre et al., 2005).

Another representative algorithm of this family and the most widely used is the k-Nearest Neighbour (kNN) algorithm. In this algorithm the classification of a new example is performed by searching the set of the k most similar examples (or *nearest neighbours*) among a pre-stored set of labelled examples, and selecting the most frequent sense among them, in order to make the prediction. In the simplest case, the training step stores all the examples in memory (this is why this technique is called Memory-based, Exemplar-based, Instance-based, or Case-based learning) and the generalisation is postponed until a new example is being classified (this is why sometimes is also called lazy learning). A very important issue in this technique is the definition of an appropriate similarity (or distance) metric for the task, which should take into account the relative importance of each attribute and should be efficiently computable. The combination scheme for deciding the resulting sense among the k nearest neighbours also leads to several alternative algorithms.

First works on kNN for WSD were developed by (Ng and Lee, 1996) on the DSO corpus. Lately, Ng (1997a) automatically identified the optimal value of k for each word improving the previously obtained results. Section 3.3 of this document focuses on certain contradictory results in the literature regarding the comparison of Naive Bayes and kNN methods for WSD. The kNN approach seemed to be very sensitive to the attribute representation and to the presence of irrelevant features. For that reason alternative representations were developed demonstrating to be more efficient and effective. The experiments demonstrated that kNN was clearly superior to NB when applying adequate feature representation together with feature and example weighting, and quite sophisticated similarity metrics (see chapter 3). Hoste et al. (2002a) also used a kNN system in the *English all words* task of Senseval-2 (Hoste et al., 2001) and won Senseval-3 (Decadt et al., 2004) (see section 2.4). The system was trained on the SemCor corpus, combining different types of ML algorithms like Memory-based learning (TiMBL) and rule induction (Ripper), and used several knowledge sources[11]. Mihalcea and Faruque (2004)

[11] See appendix C for Web references of both the resources and the systems.

obtained very good results also in Senseval-3 English all words task with a very interesting system that includes kNN as its learning component. Daelemans et al. (1999) provide empirical evidence about the appropriateness of Memory-based learning for general NLP problems. They show the danger of dropping exception in generalisation processes. But, Memory-based learning has the known drawback that is very sensitive to feature selection. In fact, Decadt et al. (2004) devote a lot of effort to perform feature selection applying Genetic Algorithms.

Methods based on discursive properties

The methods based on corpus can exploit several discourse properties for WSD: "one sense per discourse" (Gale et al., 1992b), "one sense per collocation" and "attribute redundancy" (Yarowsky, 1995b) properties:

- **One sense per discourse.** The occurrences of a same word in a discourse (or document) usually denote the same sense. For instance, having a business document, it is more likely that the occurrences of the word "company" in the document will denote <business institution> instead of <military unit>.
- **One sense per collocation.** There are certain word senses that are completely determined by means of a collocation. For example, the word "party" in the collocation "Communist Party" unambiguously refers to the *political* sense.
- **Attribute redundancy.** Although language is highly ambiguous, it is also highly redundant. Thus, sophisticated ML algorithms can be devised in order to: 1) Learning the main features of a subset of training examples; 2) Use the induced models to label new examples; and 3) Apply a new learning phase in this new corpus that will be able to capture new characteristics not included in the first training set. This is the base for constructing systems from scratch in a semi-supervised fashion.

Yarowsky (1995b) applied these three properties jointly in an unsupervised WSD algorithm. This approach almost excludes the manual supervision by means of the automatic acquisition of training data from an initial set of seed words. With them, an iterative and incremental process of re-training begins. This algorithm looks very effective, achieving also a high precision in a limited framework. Nevertheless, Martínez and Agirre (2000) made a set of experiments to verify the first two hypotheses in a domain with highly polysemous words obtaining far lower results.

Methods based on discriminating rules

These methods acquire selective rules associated to each word sense. Given a polysemous word, the system selects the sense that verifies some of the rules that determine one of the senses.

- **Methods based on Decision Lists.** Decision Lists are ordered lists of rules of the form (*condition, class, weight*). According to (Rivest, 1987) Decision Lists can be considered as weighted "if-then-else" rules where highly discriminant conditions appear at the beginning of the list (high weights), the general conditions appear at the bottom (low weights), and the last condition of the list is a *default* accepting all remaining cases. Weights are calculated with a scoring function describing the association between the condition and the particular class, and they are estimated from the training corpus. When classifying a new example, each rule in the list is tested sequentially and the class of the first rule whose condition matches with the example is assigned as the result.

  Yarowsky (1994) used Decision Lists to solve a particular type of lexical ambiguity, the Spanish and French accent restoration. In another work, Yarowsky (1995b) applied Decision Lists to WSD. In this work, each *condition* corresponds to a feature, the *values* are the word senses, and the weights are calculated with a log-likelihood measure indicating the probability of the sense given the feature value.

  Some recent experiments suggest that Decision Lists can be also very productive for high precision feature selection (Martínez et al., 2002; Agirre and Martínez, 2004b) which could be potentially useful for bootstrapping.

- **Methods based on Decision Trees.** A Decision Tree is a way to represent classification rules underlying data, with a n-ary branching tree structure that recursively partitions the data. Each branch of a decision tree represents a rule that test a conjunction of basic features (internal nodes) and makes a prediction of the class label in the terminal node. Although decision trees have been used for years in many classification problems in the Artificial Intelligence area and that many software implementations are freely available, decision trees have not been applied to WSD so frequently. Mooney (1996) used the C4.5 algorithm (Quinlan, 1993) in a comparative experiment with many ML algorithms for WSD. He concluded that decision trees are not between the top performing methods. Some factors that make decision trees not appropriate for WSD are: 1) The induction algorithm performs a very high data fragmentation in the presence of features with many values; 2) The computational cost is high in very large feature spaces; and 3) Terminal

nodes corresponding to rules that cover very few training examples do not produce reliable estimates of the class label. The first two factors make very difficult for a DT based algorithm to include lexical information about the words in the context to be disambiguated, while the third problem is especially bad in the presence of small training sets. It has to be noted that part of these problems can be partially mitigated by using simpler related methods such as decision lists.

- **Methods based on rule combination.** There are some algorithms specifically designed for rule combination. This is the case of AdaBoost (Freund and Schapire, 1997).

  The main idea of the AdaBoost algorithm is to linearly combine many simple and not necessarily very accurate classification rules (called *weak rules* or *weak hypotheses*) into a strong classifier with an arbitrarily low error rate on the training set. Weak rules are trained sequentially by maintaining a distribution of weights over training examples and by updating it so as to concentrate weak classifiers on the examples that were most difficult to classify by the ensemble of the preceding weak rules. AdaBoost has been successfully applied to many practical problems, including several NLP tasks (Schapire, 2002).

  Several experiments on the DSO corpus (Escudero et al., 2000c,a,b) concluded that the boosting approach surpasses many other ML algorithms on the WSD task. We can mention, among others, Naive Bayes, Exemplar-based learning and Decision Lists. In these experiments, simple *decision stumps* (rules that make a test on a single binary feature) were used as weak rules, and a more efficient implementation of the algorithm, called LazyBoosting, was used to deal with the large feature set induced. These experiments are presented in sections 3.4 and 6.2 of this document where the AdaBoost algorithm with confidence-rated predictions (Schapire and Singer, 1999) is applied to WSD.

Linear Classifiers and Kernel-based Methods

Linear classifiers have been very popular in the field of Information Retrieval (IR), since they have been successfully used as simple and efficient models for text categorisation. A linear (binary) classifier, is an hyperplane in an n-dimensional feature space that can be represented with a weight vector w and a bias b indicating the distance of the hyperplane to the origin: $h(x) = <w, x> + b$ (where $<\cdot,\cdot>$ stands for the dot product). The weight vector has a component for each feature, expressing the "importance" of this feature in the classification rule, which can be stated as: $f(x) = +1$ if $h(x) \geq 0$ and $f(x) = -1$ otherwise. There are many on-line learning algorithms for training such linear classifiers (Perceptron,

Widrow-Hoff, Winnow, Exponentiated-Gradient, Sleeping Experts, etc.) that have been applied to text categorisation –see, for instance (Dagan et al., 1997).

Escudero et al. (2000a) used the SNoW architecture (based on Winnow) in a comparative experiment but obtaining fairly low results. A particular algorithm for learning linear classifiers has been gaining popularity in the NLP field, recently. We refer to Support Vector Machines (Cristianini and Shawe-Taylor, 2000), which choose the hyperplane with *maximum margin* among all possible hyperplanes that separate the positive examples from the negatives (that with same distance to positive than to negative examples). This learning bias has proven to be very robust and powerful leading to very good results in many NLP problems.

Regarding kernel-based methods, we find applications of SVMs in (Cabezas et al., 2001; Murata et al., 2001; Lee and Ng, 2002). In the third paper, SVMs achieved the best results in a comparative study among many machine learning algorithms. The experiments reported in this report seem to confirm the superiority of SVMs in the WSD domain. In Senseval-3 English Lexical Sample Task kernel-based methods obtained very good results using SVMs (Agirre and Martínez, 2004b; Escudero et al., 2004; Lee et al., 2004; Strapparava et al., 2004), Kernel PCA (Carpuat et al., 2004) and RLSC (Grozea, 2004; Popescu, 2004).

One of the biggest advantages of the use of kernels is the design of specific kernels for the task at hand. We can design a kernel with semantic information from features for the WSD task. Strapparava et al. (2004) and Popescu (2004) uses a rich combination of features using a kernel for each kind of features achieving very good results.

Considering a network of interconnected linear units is a way for training non-linear separators. Such connectionist methods were broadly used in the late eighties and first nineties to represent semantic models in the form of networks. More recently, a standard supervised feed-forward neural network model was presented by (Towell and Voorhees, 1998) for disambiguating highly ambiguous words, in a framework including the combined use of labelled and unlabelled examples.

2.3 Current Open Issues

2.3.1 Bootstrapping Approaches

Steven Abney (2002) defines *bootstrapping* as a problem setting in which the task is to induce a classifier from a small set of labelled data and a large set of unlabelled data. In principle, bootstrapping is a very interesting approach for Word Sense Disambiguation

due to the availability of a large amount of unlabelled data, and the lacking of labelled data.

One of the first bootstrapping algorithms applied in Computational Linguistics is that of (Yarowsky, 1995b) which was applied to WSD. Decision Lists were used as the supervised base learner. The input of the Yarowsky's algorithm is a set of labelled examples, also called *seeds*, and a set of unlabelled examples, typically about the 90% of the total. The algorithm consists of an iterative process in which a decision list learner is built with the corpus of seeds and applied to the unlabelled data. In the next iteration the algorithm gets the rules of the seeds plus those of best confidence acquired from the unlabelled set and a new learner is built. The process is repeated until reaching some training parameters. One of the drawbacks of bootstrapping is the difficultness of guarantee theoretically the learning process. There are few theorems on that issue (Blum and Mitchell, 1998) but they suppose no realistic assumptions.

Co-training (Blum and Mitchell, 1998) is an alternative algorithm appeared three years later. It was initially applied to classify Web pages. It considers two views of the problem and apply at each learning iteration one of them alternatively. Each view is a learning classifier focussing on a set of features. The sets of features of both views have to be mutually exclusive. Collins and Singer (1999) applied a modification of the Yarowsky's algorithm and co-training to Name Entity Classification obtaining accuracies over 90%. Co-training has a theoretical analysis by (Dasgupta et al., 2001). It assumes the conditional independence of the two alternative views given the class label. Abney (2002) relaxed the independence rule and implemented the algorithm suggested by Dasgupta et al. (2001). This algorithm is called Greedy Agreement algorithm. It is a very simple iterative process that selects a rule at each iteration which maximises the agreement between the two view classifiers on labelled data. Steven Abney also gives a theoretical analysis of the good performance of co-training and the Yarowsky's algorithm in (Abney, 2002, 2004). Suárez (2004) achieved moderate results on WSD data using a co-training variant he called re-training. Mihalcea (2004) proposes a combination of co-training and majority voting that smoothes the learning curves and improves the average performance. However, this approach assumes the same sense distribution in both labelled and unlabelled example sets.

Another bootstrapping approach is transductive learning. It consists of introducing the unlabelled data in the learning process. It begins by learning by the usual inductive approach from labelled data. Then it adjust the separating hyperplane by guessing the class of the unlabelled examples. Joachims (1999) used Transductive Support Vector Machines (TSVMs) to bootstrap examples of text categorisation. He reports good performance on the Reuters Text Categorisation corpus. He used a few set of examples as

seeds obtaining better results than using inductive SVMs. Unfortunately, Escudero and Màrquez (2003) report very irregular and bad performance using the DSO WSD corpus (see chapter 5).

However, bootstrapping is one of the most promising research lines to open the knowledge acquisition bottleneck in Word Sense Disambiguation, due to the lack of labelled data and the availability of a huge amount of unlabelled data. The research community is dedicating a lot of effort to this issue. Recently, Ando and Zhang (2005) have improved a system by using unlabelled examples to create auxiliary problems in learning structural data from CoNLL'00 (syntactical chunking) and CoNLL'03 (named entity chunking) tasks. Unfortunately, to date neither theoretical nor experimental evidences of the usefulness of bootstrapping techniques to WSD have been shown.

2.4 Evaluation

In the WSD field there have been a inflexion point since the organisation of the Senseval evaluation exercises. This section describes the evaluation in the WSD field before and after this point.

2.4.1 Pre-Senseval evaluations

There have been very few direct comparisons between alternative methods for WSD before the Senseval era. It was commonly stated that Naive Bayes, Neural Networks and Exemplar-based learning represented state-of-the-art accuracy on supervised WSD (Mooney, 1996; Ng, 1997a; Leacock et al., 1998; Fujii et al., 1998; Pedersen and Bruce, 1998). There were two well-known comparisons between Naive Bayes and Exemplar-based methods: the works by Mooney (1996) and by Ng (1997a). The first comparison used more algorithms but on a small set of words. The second comparison used fewer number of algorithms but on more words and more examples per word.

Mooney's paper showed that the Bayesian approach was clearly superior to the Exemplar-based approach. Although it was not explicitly said, the overall accuracy of Naive Bayes was about 16 points higher than that of the Example-based algorithm, and the latter was only slightly above the accuracy that a Most Frequent Classifier. In the Exemplar-based approach, the algorithm applied for classifying new examples was a standard k-Nearest-Neighbour (kNN), using the Hamming distance for measuring closeness. Neither example weighting nor attribute weighting were applied, k was set to 3, and the number of attributes used was said to be almost 3,000.

The second paper compared the Naive Bayes approach with PEBLS (Cost and Salzberg, 1993), a more sophisticated Exemplar-based learner especially designed for dealing with examples having symbolic features. This paper showed that, for a large number of nearest-neighbours, the performance of both algorithms was comparable, while if cross-validation was used for parameter setting, PEBLS slightly outperformed Naive Bayes. It has to be noted that the comparison was carried out in a limited setting, using only 7 feature types, and that the attribute/example-weighting facilities provided by PEBLS were not used. The author suggests that the poor results obtained in Mooney's work were due to the metric associated to the kNN algorithm, but he did not test if the Modified Value Difference Metric (MVDM), proposed by Cost and Salzberg (1993), used in PEBLS was superior to the standard Hamming distance or not.

Another surprising result that appears in Ng's paper is that the accuracy results obtained were 1-1.6% higher than those reported by the same author one year before (Ng and Lee, 1996), when running exactly the same algorithm on the same data, but using a larger and richer set of attributes. This apparently paradoxical difference is attributed, by the author, to the feature pruning process performed in the older paper.

Apart from the contradictory results obtained by the previous papers, some methodological drawbacks of both comparisons should also be pointed out. On the one hand, Ng applies the algorithms on a broad-coverage corpus but reports the accuracy results of a single testing experiment, providing no statistical tests of significance. On the other hand, Mooney performs thorough and rigorous experiments, but he compares the alternative methods on a limited domain consisting of a single word with a reduced set of six senses. Obviously, this extremely specific domain does not guarantee the reaching of reliable conclusions about the relative performances of alternative methods when applied to broad-coverage domains.

Section 3.3 clarifies some of the contradictory information in both previous papers and proposes changes in the representation of features to better fit Naive Bayes and Exemplar-based algorithms on WSD data. Section 3.4 is devoted to the adaptation of AdaBoost algorithm to WSD data and compares it with both previous algorithms showing a better performance of AdaBoost.

Lee and Ng (2002) compared different algorithms and showed the importance and influence of the knowledge sources, that is, how we represent and provide examples to the machine learning algorithms. They pointed Support Vector Machines to be the best option. They obtained very good results on Senseval-3 English Lexical Sample task.

Chapter 4 reports bad experimental results when training and testing across corpora. The best performance is achieved by margin-based classifiers. Finally, section 3.5 is

devoted to perform a comparison taking into account not only the typical accuracy measures, but also test measures like kappa values and agreement rates. It shows the best performance for Support Vector Machines when small training set is available and for AdaBoost when we have large amount of examples.

2.4.2 Senseval Evaluation

Since 1998 a broad-coverage comparison have been performed on the Senseval evaluation exercise (Kilgarriff, 1998a; Kilgarriff and Rosenzweig, 2000). Under the auspices of ACL-SIGLEX and EURALEX, Senseval is attempting to run ARPA-like comparisons between WSD systems. Like other international competitions of the style of those sponsored by the American government, MUC or TREC, Senseval was designed to compare, within a controlled framework, the performance of different approaches and systems for Word Sense Disambiguation. The goal of Senseval is to evaluate the power and the weakness of WSD programs with respect to different words, different languages and different tasks. Typically, in an "all words" task, the evaluation consists of assigning the correct sense to almost all the words of a text. In a "lexical sample" task, the evaluation consists of assigning the correct sense to different occurrences of the same word. In a "translation task", senses correspond to distinct translations of a word in another language.

Basically, Senseval classifies the systems into two different types: supervised and unsupervised systems. However, there are systems difficult to classify. In principle, *knowledge-based* systems (mostly unsupervised) can be applied to all three tasks, whereas *exemplar-based* systems (mostly supervised) can participate preferably in the lexical-sample and translation tasks.

The first Senseval edition (hereinafter Senseval-1) was carried out during summer of 1998 and it was designed for English, French and Italian, with 25 participating systems. It included only the lexical sample task. Up to 17 systems participated in the *English lexical-sample* task (Kilgarriff and Rosenzweig, 2000), and the best results (all supervised) achieved 75-80% accuracy. Although senses were based on the HECTOR dictionary most systems were based on WordNet senses. These systems were in severe disadvantage due to the mappings of senses.

The second Senseval contest (hereinafter Senseval-2) was made in July 2001 and included tasks for 12 languages: Basque, Czech, Dutch, English, Estonian, Chinese, Danish, Italian, Japanese, Korean, Spanish and Swedish (Edmonds and Cotton, 2001). About 35 teams participated, presenting up to 94 systems. Some teams participated in several tasks allowing the analysis of the performance across tasks and languages. Furthermore, some words for several tasks were selected to be "translation-equivalents" to

some English words to perform further experiments after the official competition. All the results of the evaluation and data are now in the public domain including: results (system scores and plots), data (system answers, scores, training and testing corpora, and scoring software), system descriptions, task descriptions and scoring criteria. About 26 systems took part in the *English lexical-sample* task, and the best results achieved 60-65% precision (see table 2.1). After Senseval-2, (Lee and Ng, 2002; Florian and Yarowsky, 2002) have reported 65.4% and 66.5% accuracy results, respectively. The first one focussed in the study of the features and the second work on the combination of classifiers. Table 2.2 shows the accuracy of the best systems on the All Words task of Senseval-2 event. The best system achieved an accuracy of 69%.

system	**fine accuracy**
JHU(R)	.64
SMUIs	.63
KUNLP	.62
Stanford-CS224N	.61
Sinequa-LIA SCT	.61
TALP	.59
Duluth-3	.57
UMD-SST	.57
UNED LS-T	.50

Table 2.1: Senseval-2 English Lexical Sample Task Results

An explanation for these low results, with respect to Senseval-1, is the change of the dictionary and the addition of new senses. In Senseval-1, the *English lexical-sample* task was made using the HECTOR dictionary, whereas in Senseval-2 a preliminary version of WordNet 1.7 was used. In the first case, also manual connections to WordNet 1.5 and 1.6 were provided. The second factor is the addition of many proper nouns, phrasal verbs, multiwords to the different tasks.

system	**precision**
SMUaw-	.69
AVe-Antwerp	.64
LIA-Sinequa-AllWords	.62
david-fa-UNED-AW-T	.58
david-fa-UNED-AW-U	.56

Table 2.2: Senseval-2 All Words Task Results

It is very difficult to make a direct comparison among the Senseval systems because they differ very much in the methodology, the knowledge used, the task in which they participated, and the measurement that they wanted to optimise ("coverage" or "precision"). Instead, we prefer to briefly describe some of the top-ranked supervised systems that participated in Senseval-2. With respect to the *English-all-words* task we describe SMUaw (Mihalcea and Moldovan, 2001) and Antwerp (Hoste et al., 2001, 2002a) systems. Regarding the *English lexical-sample* we describe JHU-English (Yarowsky et al., 2001) and TALP (Escudero et al., 2001) systems.

The SMUaw system acquired patterns and rules from the SemCor corpus and WordNet, via a set of heuristics. The semantic disambiguation of a word was performed by means of its semantic relations with respect to the surrounding words in the context. These semantic relations were extracted from GenCor, a large corpus of 160,000 sense annotated word pairs. This corpus was obtained from SemCor and WordNet. As this initial system had not full coverage, afterwards a new method was applied to propagate the senses of the disambiguated words to the rest of ambiguous words in the context. For those words remaining ambiguous, the most frequent sense of WordNet was applied. The performance of this system was 63.8% precision and recall.

The Antwerp system also used SemCor. This system applied ML classifiers. The system combined, using a voting scheme, different types of learning algorithms: MBL (TiMBL) and rule induction (Ripper). They learnt a classifier for each word (a word expert) choosing the best option after an optimisation process. The performance of this system was of 64% precision and recall.

The JHU-English system integrated 6 supervised subsystems. The final result was obtained combining the output of all the classifiers. The six subsystems were based on: decision lists (Yarowsky, 2000), error-driven transformation-based learning (Brill, 1995; Ngai and Florian, 2001), a vector space model, decision trees, and two Naive Bayes probabilistic models (one trained on words and the other on lemmas). Each subsystem was trained using a set of words in a fixed context window, its morphologic analysis, a variety of syntactic features like direct subjects, objects, circumstantial complements and several modifiers. These syntactic features were obtained using a set of heuristics based on patterns (Ngai and Florian, 2001). The output of each subsystem was combined by means of a meta-classifier. Results achieved by this system were 64.2% precision and recall and 66.5% in a extended version (Florian and Yarowsky, 2002).

The TALP system used the LazyBoosting algorithm to classify the senses of the words using a rich set of attributes, including domain codes associated to the WordNet synsets (Magnini and Cavaglia, 2000). Like other systems, the examples were preprocessed (for example eliminating the collocations) to decrease the degree of polysemy. The system also

used a hierarchical model in the style of (Yarowsky, 2000) although it was seen that this decomposition was counter-productive. The method itself produced a combination of weak rules (decision stumps) without explicitly combine the different outputs. The performance achieved by this system was 59.4% of precision and recall. Further description of the TALP system can be found in section 6.2 of this document.

Section 6.3 is devoted to systematically study the outputs of the best six systems of Senseval-2 English Lexical Sample task. This study aims at understanding the characteristics and influences of the different components of these systems on the final performance. Some interesting conclusions are drawn from this study pointing out the importance of the preprocessing of the data and the treatment of special cases, such as name-entities or phrasal verbs.

Senseval-3

The third edition of Senseval (hereinafter Senseval-3) was made in July 2004. 27 teams participated in the English Lexical Sample task, presenting 47 systems (Mihalcea et al., 2004); and 16 teams in the All-Words task, presenting 26 systems (Snyder and Palmer, 2004). Senseval is growing in interest as shown by the large number of participants. The best systems obtained 72.9% and 65.2% accuracy for both tasks respectively. A remarkable result was that the first 17 systems in the Lexical Sample Task achieves a very similar accuracy (with a maximum difference of 3.8%).

It seems that in the English Lexical Sample task there were no qualitative improvements, comparing with previous editions of the task. There is an increment of 10% of accuracy from Senseval-2 to Senseval-3, but is probably due to the decrease in the number of senses. In Senseval-3 English Lexical Sample there are not multiwords, phrasal verbs and these kind of satellite senses. Table 2.3 shows the results of the best system of each team. The results of this event evidence a clear dependence of the knowledge (features) and not the Machine Learning algorithm applied. Another important issue is how features are combined. The most used algorithms are those based in Kernels. Carpuat et al. (2004); Agirre and Martínez (2004b) perform a combination of several ML components. Lee et al. (2004); Escudero et al. (2004) use SVMs as learning methods focussing the attention on the feature set. The former has a fine tuned set of features (taking special attention to the syntactical features) and the later has a huge set of them and performs feature selection and multiclass binarisation selection (between one-vs-all and constraint classification approaches). Strapparava et al. (2004) also use SVMs but developing latent Semantic Kernels for the different classes of features. They obtained the second position in the ranking achieving an accuracy of 72.6%. Popescu (2004); Grozea

(2004) used RLSC learning algorithm. The first used an standard feature set; and the second extends the feature set, converting the classification problem to a regression one. A postprocess on the output tries to correct the tendency to vote the most frequency sense by taking into account the prior distribution of senses in the training data. This system won the task achieving an accuracy of 72.9%. The most novel aspects of these systems were: the postprocessing of the output weights by (Grozea, 2004); the use of kernels in the SVMs for the different classes of features by (Strapparava et al., 2004); and the study of the features used by (Lee et al., 2004; Escudero et al., 2004; Decadt et al., 2004).

system	**fine accuracy**
htsa3 (U. Bucharest)	72.9
IRST-Kernels (ITC-IRST)	72.6
nusels (N.U. Singapure)	72.4
BCU_comb (Basque C. U.)	72.3
rlsc-comb (U. Bucharest)	72.2
GLTC_HKUST_all (HKUST)	71.4
TALP (UPC)	71.3
MC-WSD (Brown U.)	71.1
NRC-Fine (NRC)	69.4
GAMBL (U. Antwerp)	67.4
SinequaLex (Sinequa Labs)	67.2
CLaC1 (Concordia U.)	67.2
UMD_SST4 (U. Maryland)	66.0
Prob1 (Cambridge U.)	65.1
SyntaLex-3 (U. Toronto)	64.6
UNED (UNED)	64.1
Duluth-ELSS (U. Minnesota)	61.8
UJAEN (U. Jaén)	61.3

Table 2.3: Senseval-3 English Lexical Sample Task Results (in %)

Table 2.4 shows the results of the English All-Words task (Snyder and Palmer, 2004). The best system achieved an accuracy of 65.2% (Decadt et al., 2004). This system is the continuation of that presented in Senseval-2 plus a feature and parameter selection performed using Genetic Algorithms. They developed a supervised system based on memory-based learning, taking as training data all available corpora: Semcor (that included in WordNet 1.7.1); those of the previous editions of Senseval (lexical sample and all-words); the *line-*, *hard-* and *serve-* corpora; and the examples of WordNet 1.7.1. They

tuned the features and parameters of the system with a Genetic Algorithm. They applied the same system to the English Lexical Sample task but they did not achieve so good results. The second best-performing system (Mihalcea and Faruque, 2004) achieved an accuracy of 64.6%. It was composed by two components: a regular memory-based classifier with a small set of features focussing on syntactical dependencies and an interesting generalisation learner. This last component learns, through syntactical dependencies and WordNet hierarchies, generalisation patterns that could be applied to disambiguate unseen words[12] . Next system (Yuret, 2004) used Naive Bayes with the standard set of features and studied the size of the context. (Vázquez et al., 2004; Villarejo et al., 2004) perform a combination of multiple systems from different research teams. The most interesting issue presented in this task is that of the generalisation learner for the treatment of the unseen words (Mihalcea and Faruque, 2004).

system	**precision**
GAMBL-AW-S	65.2
SenseLearner-S	64.6
Koc University-S	64.1
R2D2: Enlish-all-words	64.1
Meaning-allwords-S	62.4
upv-shmm-eaw-S	60.9
LCCaw	60.7
UJAEN-S	59.0
IRST-DDD-00-U	58.3
University of Sussex-Prob5	57.2
DFA-Unsup-AW-U	54.8
KUNLP-Eng-All-U	50.0
merl.system3	45.8
autoPS-U	43.6
clr04-aw	43.4
DLSI-UA-all-Nosu	28.0

Table 2.4: Senseval-3 All Words Task Results (in %)

[12]This component is capable to disambiguate *take tea* from a training example containing *drink water*, by using the WordNet hierarchy and the dependency *verb-object*.

2.5 Further Readings

During the compilation of this state of the art, we benefited from a number of good papers and surveys. These references represent an excellent starting point to begin the research in both WSD and ML fields. Most of them are included in the following list:

- **Word Sense Disambiguation**: (Ide and Véronis, 1998), (Kilgarriff, 1998a), (Rigau, 1998), (Kilgarriff and Rosenzweig, 2000), and (Agirre and Edmonds, 2006).
- **Machine Learning**: (Dietterich, 1997), (Mitchell, 1997), (Cardie and Mooney, 1999), (Màrquez, 1999), and (Manning and Schütze, 1999).
- **Word Sense Disambiguation and Machine Learning**: (Martínez, 2004), and (Suárez, 2004), and (Màrquez et al., 2006).

Chapter 3

A Comparison of Supervised ML Algorithms for WSD

This chapter presents a comparison between Machine Learning Algorithms when applied the Word Sense Disambiguation. In particular we work with: Naive Bayes, Exemplar-based, Decision Lists, AdaBoost, and Support Vector Machines. They are explained in detail in section 3.1. First, we adapt the algorithms to the Word Sense Disambiguation setting. The standard datasets used typically by the Machine Learning community in the 90's are represented as matrices of few columns (attributes) and possibly many rows (examples). On the contrary, Word Sense Disambiguation available datasets are represented as matrices of many columns (attributes) and usually few rows (examples) with lots of zeros (sparse information). The codification lexical information implies the treatment of a large number of features or alternatively features with lots of values (since thousands of different words appear in a training set). Both the concepts and the representation appears very sparsely and there are few examples. These characteristics of the data implie that not all algorithms work well on this problem. So, the first issue is to adapt the algorithms to the Word Sense Disambiguation data and to find an appropriate representation of the information. The computational cost is also a very important issue to take into account.

Section 3.3 analyses how to represent the information for Naive-Bayes and Exemplar-Based algorithms, and compares both algorithms under the same framework. The proposed data representation is also used later on Decision Lists.

Section 3.4 explains an adaptation of the AdaBoost.MH boosting algorithm (this adaptation has been also applied to the last algorithm, Support Vector Machines). This

work contains also some ways of speeding up the algorithm by reducing the feature space to explore. The best variant has been called LazyBoosting.

The final section contains a comparison of the preceding algorithms under the same conditions: Naive-Bayes, Exemplar-Based, Decision Lists, AdaBoost, and Support Vector Machines. The comparison not only takes into account the typical precision-recall measures, but also the kappa statistic, and agreement ratio. The t-Student measure of significance has been applied to check the statistical significance of the differences observed. See appendix A for all the formulae.

3.1 Machine Learning for Classification

The goal in supervised learning for *classification* consists of inducing an approximation (or *hypothesis*) h of an unknown function f defined from an input space X to a discrete unordered output space $Y = \{1, \ldots, K\}$, given a training set S (if the output space is continuous the problem is called *regression* instead of classification).

The training set contains n training examples, $s = \{(x_1, y_1), \ldots, (x_n, y_n)\}$, which are pairs (x, y) where x belongs to X and $y = f(x)$. The x component of each example is typically a vector $x = (x_1, \ldots, x_n)$, whose components, called *attributes* (or *features*), are discrete- or real-valued and describe the relevant information/properties about the example. The values of the output space Y associated to each training example are called *classes* (or *categories*). Therefore, each training example is completely described by a set of attribute-value pairs, and a class label.

In the *Statistical Learning Theory* field (Vapnik, 1998), the function f is viewed as a probability distribution $P(X, Y)$ and not as a deterministic mapping, and the training examples are considered as a set being i.i.d. generated from this distribution. Additionally, X is usually identified as $\Re^n$, and each example x as a point in $\Re^n$ with one real-valued feature for each dimension. All along this document we will try to maintain the descriptions and notation compatible with both approaches.

Given a training set S, a learning algorithm induces a classifier, denoted h, which is an hypothesis about the true function f. In doing so, the learning algorithm can choose among a set of possible functions H, which is referred to as the *space of hypotheses*. Learning algorithms differ in which space of hypotheses take into account (e.g., linear functions, domain partitioning by axis parallel hyperplanes, radial basis functions, etc.), in the representation language used (e.g., decision trees, sets of conditional probabilities, neural networks, etc.), and in the *bias* they use for choosing the *best* hypothesis among

several that can be compatible with the training set (e.g., simplicity, maximal margin, etc.).

Given new x vectors, h is used to predict the corresponding y values, i.e., to *classify* the new examples, and it is expected to be coincident with f in the majority of the cases, or, equivalently, to perform a small number of errors.

The measure of the error rate on unseen examples is called *generalisation* (or *true*) *error*. It is obvious that generalisation error cannot be directly minimised by the learning algorithm since the true function f, or the distribution $P(X, Y)$, is unknown. Therefore, an inductive principle is needed. The most common way to proceed is to directly minimise the *training* (or *empirical*) error, that is, the number of errors on the training set. This principle is known as *Empirical Risk Minimisation*, and gives a good estimation of the generalisation error in the presence of many training examples. However, in domains with few training examples, forcing a zero training error can lead to overfit the training data and to generalise badly. A notion of complexity of the induced functions is also related to the risk of overfitting –see again (Vapnik, 1998)– and this risk is increased in the presence of outliers and noise (i.e., very exceptional and wrongly classified training examples, respectively). This tradeoff between training error and complexity of the induced classifier is something that has to be faced in any experimental setting in order to guarantee a low generalisation error.

Finally, we would like to briefly comment a terminology issue that can be rather confusing in the WSD literature. Recall that, in machine learning, the term *supervised learning* refers to the fact that the training examples are annotated with the *class labels*, which are taken from a pre-specified set. Instead, *non supervised learning* refers to the problem of learning from examples when there is no set of pre-specified class labels. That is, the learning consists of acquiring the similarities between examples to form clusters that can be later interpreted as classes (this is why it is usually referred to as *clustering*). In the WSD literature, the term *unsupervised learning* is used with another meaning, which is the acquisition of disambiguation models or rules from non-annotated examples and external sources of information (e.g., lexical databases, aligned corpora, etc.). Note that in this case the set of class labels (which are the senses of the words) are also specified in advance.

3.1.1 Naive Bayes

Naive Bayes (NB) is the simplest representative of probabilistic learning methods. It has been used in its most classical setting (Duda and Hart, 1973), in which, assuming independence of features, it classifies a new example $x = (x_1, \ldots, x_m)$ by assigning the

sense k that maximises the conditional probability of the sense given the observed sequence of feature of that example. That is,

$$\arg\max_k P(k|x_1,\ldots,x_m) = \arg\max_k \frac{P(x_1,\ldots,x_m|k)P(k)}{P(x_1,\ldots,x_m)} \approx \arg\max_k P(k)\prod_{i=1}^{m} P(x_i|k)$$

where $P(k)$ and $P(x_i|k)$ are estimated from the training set, using relative frequencies (maximum likelihood estimation). To avoid the effects of zero counts when estimating the conditional probabilities of the model, a very simple smoothing technique, proposed in (Ng, 1997a), has been used. It consists in replacing zero count of $P(x_i|k)$ with $P(k)/n$ where n is the number of training examples.

3.1.2 Exemplar-based learning

We will use a k-Nearest Neighbour (kNN) algorithm as a representative of exemplar-based learning. kNN-based learning is said to be the best option for WSD in (Ng, 1997a). Other authors (Daelemans et al., 1999) argue that exemplar-based methods tend to be superior in NLP problems because they do not forget exceptions.

In kNN, all examples are stored in memory during training and the classification of a new example is based on the senses of the k most similar stored examples. In order to obtain the set of nearest neighbours, the example to be classified $x = (x_1,\ldots,x_m)$ is compared to each stored example $x_i = (x_{i_1},\ldots,x_{i_m})$, and the *distance* between them is calculated. The most basic metric (also called Hamming distance), defined as follows:

$$\Delta(\mathbf{x},\mathbf{x}_i) = \sum_{j=1}^{m} w_j \delta(x_j, x_{i_j})$$

where w_j is the weight of the j-th feature and $\delta(x_j,x_{i_j})$ is the distance between two values, which is 0 if $x_j = x_{i_j}$ and 1 otherwise.

In the implementation tested in these experiments, we used Hamming distance to measure closeness and the Gain Ratio measure (Quinlan, 1993) for estimating feature weights. For $k's$ greater than 1, the resulting sense is the weighted majority sense of the k nearest neighbours -where each example votes its sense with a strength proportional to its closeness to the test example. The kNN algorithm is run several times using different number of nearest neighbours (1, 3, 5, 7, 10, 15, 20 and 25) and the results corresponding to the best choice for each word are reported[1].

[1] In order to construct a real kNN-based system for Word Sense Disambiguation, the k parameter should have been estimated by cross-validation using only the training set (Ng, 1997a), however, in our case, this cross-validation inside the cross-validation involved in the testing process would have introduced a prohibitive overhead.

One of the main issues of section 3.3 is the adaptation of NB and kNN to the WSD. In order to test some of the hypotheses about the differences between Naive Bayes (NB) and Exemplar-based ($EB_{h,k}$[2]) approaches, some variants of the basic kNN algorithm have been implemented:

- **Example weighting**. This variant introduces a simple modification in the voting scheme of the k nearest neighbours, which makes the contribution of each example proportional to their importance. When classifying a new test example, each example of the set of nearest neighbours votes for its class with a weight proportional to its closeness to the test example. In the related section, this variant will be referred to as $EB_{h,k,e}$.

- **Attribute weighting**. This variant consists of ranking all attributes by relevance and making them contribute to the distance calculation with a weight proportional to their importance. The attribute weighting has been done using the *RLM* distance measure by López de Mántaras (1991). This measure, belonging to the distance/information-based families of attribute selection functions, has been selected because it showed better performance than seven other alternatives in an experiment of decision tree induction for POS tagging (Màrquez, 1999). In the related section, this variant will be referred to as $EB_{h,k,a}$.

When both modifications are put together, the resulting algorithm will be referred to as $EB_{h,k,e,a}$. Finally, we have also investigated the effect of using an alternative metric.

- **Modified Value Difference Metric** (*MVDM*), proposed by Cost and Salzberg (1993), allows making graded guesses of the match between two different symbolic values. Let v_1 and v_2 be two values of a given attribute a. The MVDM distance between them is defined as:

$$d(v_1, v_2) = \sum_{i=1}^{m} |P(C_i|v_1) - P(C_i|v_2)| \approx \sum_{i=1}^{m} \left| \frac{N_{1,i}}{N_1} - \frac{N_{2,i}}{N_2} \right|$$

  where m is the number of classes, $N_{x,i}$ is the number of training examples with value v_x of attribute a that are classified as class i in the training corpus and N_x is the number of training examples with value v_x of attribute a in any class. Hereinafter, this variant will be referred to as $EB_{cs,k}$. This algorithm has also been used with the example-weighting option ($EB_{cs,k,e}$).

TiMBL[3] is an available software that implements this method with all the options explained in this section (Daelemans et al., 2001).

[2] h stands for hamming distance and k for the number of nearest neighbours.

[3] `http://ilk.kub.nl/software.html`

3.1.3 Decision Lists

A decision list consists of a set of ordered rules of the form (*feature-value, sense, weight*). In this setting, the decision lists algorithm works as follows: the training data is used to estimate the features, which are weighted with a log-likelihood measure (Yarowsky, 1995b) indicating the likelihood of a particular sense given a particular feature value. The list of all rules is sorted by decreasing values of this weight; when testing new examples, the decision list is checked, and the feature with highest weight that is matching the test example selects the winning word sense.

The original formula in (Yarowsky, 1995b) can be adapted in order to handle classification problems with more than two classes. In this case, the weight of $sense_k$ when $feature_i$ occurs in the context is computed as the logarithm of the probability of $sense_k$ given $feature_i$ divided by the sum of the probabilities of the other senses given $feature_i$. That is:

$$weight(sense_k, feature_i) = Log(\frac{P(sense_k|feature_i)}{\sum_{j \neq k} P(sense_j|feature_i)})$$

These probabilities can be estimated using the maximum likelihood estimate, and some kind of *smoothing* to avoid the problem of 0 counts. There are many approaches for smoothing probabilities –see, for instance, (Chen, 1996). In these experiments a very simple solution has been adopted, which consists of replacing the denominator by 0.1 when the frequency is zero.

3.1.4 AdaBoost

AdaBoost (AB) is a general method for obtaining a highly accurate classification rule by combining many *weak* classifiers, each of which may be only moderately accurate. A generalised version of the AdaBoost algorithm, which combines weak classifiers with confidence-rated predictions (Schapire and Singer, 1999), has been used in these experiments. This algorithm is able to work efficiently in very high dimensional feature spaces, and has been applied, with significant success, to a number of practical problems.

The weak hypotheses are learned sequentially, one at a time, and, conceptually, at each iteration the weak hypothesis is biased to classify the examples which were most difficult to classify by the ensemble of preceding weak hypotheses. AdaBoost maintains a vector of weights as a distribution D_t over examples. At round t, the goal of the weak learner algorithm is to find a weak hypothesis, $h_t : X \rightarrow \Re$, with moderately low error

with respect to the weight distribution D_t. In this setting, weak hypotheses $h_t(x)$ make real-valued confidence-rated predictions. Initially, the distribution D_1 is uniform, but after each iteration, the boosting algorithm exponentially increases (or decreases) the weights $D_t(i)$ for which $h_t(x_i)$ makes a bad (or good) prediction, with a variation exponentially dependant to the confidence $|h_t(x_i)|$. The final combined hypothesis, $h_t : X \rightarrow \Re$, computes its predictions using a weighted vote of the weak hypotheses,

$$f(x) = \sum_{t=1}^{T} \alpha_t \cdot h_t(x)$$

For each example x, the sign of $f(x)$ is interpreted as the predicted class (the basic AdaBoost works only with binary outputs, -1 or +1), and the magnitude $|f(x)|$ is interpreted as a measure of confidence in the prediction. Such a function can be used either for classifying new unseen examples or for ranking them according to the confidence degree.

WSD defines multi-class classification problems, instead of binary. We have used the AdaBoost.MH algorithm (the pseudo-code is presented in figure 3.1) that generalises AdaBoost to multiclass multilabel classification (Schapire and Singer, 2000).

Let $S = \{(x_1, Y_1), \ldots, (x_m, Y_m)\}$ be the set of m training examples, where each instance x_i belongs to an instance space $\mathcal{X}$ and each Y_i is a subset of a finite set of labels or classes $\mathcal{Y}$. The size of $\mathcal{Y}$ is denoted by $k = |\mathcal{Y}|$. AdaBoost.MH maintains an $m \times k$ matrix of weights as a distribution D over m examples and k labels. The goal of the *WeakLearner* algorithm is to find a weak hypothesis with moderately low error with respect to these weights. Initially, the distribution D_1 is uniform, but the boosting algorithm updates the weights on each round to force the weak learner to concentrate on the pairs (example, label) which are hardest to predict.

More precisely, let D_t be the distribution at round t, and $h_t : \mathcal{X} \times \mathcal{Y} \rightarrow \Re$ the weak rule acquired according to D_t. The sign of $h_t(x, l)$ is interpreted as a prediction of whether label l should be assigned to example x or not. The magnitude of the prediction $|h_t(x, l)|$ is interpreted as a measure of confidence in the prediction. In order to understand correctly the updating formula this last piece of notation should be defined. Thus, given $Y \subseteq \mathcal{Y}$ and $l \in \mathcal{Y}$, let $Y[l]$ be $+1$ if $l \in Y$ and -1 otherwise.

Now, it becomes clear that the updating function increases (or decreases) the weights $D_t(i, l)$ for which h_t makes a good (or bad) prediction, and that this variation is proportional to $|h_t(x, l)|$.

Note that WSD is not a multi-label classification problem since a unique sense is expected for each word in context. In our implementation, the algorithm runs exactly in

procedure AdaBoost.MH
Input: $S = \{(x_i, Y_i), \forall i = 1..m\}$
S is the set of training examples
Initialize distribution D_1
m is the number of training examples
k is the number of classes
$D_1(i,l) = \frac{1}{mk}, \forall i = 1..m, \forall l = 1..k$
Perform T rounds:
for $t := 1$ **to** T **do**
Get the weak hypothesis $h_t : X \times Y \rightarrow \Re$
h_t = WeakLearner(X, D_t)
Update distribution D_t
$\boldsymbol{D_{t+1}(i,l) = \frac{D_t(i,l)\exp(-Y_i[l]h_t(x_i,l))}{Z_t}, \forall i = 1..m, \forall l = 1..k}$
Z_t is a normalisation factor
###(chosen so that D_{t+1} will be a distribution)
end for
return the combined hypothesis: $f(x,l) = \sum_{t=1}^{T} h_t(x,l)$
end AdaBoost.MH

Figure 3.1: The AdaBoost.MH algorithm

the same way as explained above, except that sets Y_i are reduced to a unique label, and that the combined hypothesis is forced to output a unique label, which is the one that maximises $f(x,l)$.

Up to now, it only remains to be defined the form of the WeakLearner. Schapire and Singer (1999) prove that the Hamming loss of the AdaBoost.MH algorithm on the training set[4] is at most $\prod_{t=1}^{T} Z_t$, where Z_t is the normalisation factor computed on round t. This upper bound is used in guiding the design of the WeakLearner algorithm, which attempts to find a weak hypothesis h_t that minimises:

$$Z_t = \sum_{i=1}^{m} \sum_{l \in \mathcal{Y}} D_t(i,l)\exp(-Y_i[l]h_t(x,l))$$

As in (Abney et al., 1999), very simple weak hypotheses are used to test the value of a boolean predicate and make a prediction based on that value. The predicates used are of the form "$f = v$", where f is a feature and v is a value (e.g.: "previous_word = *hospital*").

[4] i.e. the fraction of training examples i and labels l for which the sign of $f(x_i, l)$ differs from $Y_i[l]$.

Formally, based on a given predicate p, our interest lies on weak hypotheses h which make predictions of the form:

$$h(x,l) = \begin{cases} c_{0l} & \text{if } p \text{ holds in } x \\ c_{1l} & \text{otherwise} \end{cases}$$

where the c_{jl}'s are real numbers.

For a given predicate p, and bearing the minimisation of Z_t in mind, values c_{jl} should be calculated as follows. Let X_1 be the subset of examples for which the predicate p holds and let X_0 be the subset of examples for which the predicate p does not hold. Let $[\![\pi]\!]$, for any predicate π, be 1 if π holds and 0 otherwise. Given the current distribution D_t, the following real numbers are calculated for each possible label l, for $j \in \{0,1\}$, and for $b \in \{+1,-1\}$:

$$W_b^{jl} = \sum_{i=1}^{m} D_t(i,l)[\![x_i \in X_j \wedge Y_i[l] = b]\!]$$

That is, W_{+1}^{jl} (W_{-1}^{jl}) is the weight (with respect to distribution D_t) of the training examples in partition X_j which are (or not) labelled by l.

As it is shown in (Schapire and Singer, 1999), Z_t is minimised for a particular predicate by choosing:

$$c_{jl} = \frac{1}{2}\ln(\frac{W_{+1}^{jl}}{W_{-1}^{jl}})$$

These settings imply that:

$$Z_t = 2 \sum_{j \in \{0,1\}} \sum_{l \in \mathcal{Y}} \sqrt{W_{+1}^{jl} W_{-1}^{jl}}$$

Thus, the predicate p chosen is that for which the value of Z_t is smallest.

Very small or zero values for the parameters W_b^{jl} cause c_{jl} predictions to be large or infinite in magnitude. In practice, such large predictions may cause numerical problems to the algorithm, and seem to increase the tendency to overfit. As suggested in (Schapire and Singer, 1999), smoothed values for c_{jl} have been used.

A simple modification of the AdaBoost algorithm, which consists of dynamically selecting a very reduced feature set at each iteration, has been used to significantly increase the efficiency of the learning process with no loss in accuracy (see section 3.4 for the details about Lazy Boosting and how to set the number of rounds when learning.). This variant has been called LazyBoosting.

3.1.5 Support Vector Machines

Support Vector Machines (SVM) were introduced by (Boser et al., 1992), but they have only recently been gaining popularity in the machine learning community since 1996. Nowadays, SVM have been successfully applied to a number of problems related to pattern recognition in bioinformatics and image recognition. Regarding text processing, SVM have obtained the best results to date in text categorisation (Joachims, 1998) and they are being used in more and more NLP-related problems, e.g., chunking (Kudo and Matsumoto, 2001), parsing (Collins, 2002), or WSD (Murata et al., 2001; Lee and Ng, 2002; Escudero et al., 2004; Lee et al., 2004).

SVM are based on the *Structural Risk Minimisation* principle from the Statistical Learning Theory (Vapnik, 1998) and, in their basic form, they learn a linear hyperplane that separates a set of positive examples from a set of negative examples with *maximum margin* (the margin γ is defined by the distance of the hyperplane to the nearest of the positive and negative examples). This learning bias has proved to have good properties in terms of generalisation bounds for the induced classifiers. The linear classifier is defined by two elements: a weight vector w (with one component for each feature), and a bias b which stands for the distance of the hyperplane to the origin: $h(x) = <x, w> + b$ (see figure 3.2). The classification rule $f(x)$ assigns $+1$, to a new example x, when $h(x) =\geq 0$, and -1 otherwise. The positive and negative examples closest to the (w, b) hyperplane are called *support vectors*.

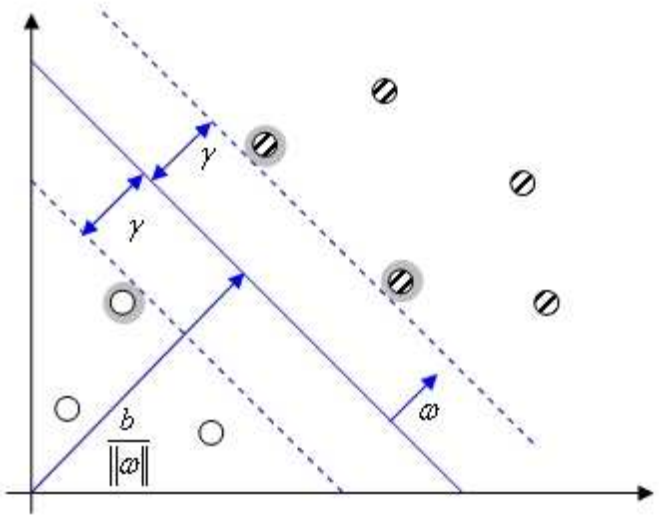

Figure 3.2: Example of margin γ and hyperplane $<x, w> + b$

Learning the maximal margin hyperplane (w, b) can be simply stated as a convex quadratic optimisation problem with a unique solution, consisting of (*primal formulation*):

$$minimize: \ ||w||$$

$$subject\,to: \ y_i(<w, x_i> + b) \geq 1, \ \forall\, 1 \leq i \leq N$$

where N is the number of training examples.

It is possible to obtain a *dual formulation* in which the solution is denoted as al lineal combination of the training examples (as vectors):

$$maximize: \ \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} y_i y_j \alpha_i \alpha_j <x_i, x_j>$$

$$subject\,to: \ \sum_{i=1}^{N} y_i \alpha_i = 0, \alpha_i \geq 0, \ \forall\, 1 \leq i \leq N$$

From this formulation, the orthogonal vector to the hyperplane rests defined as:

$$h(x) = <w, x> + b = b + \sum_{i=1}^{N} y_i \alpha_i <x_i, x>$$

where $\alpha_i \neq 0$ for all training examples in the margin (support vectors) and $\alpha_i = 0$ for the others. "$f(x) = +1$ when $h(x) \geq 0$ and -1 otherwise" defines the classifier from the dual formulation.

Despite the linearity of the basic algorithm, SVM can be transformed into a *dual form* (in which the training examples only appear inside dot product operations), allowing the use of *kernel functions* to produce non-linear classifiers. The kernel functions make SVM to work efficiently and implicitly in very high dimensional feature spaces, where new features can be expressed as combinations of many basic features (Cristianini and Shawe-Taylor, 2000). If we suppose we have a non lineal transformation Φ from $\Re^D$ to some Hilbert space $\Im$, then we can define a dot product using a kernel function like $K(x, y) = <\Phi(x), \Phi(y)>$ and obtain a dual formulation:

$$maximize: \ \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} y_i y_j \alpha_i \alpha_j K(x_i, x_j)$$

$$subject\,to: \ \sum_{i=1}^{N} y_i \alpha_i = 0, \alpha_i \geq 0, \ \forall\, 1 \leq i \leq N$$

and obtain the orthogonal vector to the hyperplane as:

$$h(x) = <w, \Phi(x)> + b = b + \sum_{i=1}^{N} y_i \alpha_i K(x_i, x)$$

Sometimes, training examples are not linearly separable or, simply, it is no desirable to obtain a perfect hyperplane. In these cases it is preferable to allow some errors in the training set so as to maintain a "better" solution hyperplane (see figure 3.3). This is achieved by a variant of the optimisation problem, referred to as *soft margin*, in which the contribution to the objective function of the margin maximisation and the training errors can be balanced through the use of a parameter called C. This parameter affects to the slack variables ξ_i in the function (see figure 3.3). This problem can be formulated as:

$$minimize: \ \frac{1}{2}<w, w> + C \sum_{i=1}^{N} \xi_i$$

$$subject\,to: \ y_i(<w, x_i> + b) \geq 1 - \xi_i, \ \xi_i \geq 0, \ \forall\, 1 \leq i \leq N$$

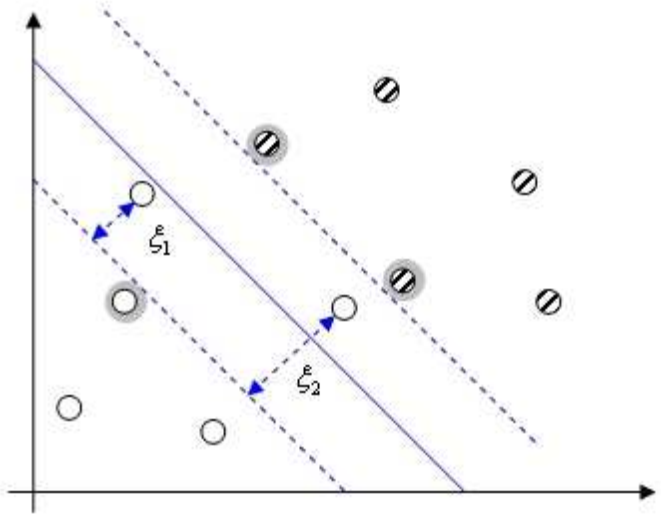

Figure 3.3: Example of soft margin with slack variables ξ_i

In the experiments presented below, the SVM^{light} software[5] has been used. We have worked only with linear kernels, performing a tuning of the C parameter directly on the DSO corpus. It has to be noted that no significant improvements were achieved by using non-linear kernels (polynomial or Gaussian).

[5]See appendix C for the web reference of this freely available implementation.

3.2 Setting

The comparison of algorithms has been performed in series of controlled experiments using exactly the same training and test sets for each method. The experimental methodology consisted of a 10-fold cross-validation. All accuracy/error rate results have been averaged over the results of the 10 folds. The statistical tests of significance have been performed using a 10-fold cross validation paired Student's t-test (Dietterich, 1998) with a confidence value of: $t_{9,0.975} = 2.262$.

3.2.1 Corpora

In our experiments, all approaches have been evaluated on the *DSO* corpus, a semantically annotated corpus containing 192,800 occurrences of 121 nouns and 70 verbs corresponding to the most frequent and ambiguous English words. These examples, consisting of the full sentence in which the ambiguous word appears, are tagged with a set of labels corresponding, with minor changes, to the senses of WordNet 1.5 (Miller et al., 1990). This corpus was collected by Ng and colleagues (Ng and Lee, 1996) and it is available from the Linguistic Data Consortium[6].

In the experiments reported in sections 3.3 and 3.4 a smaller group of 10 nouns (*age*, *art*, *car*, *child*, *church*, *cost*, *head*, *interest*, *line* and *work*) and 5 verbs (*fall*, *know*,*set*, *speak* and *take*) which frequently appear in the WSD literature has been selected. Since our goal is to acquire a classifier for each word, each word represents a classification problem. Table 3.1 shows these words with their number of classes (senses), the number of training examples and the accuracy of a naive Most Frequent Sense classifier (MFS).

3.2.2 Basic Features

Two sets of attributes have been used, which will be referred to as *SetA* and *SetB*, respectively. Let "$\ldots\ w_{-3}\ w_{-2}\ w_{-1}\ w\ w_{+1}\ w_{+2}\ w_{+3}\ldots$" be the context of consecutive words around the word w to be disambiguated. Attributes refer to this context as follows.

- *SetA* contains the seven following attributes: w_{-2}, w_{-1}, w_{+1}, w_{+2}, (w_{-2}, w_{-1}), (w_{-1}, w_{+1}), and (w_{+1}, w_{+2}), where the last three correspond to collocations of two consecutive words. These attributes, which are exactly those used in (Ng, 1997a), represent the *local context* of the ambiguous word and they have been proven to be very informative features for WSD. Note that whenever an attribute refers to a

[6] http://www.ldc.upenn.edu/Catalog/LDC97T12.html

word.pos	senses	examples	attributes	*MFS*
age.n	4	493	1662	62.1
art.n	5	405	1557	46.7
car.n	5	1381	4700	95.1
child.n	4	1068	3695	80.9
church.n	4	373	1420	61.1
cost.n	3	1500	4591	87.3
fall.v	19	1500	5063	70.1
head.n	14	870	2502	36.9
interest.n	7	1500	4521	45.1
know.v	8	1500	3965	34.9
line.n	26	1342	4387	21.9
set.v	19	1311	4396	36.9
speak.v	5	517	1873	69.1
take.v	30	1500	5181	35.6
work.n	7	1469	4923	31.7
nouns	8.6	1040.1	3978.5	57.4
verbs	17.9	1265.6	4431.9	46.6
all	12.1	1115.3	4150.0	53.3

Table 3.1: Set of 15 reference words

position that falls beyond the boundaries of the sentence for a certain example, a default value "_" is assigned.

Let $p_{\pm i}$ be the part-of-speech tag[7] of word $w_{\pm i}$, and $c_1, \ldots, c_m$ the unordered set of open class words appearing in the sentence.

- *SetB* enriches the local context: w_{-1}, w_{+1}, (w_{-2}, w_{-1}), (w_{-1}, w_{+1}), (w_{+1}, w_{+2}), (w_{-3}, w_{-2}, w_{-1}), (w_{-2}, w_{-1}, w_{+1}), (w_{-1}, w_{+1}, w_{+2}) and (w_{+1}, w_{+2}, w_{+3}), with the part-of-speech information: p_{-3}, p_{-2}, p_{-1}, p_{+1}, p_{+2}, p_{+3}, and, additionally, it incorporates *broad context* information: $c_1 \ldots c_m$. *SetB* is intended to represent a more realistic set of attributes for WSD[8]. Note that c_i attributes are binary-valued, denoting the the presence or absence of a content word in the sentence context.

[7] We have used the FreeLing tagger (Atserias et al., 2006) to get those tags. See appendix C for Web references.

[8] Actually, it incorporates all the attributes used in (Ng and Lee, 1996), with the exception of the morphology of the target word and the verb-object syntactic relation.

There is an important issue to comment, *SetA* has a constant number of attributes (7), while for *SetB* this number depends on the concrete word (it ranges from 2,641 to 6,428 with an average of 5,036.6 for the 15 words of the DSO corpus selected for section 3.3).

3.3 Adapting Naive-Bayes and Exemplar-Based Algorithms

This section describes an experimental comparison between two standard supervised learning methods, namely Naive Bayes and Exemplar-based classification, on the Word Sense Disambiguation problem. The aim of the work is twofold. First, it attempts to contribute to clarify some confusing information about the comparison between both methods appearing in the related literature. Thus, several directions have been explored, including: testing several modifications of the basic learning algorithms and varying the feature space. Second, an improvement of both algorithms is proposed, in order to deal with large attribute sets. This modification, which basically consists of using only the *positive* information appearing in the examples, allows to greatly improve the efficiency of the methods, with no loss in accuracy. Results show that the Exemplar-based approach to Word Sense Disambiguation is generally superior to the Bayesian approach, especially when a specific metric for dealing with symbolic attributes is used.

3.3.1 Comments about Related Work

Regarding the comparison between Naive Bayes and Exemplar-based methods, the works by Mooney (1996) and Ng (1997a) will be the ones basically referred in this section. Mooney's paper claims that the Bayesian approach is clearly superior to the Exemplar-based approach. Although it is not explicitly said, the overall accuracy of Naive Bayes is about 16 points higher than that of the Example-based algorithm, and the latter is only slightly above the accuracy that a Most Frequent Sense Classifier would obtain. In the Exemplar-based approach, the algorithm applied for classifying new examples was a standard k-NN, using the Hamming distance for measuring closeness. Neither example weighting nor attribute weighting are applied, k is set to 3, and the number of attributes used is said to be almost 3,000. The second paper compares the Naive Bayes approach with *PEBLS* (Cost and Salzberg, 1993), an Exemplar-based learner especially designed for dealing with symbolic features. This paper shows that, for a large number of nearest-neighbours, the performance of both algorithms is comparable, while if cross-validation is used for parameter setting, *PEBLS* slightly outperforms Naive Bayes. It has to be noted that the comparison was carried out in a limited setting, using only

7 feature types, and that the attribute/example-weighting facilities provided by *PEBLS* were not used. The author suggests that the poor results obtained in Mooney's work were due to the metric associated to the k-NN algorithm, but he did not test if the *MVDM* metric used in *PEBLS* is superior to the standard Hamming distance or not.

Another surprising result that appears in Ng's paper is that the accuracy results obtained were 1-1.6% higher than those reported by the same author one year before (Ng and Lee, 1996), when running exactly the same algorithm on the same data, but using a larger and richer set of attributes. This apparently paradoxical difference is attributed, by the author, to the feature pruning process performed in the older paper. Apart from the contradictory results obtained by the previous two papers, some methodological drawbacks of both comparisons should also be pointed out. On the one hand, Ng applies the algorithms on a broad-coverage corpus but reports the accuracy results of a single testing experiment, providing no statistical tests of significance. On the other hand, Mooney performs thorough and rigorous experiments, but he compares the alternative methods on a limited domain consisting of a single word with a reduced set of six senses. Thus, it is our claim that this extremely specific domain does not guarantee reliable conclusions about the relative performances of alternative methods when applied to broad-coverage domains. Consequently, the aim of this section is twofold: 1) To study the source of the differences between both approaches in order to clarify the contradictory and incomplete information. 2) To empirically test the alternative algorithms and their extensions on a broad-coverage sense tagged corpus, in order to estimate which is the most appropriate choice.

3.3.2 Description of the Experiments

Both Naive Bayes and Exemplar-based (during this section $EB_{h,k}$) classifiers have been used as they are explained in section 3.1. All settings of the experiments have been already described in section 3.2.

Using SetA: Local Features

This experiment is devoted to test the Naive Bayes and all the variants of Exemplar-based algorithm on a simple and commonly used set of local features. Table 3.2 shows the results of all methods and variants tested on the 15 reference words, using the $SetA$ set of attributes: Most Frequent Sense (MFS), Naive Bayes (NB), Exemplar-based using Hamming distance (EB_h variants, 4th to 8th columns), and Exemplar-based approach using the MVDM metric (EB_{cs} variants, 9th to 11th columns) are included. The best

			EB_h					EB_{cs}		
word	***MFS***	***NB***	**1**	**7**	**15, *e***	**7, *a***	**7, *e*, *a***	**1**	**10**	**10, *e***
age.n	62.1	73.8	71.4	69.4	71.0	74.4	**75.9**	70.8	73.6	73.6
art.n	46.7	54.8	44.2	59.3	58.3	58.5	57.0	54.1	59.5	**61.0**
car.n	95.1	95.4	91.3	95.5	95.8	96.3	96.2	95.4	**96.8**	**96.8**
child.n	80.9	86.8	82.3	89.3	89.5	91.0	**91.2**	87.5	91.0	90.9
church.n	61.1	62.7	61.9	62.7	63.0	62.5	64.1	61.7	**64.6**	64.3
cost.n	87.3	86.7	81.1	87.9	87.7	**88.1**	87.8	82.5	85.4	84.7
fall.v	70.1	76.5	73.3	78.2	79.0	78.1	79.8	78.7	81.6	**81.9**
head.n	36.9	76.9	70.0	76.5	76.9	77.0	78.7	74.3	78.6	**79.1**
interest.n	45.1	64.5	58.3	62.4	63.3	64.8	66.1	65.1	67.3	**67.4**
know.v	34.9	47.3	42.2	44.3	46.7	44.9	46.8	45.1	49.7	**50.1**
line.n	21.9	51.9	46.1	47.1	49.7	50.7	51.9	53.3	**57.0**	56.9
set.v	36.9	55.8	43.9	53.0	54.8	52.3	54.3	49.7	**56.2**	56.0
speak.v	69.1	**74.3**	64.6	72.2	73.7	71.8	72.9	67.1	72.5	72.9
take.v	35.6	44.8	39.3	43.7	46.1	44.5	46.0	45.3	48.8	**49.1**
work.n	31.7	51.9	42.5	43.7	47.2	48.5	48.9	48.5	52.0	**52.5**
nouns	57.4	71.7	65.8	70.0	71.1	72.1	72.6	70.6	73.6	**73.7**
verbs	46.6	57.6	51.1	56.3	58.1	56.4	58.1	55.9	60.3	**60.5**
all	53.3	66.4	60.2	64.8	66.2	66.1	67.2	65.0	68.6	**68.7**

Table 3.2: Accuracy results of all algorithms on the set of 15 reference words using *SetA*.

result for each word is printed in boldface. From these figures, several conclusions can be drawn:

- All methods significantly outperform the MFS classifier.
- Referring to the EB_h variants, $EB_{h,7}$ performs significantly better than $EB_{h,1}$, confirming the results of Ng (Ng, 1997a) that values of k greater than one are needed in order to achieve good performance with the kNN approach. Additionally, both example weighting ($EB_{h,15,e}$) and attribute weighting ($EB_{h,7,a}$) significantly improve $EB_{h,7}$. Further, the combination of both ($EB_{h,7,e,a}$) achieves an additional improvement
- The MVDM metric is superior to Hamming distance. The accuracy of $EB_{cs,10,e}$ is significantly higher than those of any EB_h variant. Unfortunately, the use of weighted examples does not lead to further improvement in this case. A drawback of using the MVDM metric is the computational overhead introduced by its calculation.

	NB	PNB	$EB_{h,15,e}$	$PEB_{h,7,e}$	$PEB_{h,7,a}$	$PEB_{cs,10,e}$
SetA	00:07		00:08		00:11	09:56
SetB	16:13	00:12	06:04	00:25	03:55	49:43

Table 3.3: CPU-time elapsed on the set of 15 words ("hh:mm").

POS	MFS	NB	PNB	$EB_{h,15}$
nouns	57.4	72.2	72.4	64.3
verbs	46.6	55.2	55.3	43.0
all	53.3	65.8	66.0	56.2

	PEB_h					PEB_{cs}		
POS	**1**	**7**	$7, e$	$7, a$	$10, e, a$	**1**	**10**	$10, e$
nouns	70.6	72.4	73.7	72.5	73.4	73.2	75.4	**75.6**
verbs	54.7	57.7	59.5	58.9	60.2	58.6	61.9	**62.1**
all	64.6	66.8	**68.4**	67.4	68.4	67.7	70.3	**70.5**

Table 3.4: Results of all algorithms on the set of 15 reference words using $SetB$.

Table 3.3 shows that EB_h is fifty times faster than EB_{cs} using $SetA$[9].

- The Exemplar-based approach achieves better results than the Naive Bayes algorithm. This difference is statistically significant when comparing the $EB_{cs,10}$ and $EB_{cs,10,e}$ against NB.

Using SetB: Local and Topical Features

The aim of the experiments with $SetB$ is to test both methods with a realistic large set of features. Table 3.4 summarises the results of these experiments[10]. Let's now consider only NB and EB_h (65.8 and 56.2 accuracy). A very surprising result is observed: while NB achieves almost the same accuracy that in the previous experiment, the exemplar-based approach shows a very low performance. The accuracy of EB_h drops 8.6 points (from 66.0 to 56.2 accuracy) and is only slightly higher than that of MFS.

The problem is that the binary representation of the broad-context attributes is not appropriate for the kNN algorithm. Such a representation leads to an extremely sparse vector representation of the examples, since in each example only a few words, among all

[9]The programs were implemented using PERL-5.003 and they ran on a Sun UltraSPARC-2 machine with 192Mb of RAM.

[10]Detailed results for each word are not included.

possible, are observed. Thus, the examples are represented by a vector of about 5,000 0's and only a few 1's. In this situation two examples will coincide in the majority of the values of the attributes (roughly speaking in "all" the zeros) and will differ in those positions corresponding to 1's with high probability. This fact wrongly biases the similarity measure (and, thus, the classification) in favour of that stored examples which have less 1's, that is, those corresponding to the shortest sentences.

This situation could explain the poor results obtained by the kNN algorithm in Mooney's work, in which a large number of attributes was used. Moreover, it could explain why the results of Ng's system working with a rich attribute set (including binary-valued contextual features) were lower than those obtained with a simpler set of attributes[11].

In order to address this limitation we propose to reduce the attribute space by collapsing all binary attributes $c_1, \ldots, c_m$ in a single set-valued attribute c that contains, for each example, all content words that appear in the sentence. In this setting, the similarity S between two values $V_i = \{w_{i_1}, w_{i_2}, \ldots, w_{i_n}\}$ and $V_j = \{w_{j_1}, w_{j_2}, \ldots, w_{j_m}\}$ can be redefined as: $S(V_i, V_j) = \| V_i \cap V_j \|$, that is, equal to the number of words shared[12].

This approach implies that a test example is classified taking into account the information about the words it contains (*positive* information), but no the information about the words it does not contain. Besides, it allows a very efficient implementation, which will be referred to as PEB (standing for Positive Exemplar-Based).

In the same direction, we have tested the Naive Bayes algorithm combining only the conditional probabilities corresponding to the words that appear in the test examples[13]. This variant is referred to as PNB. The results of both PEB and PNB are included in table 3.4, from which the following conclusions can be drawn.

- The PEB approach reaches good results, improving by 10.6 accuracy points of EB (from 56.2 to 66.8 accuracy). Further, the results obtained significantly outperform those obtained using $SetA$, indicating that the careful addition of richer attributes leads to more accurate classifiers. Additionally, the behaviour of the different variants is similar to that observed when using $SetA$, with the exception that the addition of attribute-weighting to the example-weighting ($PEB_{h,10,e,a}$) seems no longer useful.

[11] Authors attributed the bad results to the absence of attribute weighting and to the attribute pruning, respectively.

[12] This measure is usually known as the *matching coefficient* (Manning and Schütze, 1999). More complex similarity measures, e.g. Jaccard or Dice coefficients, have not been explored.

[13] In this case, efficiency is the improvement.

		accuracy (%)				CPU-time (hh:mm)		
	POS	***MFS***	***PNB***	***PEB$_h$***	***PEB$_{cs}$***	***PNB***	***PEB$_h$***	***PEB$_{cs}$***
SetA	*nouns*	56.4	68.7	68.5	**70.2**	00:33	00:47	92:22
	verbs	48.7	64.8	65.3	**66.4**			
	all	53.2	67.1	67.2	**68.6**			
SetB	*nouns*	56.4	69.2	**70.1**		01:06	01:46	-
	verbs	48.7	63.4	**67.0**	-			
	all	53.2	66.8	**68.8**				

Table 3.5: Global results on the 191-word corpus.

- PNB algorithm is at least as accurate as NB. Table 3.3 shows that the *positive* approach increases greatly the efficiency of the algorithms. The acceleration factor is 80 for NB and 15 for EB_h (the calculation of EB_{cs} variants was simply not feasible working with the attributes of $SetB$).
- NB do not performs better with $SetB$ than with $SetA$. The addition of new features to a NB classifier should be done carefully, due to its sensitiveness to redundant features.
- The Exemplar-based approach achieves better results than the Naive Bayes algorithm. Furthermore, the accuracy of $PEB_{h,7,e}$ is now significantly higher than that of PNB.

Global Results

In order to ensure that the results obtained so far also hold on a realistic broad-coverage domain, the PNB and PEB algorithms have been tested on the whole sense-tagged corpus, using both sets of attributes. This corpus contains about 192,800 examples of 121 nouns and 70 verbs. The average number of senses is 7.2 for nouns, 12.6 for verbs, and 9.2 overall. The average number of training examples is 933.9 for nouns, 938.7 for verbs, and 935.6 overall.

The results obtained are presented in table 3.5. It has to be noted that the results of PEB_{cs} using $SetB$ were not calculated due to the extremely large computational effort required by the algorithm (see table 3.3). Results are coherent to those reported previously, that is:

- Using $SetA$, the Exemplar-based approach using the MVDM metric is significantly superior to the rest.

- Using $SetB$, the Exemplar-based approach using Hamming distance and example weighting significantly outperforms the Bayesian approach. Although the use of the MVDM metric could lead to better results, the current implementation is computationally prohibitive.

- Contrary to the Exemplar-based approach, Naive Bayes does not improve accuracy when moving from $SetA$ to $SetB$, that is, the simple addition of attributes does not guarantee accuracy improvements in the Bayesian framework.

3.3.3 Conclusions

This section has focused on clarifying some contradictory results obtained when comparing Naive Bayes and Exemplar-based approaches to Word Sense Disambiguation. Different alternative algorithms have been tested using two different attribute sets on a large sense-tagged corpus. The experiments carried out show that Exemplar-based algorithms have generally better performance than Naive Bayes, when they are extended with example/attribute weighting, richer metrics, etc. The experiments reported also show that the Exemplar-based approach is very sensitive to the representation of a concrete type of attributes, frequently used in Natural Language problems. To avoid this drawback, an alternative representation of the attributes has been proposed and successfully tested. Furthermore, this representation also improves the efficiency of the algorithms, when using a large set of attributes. The test on the whole corpus allows us to estimate that, in a realistic scenario, the best tradeoff between performance and computational requirements is achieved by using the Positive Exemplar-based algorithm, both local and topical attributes, Hamming distance, and example-weighting.

3.4 Applying the AdaBoost Algorithm

In this section Schapire and Singer's AdaBoost.MH boosting algorithm has been applied to Word Sense Disambiguation. Initial experiments on a set of 15 selected polysemous words show that the boosting approach surpasses Naive Bayes and Exemplar-based approaches. In order to make boosting practical for a real learning domain of thousands of words, several ways of accelerating the algorithm by reducing the feature space are studied. The best variant, which we call LazyBoosting, is tested on the DSO corpus. Again, boosting compares favourably to the other benchmark algorithms.

3.4.1 Description of the Experiments

The setting details of the experiments reported in this section are those of section 3.2. The data applied are the 15 words selected from the DSO corpus and the feature set is that called *SetA*. *AdaBoost.MH* has been compared to Naive Bayes (NB) and Exemplar-Based ($EB_{h,15,e}$) algorithms, using them as baselines (see sections 3.2 and 3.3).

Comparing AdaBoost.MH with the Baselines

Figure 3.4 shows the error rate curve of AdaBoost.MH, averaged over the 15 reference words, and for an increasing number of weak rules per word. This plot shows that the error obtained by AdaBoost.MH is lower than those obtained by NB and EB for a number of rules above 100. It also shows that the error rate decreases slightly and monotonically, as it approaches the maximum number of rules reported[14].

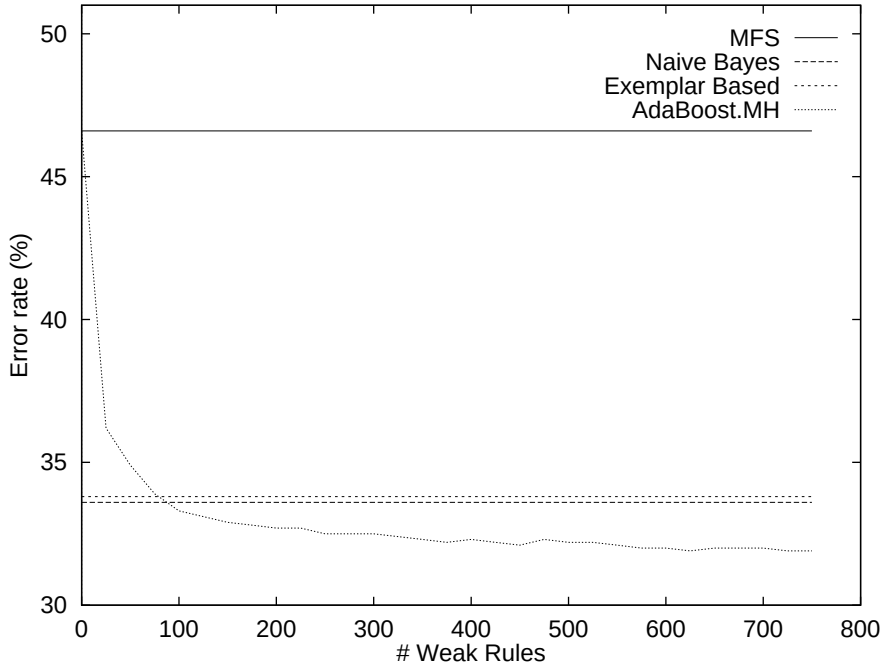

Figure 3.4: Error rate of AdaBoost.MH related to the number of weak rules

According to the plot in figure 3.4, no overfitting is observed while increasing the number of rules per word. Although it seems that the best strategy could be "learn as many rules as possible", the number of rounds must be determined individually for each

[14]The maximum number of rounds considered is 750.

word since they have different behaviour. The adjustment of the number of rounds can be done by cross-validation on the training set, as suggested in (Abney et al., 1999). However, in our case, this cross-validation inside the cross-validation of the general experiment would generate a prohibitive overhead. Instead, a very simple stopping criterion (*sc*) has been used, which consists in stopping the acquisition of weak rules whenever the error rate on the training set falls below 5%, with an upper bound of 750 rules. This variant, which is referred to as AB_{sc}, obtained comparable results to AB_{750} but generating only 370.2 weak rules per word on average, which represents a very moderate storage requirement for the combined classifiers.

The numerical information corresponding to this experiment is included in table 3.6. This table shows the accuracy results, detailed for each word, of NB, $EB_{h,1}$, $EB_{h,15,e}$, AB_{750}, and AB_{sc}. The best result for each word is printed in boldface.

word.pos	$\boldsymbol{MFS}$	$\boldsymbol{NB}$	$\boldsymbol{EB_{h,1}}$	$\boldsymbol{EB_{h,15,e}}$	$\boldsymbol{AB_{750}}$	$\boldsymbol{AB_{sc}}$
age.n	62.1	73.8	71.4	71.0	**74.7**	74.0
art.n	46.7	54.8	44.2	58.3	57.5	**62.2**
car.n	95.1	95.4	91.3	95.8	**96.8**	96.5
child.n	80.9	86.8	82.3	89.5	**92.8**	92.2
church.n	61.1	62.7	61.9	63.0	**66.2**	64.9
cost.n	87.3	86.7	81.1	87.7	87.1	**87.8**
fall.v	70.1	76.5	73.3	79.0	**81.1**	80.6
head.n	36.9	76.9	70.0	76.9	**79.0**	**79.0**
interest.n	45.1	64.5	58.3	63.3	**65.4**	65.1
know.v	34.9	47.3	42.2	46.7	**48.7**	**48.7**
line.n	21.9	51.9	46.1	49.7	**54.8**	54.5
set.v	36.9	**55.8**	43.9	54.8	**55.8**	**55.8**
speak.v	69.1	**74.3**	64.6	73.7	72.2	73.3
take.v	35.6	44.8	39.3	46.1	**46.7**	46.1
work.n	31.7	**51.9**	42.5	47.2	50.7	50.7
nouns	57.4	71.7	65.8	71.1	**73.5**	73.4
verbs	46.6	57.6	51.1	58.1	**59.3**	59.1
all	53.3	66.4	60.2	66.2	**68.1**	68.0

Table 3.6: Set of 15 reference words and results of the main algorithms

As it can be seen, in 14 out of 15 cases, the best results correspond to the boosting algorithm. When comparing global results, accuracies of either AB_{750} or AB_{sc} are significantly higher than those of any of the other methods.

Accelerating Boosting for WSD

Up to now, it has been seen that AdaBoost.MH is a simple and competitive algorithm for the WSD task. It achieves an accuracy performance superior to that of the Naive Bayes and Exemplar-based baselines. However, AdaBoost.MH has the drawback of its computational cost, which makes the algorithm not scale properly to real WSD domains of thousands of words.

The space and time-per-round requirements of AdaBoost.MH are $\mathcal{O}(mk)$ (recall that m is the number of training examples and k the number of senses), not including the call to the weak learner. This cost is unavoidable since AdaBoost.MH is inherently sequential. That is, in order to learn the $(t+1)$-th weak rule it needs the calculation of the t-th weak rule, which properly updates the matrix D_t. Further, inside the *WeakLearner*, there is another iterative process that examines, one by one, all attributes so as to decide which is the one that minimises Z_t. Since there are thousands of attributes, this is also a time consuming part, which can be straightforwardly speed up either by reducing the number of attributes or by relaxing the need to examine all attributes at each iteration.

Four methods have been tested in order to reduce the cost of searching for weak rules. The first three, consisting of aggressively reducing the feature space, are frequently applied in Text Categorisation. The fourth consists in reducing the number of attributes that are examined at each round of the boosting algorithm.

- **Frequency filtering** (*Freq*): This method consists of simply discarding those features corresponding to events that occur less than N times in the training corpus. The idea beyond that criterion is that frequent events are more informative than rare ones and unfrequent features may lead to overfitting.

- **Local frequency filtering** (*LFreq*): This method works similarly to *Freq* but considers the frequency of events locally, at the sense level. More particularly, it selects the N most frequent features of each sense and merging.

- **RLM ranking**: This third method consists in making a ranking of all attributes according to the *RLM* distance measure (de Mántaras, 1991) and selecting the N most relevant features. This measure has been commonly used for attribute selection in decision tree induction algorithms[15].

[15] *RLM* distance belongs to the distance-based and information-based families of attribute selection functions. It has been selected because it showed better performance than seven other alternatives in an experiment of decision tree induction for POS tagging (Màrquez, 1999).

- **LazyBoosting**: The last method does not filter out any attribute but reduces the number of those that are examined at each iteration of the boosting algorithm. More specifically, a small proportion p of attributes are randomly selected and the best weak rule is selected among them. The idea behind this method is that if the proportion p is not too small, probably a sufficiently good rule can be found at each iteration. Besides, the chance for a good rule to appear in the whole learning process is very high. Another important characteristic is that no attribute needs to be discarded and so we avoid the risk of eliminating relevant attributes. This approach has a main drawback: boosting changes the weight distribution at each iteration. Perhaps a good feature will loose importance at the iteration is selected due to this weight distribution. Because this feature can be not important to the examples with highest weight. This method will be called *LazyBoosting* in reference to the work by Samuel and colleagues (Samuel, 1998). They applied the same technique for accelerating the learning algorithm in a Dialogue Act tagging system.

The four methods above have been compared for the set of 15 reference words. Figure 3.5 contains the average error-rate curves obtained by the four variants at increasing levels of attribute reduction. The top horizontal line corresponds to the *MFS* error rate, while the bottom horizontal line stands for the error rate of AdaBoost.MH working with all attributes. The results contained in figure 3.5 are calculated running the boosting algorithm 250 rounds for each word.

The main conclusions that can be drawn are the following:

- All methods seem to work quite well since no important degradation is observed in performance for values lower than 95% in rejected attributes. This may indicate that there are many irrelevant or highly dependent attributes in our domain; and that concepts depend only on a small set of features. All attribute filtering methods capture the really important features.

- *LFreq* is slightly better than *Freq*, indicating a preference to make frequency counts for each sense rather than globally.

- The more informed *RLM* ranking performs better than frequency-based reduction methods *Freq* and *LFreq*.

- *LazyBoosting* is better than all other methods, confirming our expectations: it is worth keeping all information provided by the features. In this case, acceptable performance is obtained even if only 1% of the attributes is explored when looking for a weak rule. The value of 10%, for which *LazyBoosting* still achieves the same

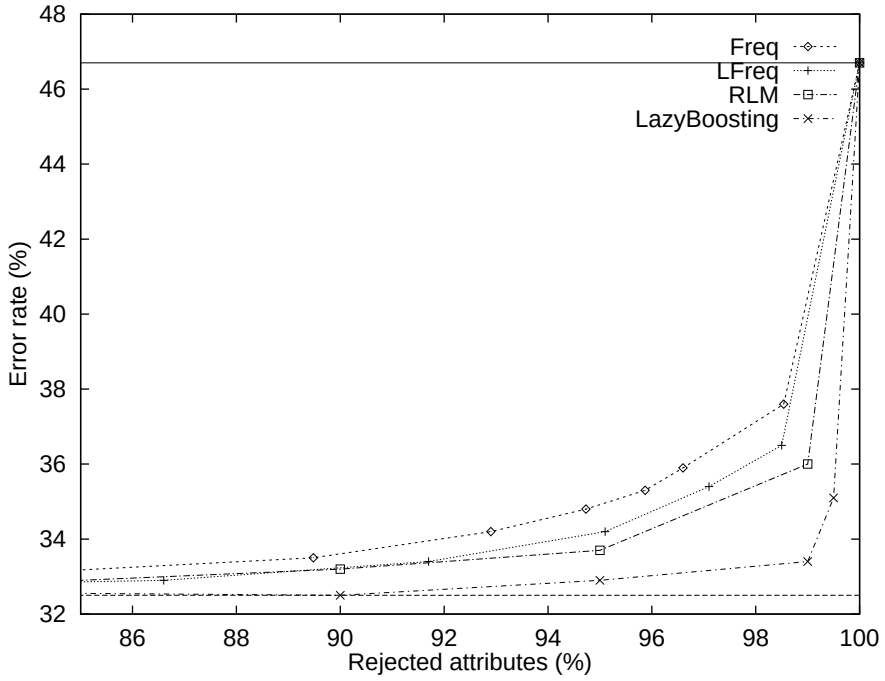

Figure 3.5: Error rate obtained by the four methods, at 250 weak rules per word, with respect to the percentage of rejected attributes

performance and runs about 7 times faster than AdaBoost.MH working with all attributes, will be selected for the experiments in next section.

Evaluating LazyBoosting

The *LazyBoosting* algorithm has been tested on the full DSO (all 191 words) with $p = 10\%$ and the same stopping criterion described above, which will be referred to as AB_{l10sc}.

The AB_{l10sc} algorithm learned an average of 381.1 rules per word, and took about 4 days of CPU time to complete[16]. It has to be noted that this time includes the cross-validation overhead. Eliminating it, it is estimated that 4 CPU days would be the necessary time for acquiring a word sense disambiguation boosting-based system covering about 2,000 words.

The AB_{l10sc} has been compared again to the benchmark algorithms using the 10-fold cross-validation methodology described in section 3.2. The average accuracy results are reported in the upper-side of table 3.7. The best figures correspond to the *LazyBoosting*

[16]The implementation was written in PERL-5.003 and it was run on a SUN UltraSparc2 machine with 194Mb of RAM.

algorithm AB_{l10sc}, and again, the differences are statistically significant using the 10-fold cross-validation paired t-test.

	accuracy (%)			
	MFS	***NB***	$\boldsymbol{EB_{15}}$	$\boldsymbol{AB_{l10sc}}$
nouns (121)	56.4	68.7	68.0	**70.8**
verbs (70)	46.7	64.8	64.9	**67.5**
all (191)	52.3	67.1	66.7	**69.5**

	wins-ties-losses	
	$\boldsymbol{AB_{l10sc}}$ **vs** $\boldsymbol{NB}$	$\boldsymbol{AB_{l10sc}}$ **vs** $\boldsymbol{EB_{15}}$
nouns (121)	99(51)-1-21(3)	100(68)-5-16(1)
verbs (70)	63(35)-1-6(2)	64(39)-2-4(0)
all (191)	162(86)-2-27(5)	164(107)-7-20(1)

Table 3.7: Results of *LazyBoosting* and the benchmark methods on the 191-word corpus

The lower-side of the table shows the comparison of AB_{l10sc} versus NB and EB_{15} algorithms, respectively. Each cell contains the number of *wins*, *ties*, and *losses* of competing algorithms. The counts of statistically significant differences are included in brackets. It is important to point out that EB_{15} only beats significantly AB_{l10sc} in one case while NB does so in five cases. Conversely, a significant superiority of AB_{l10sc} over EB_{15} and NB is observed in 107 and 86 cases, respectively.

3.4.2 Conclusions

In this section, Schapire and Singer's AdaBoost.MH algorithm has been evaluated on the Word Sense Disambiguation task. As it has been shown, the boosting approach outperforms Naive Bayes and Exemplar-based learning. In addition, a faster variant has been suggested and tested, which is called LazyBoosting. This variant scales the algorithm to broad-coverage real Word Sense Disambiguation domains, and is as accurate as AdaBoost.MH.

3.5 Comparison of Machine Learning Methods

This section is devoted to compare the five algorithms described in section 3.1. That is: Naive Bayes (NB), Exemplar-based (kNN), Decision Lists (DL), AdaBoost.MH (AB) and Support Vector Machines (SVM). We experimentally evaluated and compared them on the DSO corpus.

3.5.1 Setting of Evaluation

From the 191 words represented in the DSO corpus, a group of 21 words which frequently appear in the WSD literature has been selected to perform the comparative experiment. These words are 13 nouns (*age, art, body, car, child, cost, head, interest, line, point, state, thing, work*) and 8 verbs (*become, fall, grow, lose, set, speak, strike, tell*) and they have been treated as independent classification problems. The number of examples per word ranges from 202 to 1,482 with an average of 801.1 examples per word (840.6 for nouns and 737.0 for verbs). The level of ambiguity is quite high in this corpus. The number of senses is between 3 and 25, with an average of 10.1 senses per word (8.9 for nouns and 12.1 for verbs).

This comparison has $SetB$ as feature set (see section 3.2.2). The different methods tested translate this information into features in different ways. The AdaBoost and SVM algorithms require binary features. Therefore, local context attributes have to be binarised in a preprocess, while the topical context attributes remain as binary tests about the presence or absence of a concrete word in the sentence. As a result of this binarisation, the number of features is expanded to several thousands (from 1,764 to 9,900 depending on the particular word).

The binary representation of features is not appropriate for Naive Bayes and kNN algorithms. Therefore, the 15 local-context attributes are considered straightforwardly. Regarding the binary topical-context attributes, the variants described in section 3.3 are considered. For kNN, the topical information is codified as a single set-valued attribute (containing all words appearing in the sentence) and the calculation of closeness is modified so as to handle this type of attributes. For Naive Bayes, the topical context is conserved as binary features, but when classifying new examples only the information of words appearing in the example (positive information) is taken into account.

Regarding to kNN, we do not use MVDM here for simplicity reasons. Its implementation has a very serious computational overhead and the difference in performance is reduced when using the Hamming distance jointly with feature and example weighting, as we do in this setting (see section 3.3).

We have performed a 10-fold cross-validation experiment in order to estimate the performance of the systems. The accuracy figures reported below are (micro) averaged over the results of the 10 folds and over the results on each of the 21 words. We have applied a statistical test of significance, which is a paired Student's t-test with a confidence value of $t_{9,0.995}$=3.250 (see appendix A for all the formulae about evaluation). When classifying test examples, all methods are forced to output a unique sense, resolving potential ties between senses by choosing the most frequent sense among all those tied.

	MFC	**NB**	***k*NN**	**DL**	**AB**	**SVM**
nouns	46.59±1.08	62.29±1.25	63.17±0.84	61.79±0.95	66.00±1.47	**66.80**±1.18
verbs	46.49±1.37	60.18±1.64	64.37±1.63	60.52±1.96	66.91±2.25	**67.54**±1.75
all	46.55±0.71	61.55±1.04	63.59±0.80	61.34±0.93	66.32±1.34	**67.06**±0.65

Table 3.8: Accuracy and standard deviation of all learning methods

3.5.2 Discussion

Table 3.8 presents the results (accuracy and standard deviation) of all methods in the reference corpus. MFC stands for a *Most Frequent Sense* classifier. Averaged results are presented for nouns, verbs, and overall and the best results are printed in boldface.

All methods significantly outperform the baseline Most Frequent Classifier, obtaining accuracy gains between 15 and 20.5 points. The best performing methods are SVM and AdaBoost (SVM achieves a slightly better accuracy but this difference is not statistically significant). On the other extreme, Naive Bayes and Decision Lists are the methods with the lowest accuracy with no significant differences among them. The kNN method is in the middle of the previous two groups. That is, according to the paired t-test the partial order between methods that follows is:

$$SVM \approx AB > kNN > NB \approx DLs > MFC$$

where '$\approx$' means that the accuracies of both methods are not significantly different, and '$>$' means that the left method accuracy is significantly better than the right one.

The low results of DL seem to contradict some previous research, in which very good results were obtained with this method. One possible reason for this failure is the simple smoothing method applied. Yarowsky (1995a) showed that smoothing techniques can help to obtain good estimates for different feature types, which is crucial for methods like DL. These techniques are applied to different learning methods in (Agirre and Martínez, 2004a), and they report an improvement of 4% in recall over the simple smoothing. Another reason for the low performance is that when DL is forced to make decisions with few data points it does not make reliable predictions. Rather than trying to force 100% coverage, the DL paradigm seems to be more appropriate for obtaining high precision estimates. In (Martínez et al., 2002) DL are shown to have a very high precision for low coverage, achieving 94.90% accuracy at 9.66% coverage, and 92.79% accuracy at 20.44% coverage. These experiments were performed on the Senseval-2 datasets.

Regarding standard deviation between experiments in the cross-validation framework, SVM has shown the most stable behaviour, achieving the lowest values.

	<35	**35-60**	**60-120**	**120-200**	**>200**
AB	60.19%	57.40%	**70.21%**	**65.23%**	**73.93%**
SVM	**63.59%**	**60.18%**	**70.15%**	64.93%	72.90 %

Table 3.9: Overall accuracy of AB and SVM classifiers by groups of words of increasing average number of examples per sense

In this corpus subset, the average accuracy values achieved for nouns and verbs are very close, being the baseline MFC results almost identical (46.59% for nouns and 46.49% for verbs). This is quite different from the results reported in many papers taking into account the whole set of 191 words of the DSO corpus. For instance, differences between 3 and 4 points can be observed in favour of nouns in previous section. This is due to the singularities of the subset of 13 nouns studied here, which are particularly difficult. Note that in the whole DSO corpus the MFC over nouns (56.4%) is fairly higher than in this subset (46.6%) and that an AdaBoost-based system is able to achieve 70.8% on nouns in previous section compared to the 66.0% in this section. Also, the average number of senses per noun is higher than in the entire corpus. Despite this fact, a difference between two groups of methods can be observed regarding the accuracy on nouns and verbs. On the one hand, the worst performing methods (NB and DLs) do better on nouns than on verbs. On the other hand, the best performing methods (kNN, AB, and SVM) are able to better learn the behaviour of verb examples, achieving an accuracy value around 1 point higher than for nouns.

Some researchers, Schapire (2002) for instance, argue that the AdaBoost algorithm may perform very poorly when training using small samples. In order to verify this statement in the present study we calculated the accuracy obtained by AdaBoost in several groups of words sorted by increasing *size* of training set. The size of a training set is taken as the ratio between the number of examples and the number of senses of that word, that is, the average number of examples per sense. Table 3.9 shows the results obtained, including a comparison with the SVM method. As expected, the accuracy of SVM is significantly superior than that of AdaBoost for *small* training sets (up to 60 examples per sense). On the contrary, AdaBoost outperforms SVM in the *large* training sets (over 120 examples per sense). Recall that the overall accuracy is comparable in both classifiers (see table 3.8).

In absolute terms, the overall results of all methods can be considered quite low (61-67%). We do not claim that these results can not be improved by using richer feature representations, by a more accurate tuning of the systems, or by the addition of more training examples. Additionally, it is known that the DSO words included in this study are among the most polysemous English words and that WordNet is a very fine-grained sense

	DSO	**MFC**	**NB**	***k*NN**	**DL**	**AB**	**SVM**
DSO	–	46.6	61.5	63.6	61.3	66.3	67.1
MFC	-0.19	–	73.9	58.9	64.9	54.9	57.3
NB	0.24	-0.09	–	75.2	76.7	71.4	76.7
kNN	0.39	-0.15	0.43	–	70.2	72.3	78.0
DL	0.31	-0.13	0.39	0.40	–	69.9	72.5
AB	0.44	-0.17	0.37	0.50	0.42	–	80.3
SVM	0.44	-0.16	0.49	0.61	0.45	0.65	–

Table 3.10: Kappa statistic (below diagonal) and percentage of agreement (above diagonal) between all pairs of systems on the DSO corpus

repository. Supposing that we had enough training examples for every ambiguous word in the language, it is reasonable to think that a much more accurate all-words system could be constructed based on the current supervised technology. However, this assumption is totally unrealistic and, actually, the best current supervised systems for English all-words disambiguation achieve figures around 65% (see Senseval-3 results). Our opinion is that state-of-the-art supervised systems still have to be *qualitatively* improved in order to be really practical.

Apart from accuracy figures, the observation of the predictions made by the classifiers provides interesting information about the comparison between methods. Table 3.10 presents the percentage of agreement and the Kappa statistic between all pairs of systems on the test sets. 'DSO' stands for the annotation of the DSO corpus, which is taken as the correct annotation. Therefore, the agreement rates with respect to DSO contain the accuracy results previously reported. The Kappa statistic (Cohen, 1960) is a measure of inter-annotator agreement, which reduces the effect of chance agreement, and which has been used for measuring inter-annotator agreement during the construction of some semantically annotated corpora (Véronis, 1998; Ng et al., 1999). A Kappa value of 1 indicates perfect agreement, values around 0.8 are considered to indicate very good agreement (Carletta, 1996), and negative values are interpreted as systematic disagreement on non-frequent cases.

NB obtains the most similar results with regard to MFC in agreement rate and Kappa value. A 73.9% of agreement means that almost 3 out of 4 times it predicts the most frequent sense (which is correct in less than half of the cases). SVM and AB obtain the most similar results with regard to DSO (the manual annotation) in agreement rate and Kappa values, and they have the less similar Kappa and agreement values with regard to MFC. This indicates that SVM and AB are the methods that best learn the behaviour of the DSO examples. It is also interesting to note that the three highest values of kappa

(0.65, 0.61, and 0.50) are between the top performing methods (SVM, AB, and kNN), and that, despite NB and DLs achieve a very similar accuracy on the corpus, their predictions are quite different, since the Kappa value between them is one of the lowest (0.39).

The Kappa values between the methods and the DSO annotation are very low. But, as it is suggested in (Véronis, 1998), evaluation measures should be computed relative to the agreement between the human annotators of the corpus and not to a theoretical 100%. It seems pointless to expect more agreement between the system and the reference corpus than between the annotators themselves. Contrary to the intuition that the agreement between human annotators could be very high in the Word Sense Disambiguation task, some papers report surprisingly low figures. For instance, Ng et al. (1999) report an accuracy rate of 56.7% and a Kappa value of 0.317 when comparing the annotation of a subset of the DSO corpus performed by two independent research groups. Similarly, Véronis (1998) reports values of Kappa near to zero when annotating some special words for the ROMANSEVAL[17] corpus.

From this point of view, the Kappa value of 0.44 achieved by SVM and AB could be considered pretty high result. Unfortunately, the subset of the DSO corpora treated in this work does not coincide to that in (Ng et al., 1999) and, therefore, a direct comparison is not possible.

3.6 Conclusions

In section 3.5 a comparison between five of the State-of-the-Art algorithms used in the Word Sense Disambiguation task have been performed. The systems' performance has been compared in terms of accuracy, but other measures, agreement and kappa values between each pair of methods, have been used to compare systems' outputs. With all this data a deeper study of the behaviour of the algorithms is possible. Among the five algorithms compared, the best performing algorithms (in terms of accuracy and behaviour on all senses to be learned) for the Word Sense Disambiguation task are both AdaBoost.MH and Support Vector Machines: SVM for training sets with small number of examples per sense (up to 60) and AdaBoost for larger sets (over 120). Up to now, there is no large sense-tagged corpus available. So, probably the best system could be a combination of both systems, depending on the number of examples per sense for each word. That is, train each word with one of the algorithms depending on the number of examples per sense.

[17] http://www.lpl.univ-aix.fr/projectes/romanseval

Another important question to highlight is the behaviour of the algorithms in front of different sets of features. Section 3.3 shows important differences between Naive Bayes and kNN respect to the topical features, this fact implies that the best set of features is different for each method. We have used as set of features those most used by the community. But, a deeper study on this issue is needed in order to fully understand the differences of behaviour (section 4 of chapter 6 is devoted to this issue).

Chapter 4

Domain Dependence

This chapter describes a set of experiments carried out to explore the domain dependence of alternative supervised Word Sense Disambiguation algorithms. The aim of the work is: studying the performance of these algorithms when tested on a different corpus from that they were trained on; and exploring their ability to tune to new domains. As we will see, the domain dependence of Word Sense Disambiguation systems seems very strong and suggests that some kind of adaptation or tuning is required for cross-corpus applications.

4.1 Introduction

Supervised learning assumes that the training examples are somehow reflective of the task that will be performed by the trainee on other data. Consequently, the performance of such systems is commonly estimated by testing the algorithm on a separate part of the set of training examples (say 10-20% of them), or by N-fold cross-validation, in which the set of examples is partitioned into N disjoint sets (or folds), and the training-test procedure is repeated N times using all combinations of $N-1$ folds for training and 1 fold for testing. In both cases, test examples are different from those used for training, but they belong to the same corpus, and, therefore, they are expected to be quite similar. Although this methodology could be valid for certain NLP problems, such as English Part-of-Speech tagging, we think that there exists reasonable evidence to say that, in WSD, accuracy results cannot be simply extrapolated to other domains (contrary to the opinion of other authors Ng (1997b)): On the one hand, WSD is very dependant to the domain of application Gale et al. (1992a) –see also Ng and Lee (1996); Ng (1997a), in which quite different accuracy figures are obtained when testing an exemplar-based WSD classifier on two different corpora. On the other hand, it does not seem reasonable to

think that the training material can be large and representative enough to cover "all" potential types of examples. We think that an study of the domain dependence of WSD –in the style of other studies devoted to parsing Sekine (1997)– is needed to assess the validity of the supervised approach, and to determine to which extent a tuning process is necessary to make real WSD systems portable to new corpora. In order to corroborate the previous hypotheses, this chapter explores the portability and tuning of four different ML algorithms (previously applied to WSD) by training and testing them on different corpora. Additionally, supervised methods suffer from the "knowledge acquisition bottleneck" Gale et al. (1992b). Ng (1997b) estimates that the manual annotation effort necessary to build a broad coverage semantically annotated English corpus is about 16 person-years. This overhead for supervision could be much greater if a costly tuning procedure is required before applying any existing system to each new domain.

Due to this fact, recent works have focused on reducing the acquisition cost as well as the need for supervision in corpus-based methods. It is our belief that the research by Leacock et al. (1998); Mihalcea and Moldovan (1999b)[1] provide enough evidence towards the "opening" of the bottleneck in the near future. For that reason, it is worth further investigating the robustness and portability of existing supervised ML methods to better resolve the WSD problem. It is important to note that the focus of this work will be on the empirical cross-corpus evaluation of several ML supervised algorithms. Other important issues, such as: selecting the best attribute set, discussing an appropriate definition of senses for the task, etc., are not addressed in this chapter.

4.2 Setting

The set of comparative experiments has been carried out on a subset of 21 words of the DSO corpus. Each word is treated as a different classification problem. They are 13 nouns (*age*, *art*, *body*, *car*, *child*, *cost*, *head*, *interest*, *line*, *point*, *state*, *thing*, *work*) and 8 verbs (*become*, *fall*, *grow*, *lose*, *set*, *speak*, *strike*, *tell*)[2]. The average number of senses per word is close to 10 and the number of training examples is close to 1,000.

The DSO corpus contains sentences from two different corpora. The first is Wall Street Journal (WSJ), which belongs to the financial domain and the second is Brown Corpus (BC) being a general corpora of English usage. Therefore, it is easy to perform experiments about the portability of alternative systems by training them on the WSJ part (*A* part, hereinafter) and testing them on the BC part (B part, hereinafter), or vice-versa.

[1] In the line of using lexical resources and search engines to automatically collect training examples from large text collections or the Web.

[2] This set of words is the same used in the final comparison of the previous chapter (section 3.5).

Two kinds of information are used to train classifiers: local and topical context. The former consists of the words and part-of-speech tags appearing in a window of ± 3 items around the target word, and collocations of up to three consecutive words in the same window. The latter consists of the unordered set of content words appearing in the whole sentence[3].

The four methods tested (Naive Bayes, Exemplar-Based, Decision Lists, and LazyBoosting) translate this information into features in different ways. LazyBoosting algorithm requires binary features. Therefore, local context attributes have to be binarised in a preprocess, while the topical context attributes remain as binary tests about the presence/absence of a concrete word in the sentence. As a result the number of attributes is expanded to several thousands (from 1,764 to 9,900 depending on the particular word). The binary representation of attributes is not appropriate for Naive Bayes and Exemplar-Based algorithms. Therefore, the 15 local-context attributes are taken straightforwardly. Regarding the binary topical-context attributes, we have used the variants described in chapter 3. For Exemplar-Based, the topical information is codified as a single set-valued attribute (containing all words appearing in the sentence) and the calculation of the closeness measure is modified so as to handle this type of attribute. For Naive Bayes, the topical context is conserved as binary features, but when classifying new examples only the information of words appearing in the example (positive information) is taken into account.

4.3 First Experiment: Across Corpora evaluation

The comparison of algorithms has been performed in series of controlled experiments using exactly the same training and test sets. There are 7 combinations of training-test sets called: $A{+}B{\rightarrow}A{+}B$, $A{+}B{\rightarrow}A$, $A{+}B{\rightarrow}B$, $A{\rightarrow}A$, $B{\rightarrow}B$, $A{\rightarrow}B$, and $B{\rightarrow}B$, respectively. In this notation, the training set is placed at the left hand side of symbol "→", while the test set is at the right hand side. For instance, $A{\rightarrow}B$ means that the training set is corpus A and the test set is corpus B. The symbol "+" stands for set union, therefore $A{+}B{\rightarrow}B$ means that the training set is A union B and the test set is B. When comparing the performance of two algorithms, two different statistical tests of significance have been applied depending on the case. $A{\rightarrow}B$ and $B{\rightarrow}A$ combinations represent a single training-test experiment. In this cases, the McNemar's test of significance is used (with a confidence value of: $\chi^2_{1,0.95} = 3.842$), which is proven to be more robust than a simple test for the difference of two proportions. In the other combinations, a 10-fold cross-validation was performed in

[3]This set of features is the $SetB$ defined in section 3.2.2.

order to prevent testing on the same material used for training. The associated statistical tests of significance is a paired Student's t-test with a confidence value of: $t_{9,0.975} = 2.262$ (see appendix A for evaluation formulae).

The four algorithms, jointly with a naive Most-Frequent-sense Classifier (MFC), have been tested on 7 different combinations of training-test sets. Accuracy figures, averaged over the 21 words, are reported in table 4.1. The comparison leads to the following conclusions:

	MFC	**NB**	**EB**	**DL**	**LB**
A+B→A+B	46.55±0.71	61.55±1.04	63.01±0.93	61.58±0.98	**66.32**±1.34
A+B→A	53.90±2.01	67.25±1.07	69.08±1.66	67.64±0.94	**71.79**±1.51
A+B→B	39.21±1.90	55.85±1.81	56.97±1.22	55.53±1.85	**60.85**±1.81
A→A	55.94±1.10	65.86±1.11	68.98±1.06	67.57±1.44	**71.26**±1.15
B→B	45.52±1.27	56.80±1.12	57.36±1.68	56.56±1.59	**58.96**±1.86
A→B	36.40	41.38	45.32	43.01	**47.10**
B→A	38.71	47.66	51.13	48.83	**51.99***

Table 4.1: Accuracy results (± standard deviation) of the methods on all training-test combinations

- LazyBoosting outperforms all other methods in all cases. Additionally, this superiority is statistically significant, except when comparing LazyBoosting to the Exemplar-Based approach in 5 of the 21 cases.

- Surprisingly, LazyBoosting in A+B→A (or A+B→B) does not achieve substantial improvement to the results of A→A (or B→B). Actually, the first variation is not statistically significant and the second is only slightly significant. That is, the addition of extra examples from another domain does not necessarily contribute to improve the results on the original corpus.

- Regarding the portability of the systems, very disappointing results are obtained. Restricting to LazyBoosting results, we observe that the accuracy obtained in A→B is 47.1% while the accuracy in B→B (which can be considered an upper bound for LazyBoosting in B corpus) is 59.0%, that is, a drop of 12 points. Furthermore, 47.1% is only slightly better than the most frequent sense in corpus B, 45.5%. The comparison in the reverse direction is even worse: a drop from 71.3% (A→A) to 52.0% (B→A), which is lower than the most frequent sense of corpus A, 55.9%.

4.4 Second Experiment: tuning to new domains

The previous experiment shows that classifiers trained on the A corpus do not work well on the B corpus, and vice-versa. Therefore, it seems that some kind of tuning process is necessary to adapt supervised systems to each new domain. This experiment explores the effect of a simple tuning process consisting of adding to the original training set a relatively small sample of manually sense tagged examples of the new domain. The size of this supervised portion varies from 10% to 50% of the available corpus in steps of 10% (the remaining 50% is kept for testing). This set of experiments will be referred to as $A+\%B \rightarrow B$, or conversely, $B+\%A \rightarrow A$.

In order to determine to which extent the original training set contributes to accurately disambiguate in the new domain, we also calculate the results for $\%A \rightarrow A$ (and $\%B \rightarrow B$), that is, using only the tuning corpus for training.

Figure 4.1 graphically presents the results obtained by all methods. Each plot contains $X+\%Y \rightarrow Y$ and $\%Y \rightarrow Y$ curves, and the straight lines corresponding to the lower bound MFC, and to the upper bounds $Y \rightarrow Y$and $X+Y \rightarrow Y$.

As expected, the accuracy of all methods grows (towards the upper bound) as more tuning corpus is added to the training set. However, the relation between $X+\%Y \rightarrow Y$ and $\%Y \rightarrow Y$ reveals some interesting facts. In plots 1b, 2a, and 3b the contribution of the original training corpus is almost negligible. Furthermore, in plots 1a, 2b, and 3a a degradation on the accuracy performance is observed. Summarising, these six plots show that for Naive Bayes, Exemplar-Based and Decision Lists methods it is not worth keeping the original training examples. Instead, a better (but disappointing) strategy would be simply using the tuning corpus.

However, this is not the situation of LazyBoosting (plots 4a and 4b), for which a moderate (but consistent) improvement of accuracy is observed when retaining the original training set. Therefore, LazyBoosting shows again a better behaviour than their competitors when moving from one domain to another.

4.5 Third Experiment: training data quality

The bad results about portability may be explained by, at least, two reasons: 1) Corpus A and B have a very different distribution of senses, and, therefore, different a-priori biases; 2) Examples of corpus A and B contain different information, and, therefore, the learning algorithms acquire different (and non interchangeable) classification cues from both corpora.

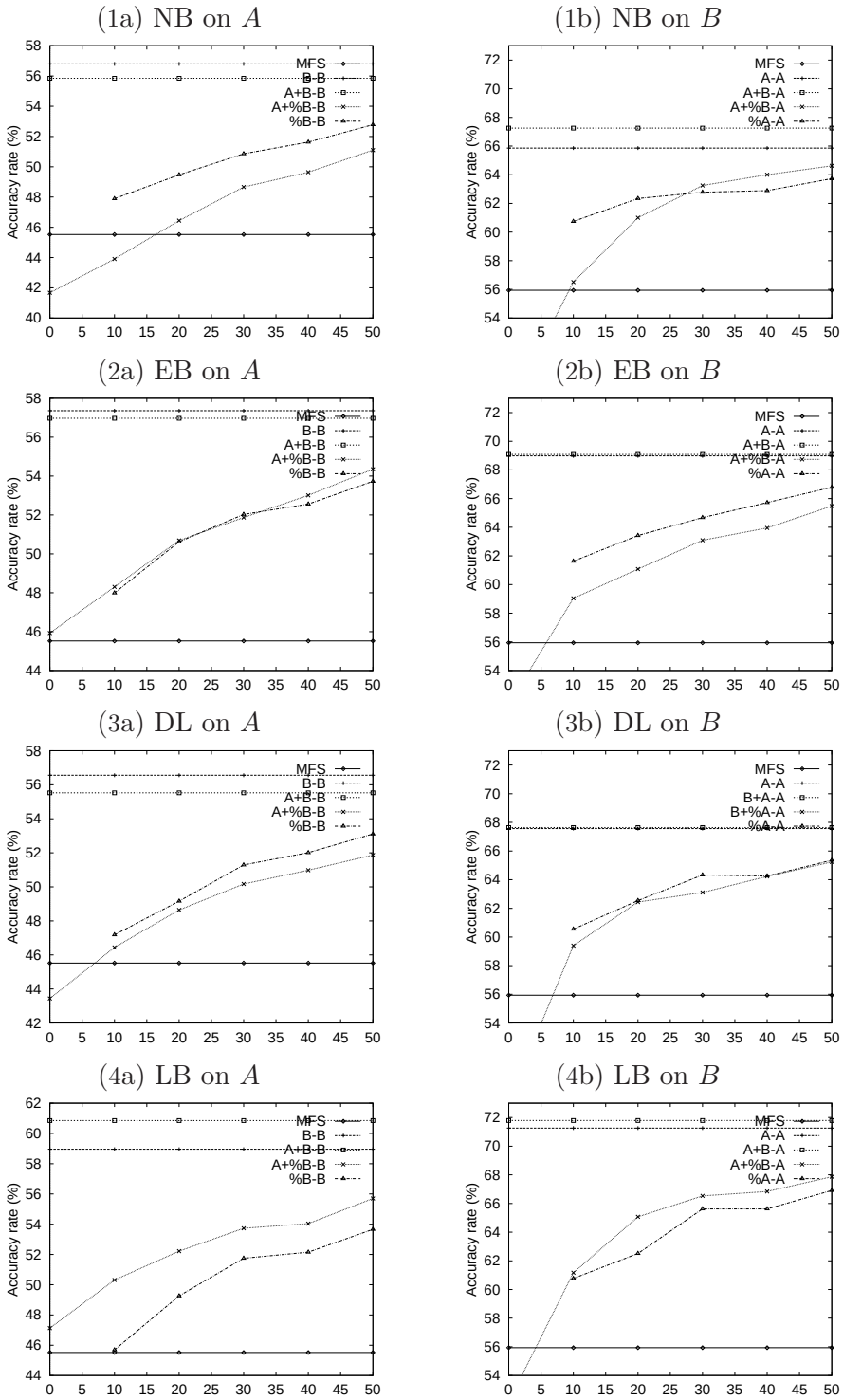

Figure 4.1: Results of the tuning experiment

The first hypothesis is confirmed by observing the bar plots of figure 4.2, which contain the distribution of the four most frequent senses of some sample words in the corpora A and B, respectively. In order to check the second hypothesis, two new sense-balanced corpora have been generated from the DSO corpus, by equilibrating the number of examples of each sense between A and B parts. In this way, the first difficulty is artificially overrided and the algorithms should be portable if examples of both parts are quite similar. Table 4.2 shows the results obtained by LazyBoosting on these new corpora.

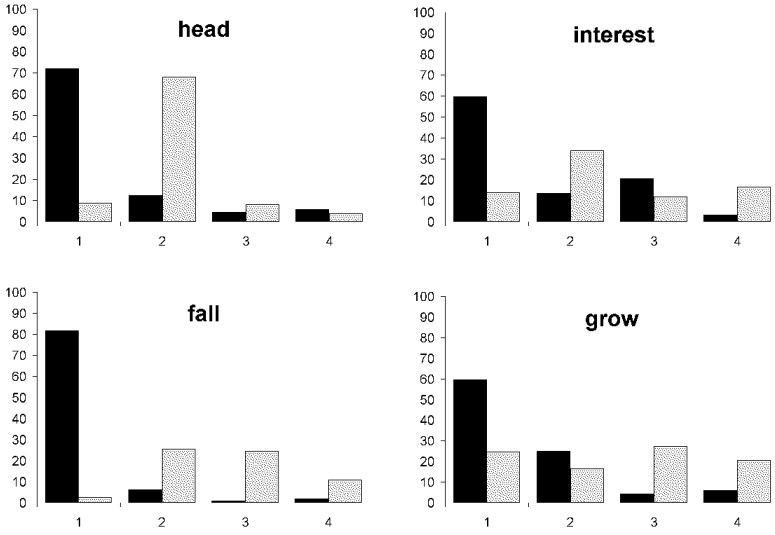

Figure 4.2: Distribution of the four most frequent senses for two nouns (head, interest) and two verbs (fall, grow). Black bars = A corpus; Grey bars = B corpus

Regarding portability, we observe a significant accuracy decrease of 7 and 5 points from $A \rightarrow A$ to $B \rightarrow A$, and from $B \rightarrow B$ to $A \rightarrow B$, respectively. This loss in accuracy is not as important as in the first experiment, due to the simplification provided by the balancing of sense distributions. That is, even when the same distribution of senses is conserved between training and test examples, the portability of the supervised WSD systems is not guaranteed. These results imply that examples have to be largely different from one corpus to another.

The observation of the rules acquired by LazyBoosting also could help improving data quality. It is known that misslabelled examples resulting from annotation errors tend to be hard examples to classify correctly, and, therefore, tend to have large weights in the final distribution. Abney et al. (1999) applied this idea from AdaBoost to PoS tagging, improving the data quality. This observation allows both to identify the noisy examples and use LB as a way to improve the training corpus. An experiment has been carried

	MFC	**LB**
$A+B \rightarrow A+B$	48.55±1.16	64.35±1.16
$A+B \rightarrow A$	48.64±1.04	66.20±2.12
$A+B \rightarrow B$	48.46±1.21	62.50±1.47
$A \rightarrow A$	48.62±1.09	65.22±1.50
$B \rightarrow B$	48.46±1.21	61.74±1.18
$A \rightarrow B$	48.70	56.12
$B \rightarrow A$	48.70	58.05

Table 4.2: Accuracy results (± standard deviation) of LazyBoosting on the sense-balanced corpora

out in this direction by studying the rules acquired by LB from the training examples of word *state*. The manually inspection of the 50 highest scored rules allowed us to identify a high number of noisy training examples -there were 11 tagging errors-, and, additionally, 17 examples not coherently tagged, probably due to the too fine grained or not so clear distinctions between the senses involved in these examples. Thus, there were 28 of 50 examples with some kind of problem among the best scored.

See, for instance, the two rules presented in table 4.3, acquired from corpus A and B respectively, which account for the collocation *state court*. For each rule and each sense there are two real numbers, which are the rule outputs when the feature holds in the example and when it does not, respectively. The sign of real numbers is interpreted as a prediction as to whether the sense should or should not be assigned to the example. The magnitude of the prediction is interpreted as a measure of confidence in the prediction. Therefore, according to the first rule, the presence of the collocation *state court* gives positive evidence to the sense number 3 and negative to the rest of senses. On the contrary, according to the second rule the presence of such collocation would contribute to sense number 5 and negatively to all the rest (including sense number 3).

		senses					
	source	**0**	**1**	**2**	**3**	**4**	**5**
rule 1	A	-0.5064	-1.0970	-0.6992	**1.2580**	-1.0125	-0.4721
		0.0035	0.0118	0.0066	-0.0421	0.0129	0.0070
rule 2	B	-0.0696	-0.2988	-0.1476	-1.1580	-0.5095	**1.2326**
		-0.0213	0.0019	-0.0037	0.0102	0.0027	-0.0094

Table 4.3: Example of WeakRule: it is referred to the collocation *state court*

It has to be noted that these rules are completely coherent with the senses assigned

to the examples containing *state court* in both corpora, and therefore, the contradiction comes from different information that characterise examples of both corpora.

4.6 Conclusions

This work has pointed out some difficulties regarding the portability of supervised WSD systems, a very important issue that has been paid little attention up to the present. According to our experiments, it seems that the performance of supervised sense taggers is not guaranteed when moving from one domain to another (e.g. from a balanced corpus, such as BC, to an economic domain, such as WSJ). These results implies that some kind of adaptation is required for cross-corpus application. Furthermore, these results are in contradiction with the idea of "robust broad-coverage WSD" introduced by Ng (1997b), in which a supervised system trained on a large enough corpora (say a thousand examples per word) should provide accurate disambiguation on any corpora (or, at least significantly better than MFS). Consequently, it is our belief that a number of issues regarding portability, tuning, knowledge acquisition, etc., should be thoroughly studied before stating that the supervised ML paradigm is able to resolve a realistic WSD problem. Regarding the ML algorithms tested, the contribution of this work consist of empirically demonstrating that the LazyBoosting algorithm outperforms other three supervised ML methods for WSD. Furthermore, this algorithm is proven to have better properties when is applied to new domains.

Chapter 5

Bootstrapping

As we have seen in previous chapters, supervised systems are one of the most promising approaches to Word Sense Disambiguation. The most important drawback with them is the need of a large quantity of labelled examples to train classifiers. One of the most important open lines of research in the field is the use of bootstrapping techniques to take advantage of the unlabelled examples in learning classifiers. It is usually easy to obtain unlabelled examples from large resources, such as the Web or large text collections. The main aim of this chapter is to explore the usefulness of unlabelled examples in Supervised Word Sense Disambiguation systems. We explore in this chapter two approaches of those described in section 2.3.1: the *transductive* learning for SVMs (Joachims, 1999) and the bootstrapping algorithm proposed by Abney (2002, 2004).

5.1 Transductive SVMs

The Transductive approach (Vapnik, 1998) consists of training a system with labelled and unlabelled examples at a time. Specifically, the transductive approach for Support Vector Machines begins by learning a usual SVM from the labelled examples and ends by adjusting the separating hyperplane by means of an iterative process, guessing the class of the unlabelled examples. The proportion of examples for each class is being maintained throughout all iterations.

Joachims (1999) suggested the utility of the *transductive* approach for Text Categorisation. The experiments show the viability of training with a small set of labelled examples and a large set of "cheap" unlabelled examples.

Word Sense Disambiguation and Text Categorisation share some important characteristics: high dimensional input space, sparse example vectors and few irrelevant

features. The transductive approach seems to work well in Text Categorisation because documents usually share words (treated as features) of the domain. That is, each domain or category has some typical words associated and the documents of the same category usually contain a subset of them. Once identifies with one of the categories, the unlabelled examples are used to generalise the classification model with other complementary features not included in the labelled set. See details at (Joachims, 1998, 1999). This section reports the experiment of Joachims (1999) ported to the Word Sense Disambiguation problem, using as data the DSO corpus.

5.1.1 Setting

The experiment have been carried out using the SVM^{light} implementation of the Inductive (SVM) and Transductive ($TSVM$) Support Vector Machines. This software, developed by Thorsten Joachims (1999), is publicly available and open source.

For this work, eight words of the DSO corpus (Ng and Lee, 1996) have been selected. Seven of them have the higher number of examples in the DSO corpus. The other, *art*, has been selected because of the availability of a large unlabelled corpus for this word. They are 4 nouns (*art*, *law*, *state*, and *system*) and 4 verbs (*become*, *leave*, *look*, and *think*). Table 5.1 shows, as a reference, the number of examples, and the 10 fold cross-validation accuracy of the naive *Most Frequent Classifier* and the implementation of AdaBoost.MH described in section 3.4. The English version of FreeLing part-of-speech tagger have been used for tagging (Atserias et al., 2006). The accuracy improvement over the MFC baseline is 21.7 points, comparable to those obtained in previous chapters.

Two partitions of examples have been generated. One of 10 folds (hereinafter PA) used for *parameter tuning* and one of 20 folds (hereinafter PB) for the *main experiment*. The examples have been randomly selected maintaining the proportion of the examples per sense for each fold.

Feature Set

Let "$\ldots\ w_{-3}\ w_{-2}\ w_{-1}\ w\ w_{+1}\ w_{+2}\ w_{+3}\ \ldots$" be the context of consecutive words around the word w to be disambiguated, and $p_{\pm i}$ ($-3 \leq i \leq 3$) be the part-of-speech tag of word $w_{\pm i}$. The feature patterns referring to local context are the following: w, p_{-3}, p_{-2}, p_{-1}, p_{+1}, p_{+2}, p_{+3}, w_{+3}, w_{-2}, w_{-1}, w_{+1}, w_{+2} and w_{+3}.

The features referring to *topical* context consist of: the unordered set of open-class words appearing in the sentence, all bigrams of open-class words (after deleting the

word.pos	#exs	mfc	boost
art.n	388	46.7	61.0
law.n	1,489	29.6	60.7
state.n	1,438	33.7	46.5
system.n	1,491	26.0	38.5
become.v	1,455	40.5	68.2
leave.v	1,471	29.7	50.1
look.v	1,450	51.0	77.3
think.v	1,442	46.1	69.5
nouns	4,806	31.1	49.6
verbs	5,818	41.8	66.2
all	10,624	37.0	58.7

Table 5.1: Number of examples and accuracy results of the two baseline systems (in %)

closed-class words of the sentence), and, finally, all the word bigrams appearing in the sentence (including open- and closed-class words).

This feature set is a small modification of *SetB* in section 3.2.2. Local trigrams have been deleted and the topical bigrams added. This decision has been motivated due to experimental results of Senseval exercises (see chapter 6).

Parameter Tuning

As in the previous chapters, one of the first things to decide when using the *Support Vector Machines* is the *kernel*. Simple linear SVMs have been used, since the use of more complex kernels (polynomials, gaussian functions, etc.) did not lead to better results in some exploratory experiments.

There is another parameter to estimate, the C value. This parameter measures the trade-off between training error and margin maximisation. Table 5.2 contains the C value for the SVMs and TSVMs estimated in a 10–fold cross-validation experiment (over the PA partition), by selecting the values maximising accuracy. The values ranged from 0 and 2.5 in steps of 0.05. In previous experiments we have seen that good values for the parameter applied to WSD data are those near to 0. In the SVM case, 9 folds are considered as training set and 1 fold as testing set each time, while in the $TSVM$ case, 1 fold is taken as the training set and the remaining 9 as test set, in order to better resemble the real usage scenario.

In performing the parameter tuning, there is another possible improvement to test.

	C value	
word	**SVM**	**TSVM**
art.n	0.1	1.35
law.n	0	0.05
state.n	0	2.05
system.n	0.05	1.25
become.v	0.1	0.05
leave.v	0.05	0.05
look.v	0.05	0.05
think.v	0	1.1

Table 5.2: Optimum values of parameter C

SVM^{light} is a binary implementation of SVM. That is, it can not directly deal with a multiclass classification problem (see section 6.4 for some binarisation examples). In order to provide a classifier for that kind of problems, a binarisation process have to be done. Thus, the parameter tuning can be done at the level of each of binary problem. However, first results in this direction showed no significant improvements.

5.1.2 Experiments with TSVM

There are two main experiments in Joachims (1999): the first one aiming to show the effect of the training set size (fixing the size of the test set) and the other one to see the effect of the test set size (fixing the training set to a small size).

Both experiments reported a better accuracy for the transductive approach than the inductive one, but not for all the classes. The first experiment showed largest differences when using very small training sets, tending to disappear when increasing the number of training examples. The second experiment showed a logarithmic-like accuracy curve, always over the accuracy result of the *inductive* approach.

Main Experiment

This section is devoted to translate experiments in (Joachims, 1999) to the Word Sense Disambiguation domain. It consists of carrying out two experiments for each problem. In this case, the *PB* partition of the corpus has been used. This experiment (hereinafter *VarTrain*) tries to show the effect of the number of examples in the training set, fixing the test set; and, the second (hereinafter *VarTest*) the effect of the number of examples in test set, fixing the training set.

Figure 5.1 shows the results of these experiments for the 4 nouns considered, and figure 5.2 for the 4 verbs. Left plots of both figures show the results of the *VarTrain* experiment, and right plots show the *VarTest* results. For better understanding, table 5.3 shows the folds used as training and test sets in both experiments. *VarTrain* outputs a learning curve in which the test set are always folds from 10 to 19 and the training set is variable. In the first iteration the training set is fold 0, in the second iteration, the training are folds 0 and 1 and so on until using for training folds from 0 to 9. *VarTest* has the same behaviour but fixing training set to fold 0, and varying test set from the use of only fold 1, to the use of folds form 1 to 19.

	VarTrain	**VarTest**
Training Set	Folds 0, 0-1, ..., 0-9	Fold 0
Testing Set	Folds 10-19	Folds 1, 1-2, ..., 1-19

Table 5.3: Number of folds used as training and test sets in each experiment

The learning curves are very disappointing. All *VarTrain* plots show worse performance for TSVM than for SVM. Only in the case of the verb *become* the accuracy is comparable. *VarTest* plots show a very irregular behaviour, that in many cases tends to decrease the accuracy when using TSVM. There is only a case (verb *become*) in which the results are similar to those observed in the Text Categorisation problem by Joachims.

Second Experiment

A second experiment has been conducted in order to test the integrity of the transductive models. It consists of running SVM and TSVM under the same conditions and with a small number of examples in the test set (9 folds for training a 1 fold for test), for minimising the effect of the *transductive* examples.

The results of this experiment are shown in table 5.4. It can be seen that when using the *transductive* approach the accuracy of the system decreases in all words. The parameters have been tuned over the *inductive* approach, but this is not a plausible reason for the comparative low results of TSVM. In the experiments of Joachims (1999) the accuracy of TSVM is always higher than that of SVM. This fact is an evidence of the differences between Text Categorisation and Word Sense Disambiguation problems.

Another evidence is the nature of examples in both problems. In Text Categorisation the examples are documents and their representation usually consists of the words of the document, and the problem is to assign one or more labels to the document. In Word Sense Disambiguation the examples are sentences where a word has to be tagged with a sense

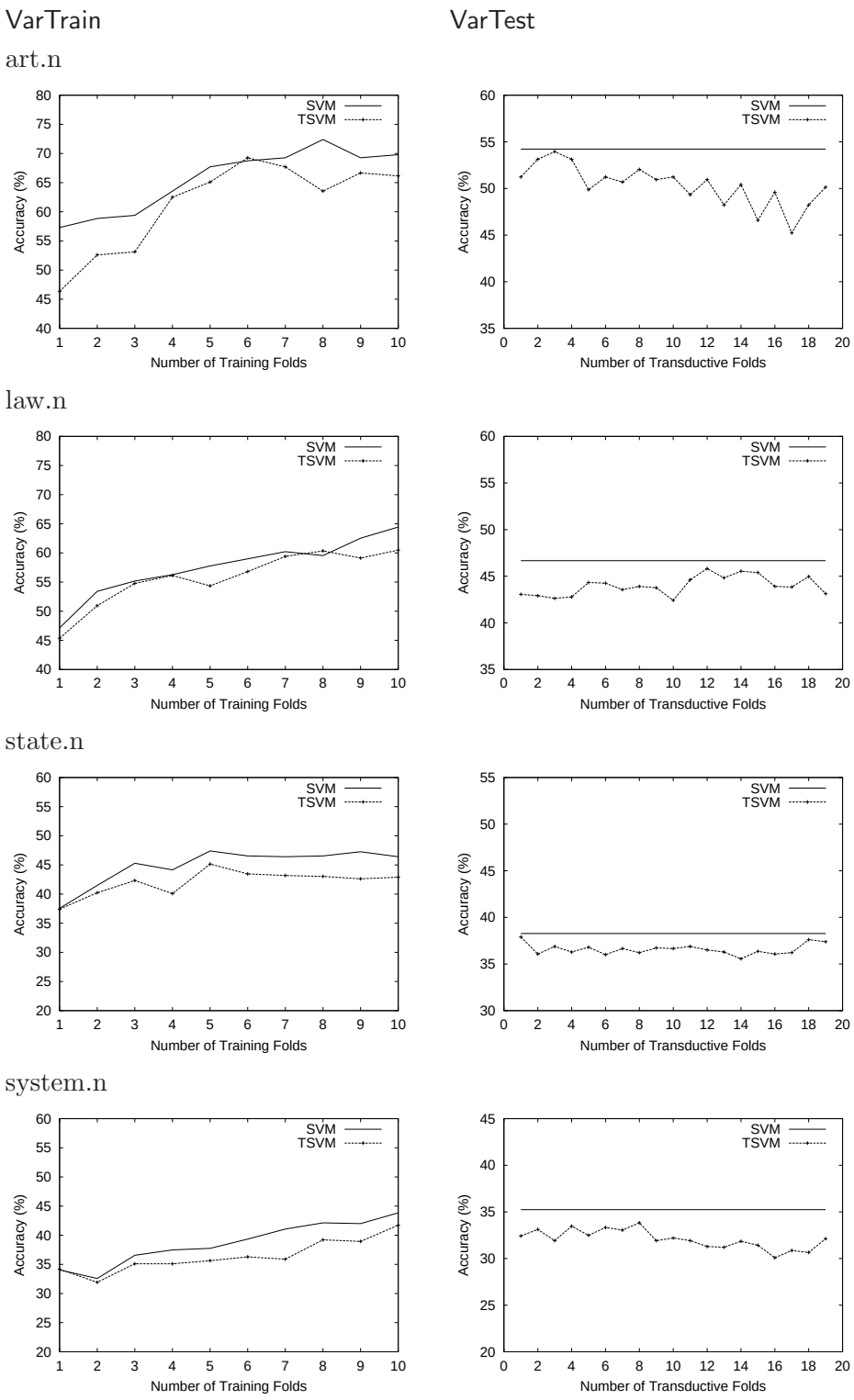

Figure 5.1: Main TSVM experiment on nouns

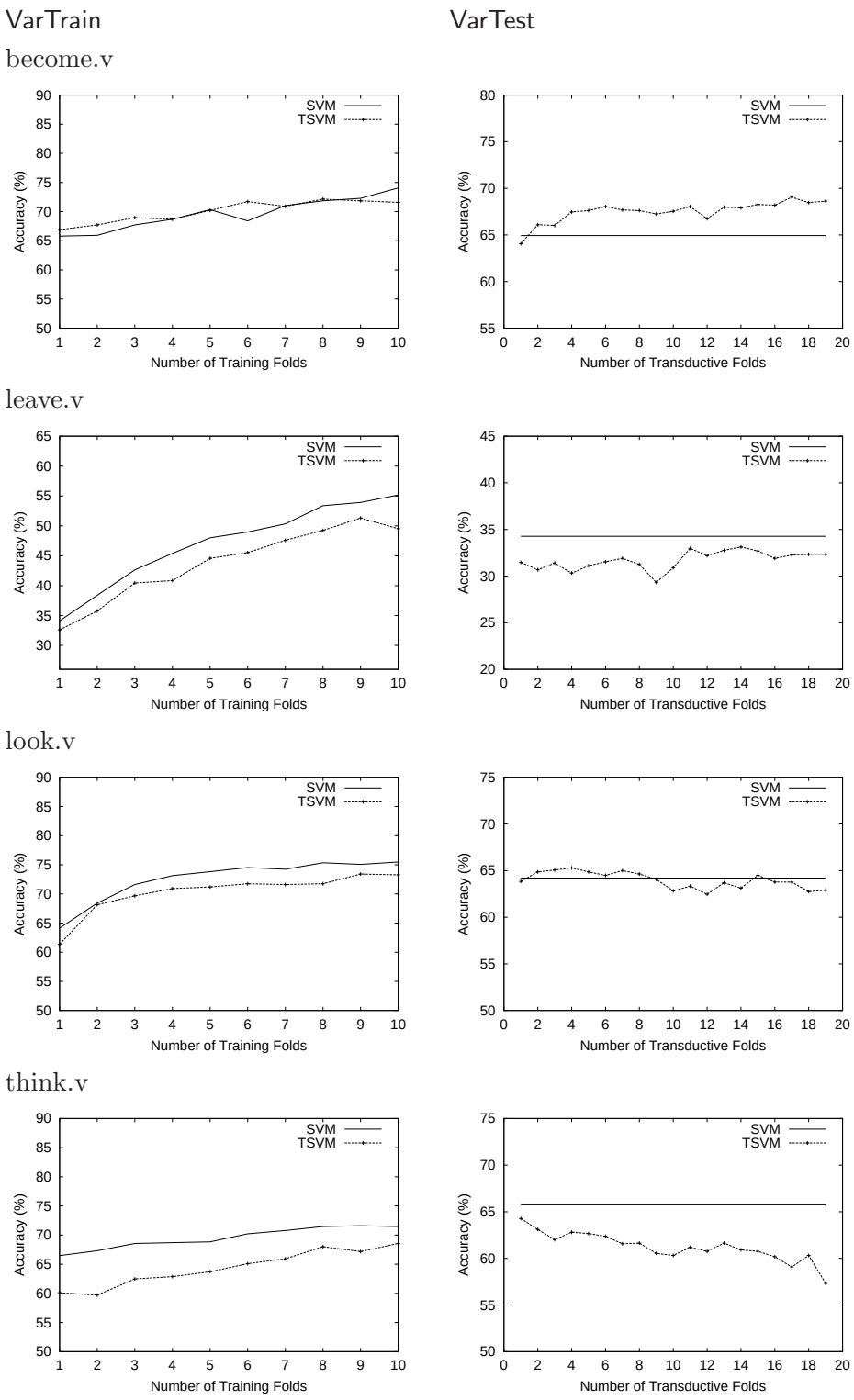

Figure 5.2: Main TSVM experiment on verbs

word.pos	MFC	SVMs	TSVMs
art.n	46.7	72.68	68.56
law.n	29.6	65.75	64.27
state.n	33.7	52.09	49.51
system.n	26.0	48.56	44.33
become.v	40.5	74.91	70.86
leave.v	29.7	57.92	53.57
look.v	51.0	79.52	77.38
think.v	46.1	72.47	69.97
nouns	31,1	56.89	54.01
verbs	41.8	71.16	67.89
all	37.0	64.70	61.61

Table 5.4: Accuracy figures of SVM and TSVM compared under the same conditions (in %)

label. In the representation of examples in Word Sense Disambiguation there are different kinds of features with a small number of values in a sentence. In Text Categorisation there is only one kind of feature (the words in the document) with lots of values; that is, there are many more words in Text Categorisation than in Word Sense Disambiguation, and the probability of word intersection between documents is far higher in Text Categorisation.

Third Experiment

This experiment is devoted to show the time needed by both *transductive* and *inductive* approaches. Table 5.5 shows the time cost in processing, under the same conditions, the learning and test process for the word *art* (the one with the lowest number of examples). Processes were run on a Pentium III with 128Mb of RAM under Linux.

method	time
SVMs $(9 \rightarrow 1)$	0m14.91s
TSVMs $(9 \rightarrow 1)$	1m14.14s
TSVMs $(1 \rightarrow 9)$	17m58.76s

Table 5.5: Processing time of SVM and TSVM for the word *art*

The inductive SVM algorithm performed a single learning (using 9 folds) and testing (using 1 fold) experiment in less than 15 seconds, while TSVM performed the same experiment in 1 minute and 15 seconds (5 times more). When learning TSVM on 1 fold

and testing over 9 folds the time requirements grow up to 18 minutes. As expected, the TSVM is very time consuming when a large amount of unlabelled examples are included in the training process.

The high cost of TSVM is due to the need of retraining at every classification of an unlabelled example. This cost could be alleviated by an *incremental* (on-line) implementation of the SVM paradigm, which is by now not available in the SVM^{light} software.

5.2 Greedy Agreement Bootstrapping Algorithm

This section describes an experiment carried out to explore the bootstrapping algorithm, described in (Abney, 2002, 2004), applied to Word Sense Disambiguation (see section 2.3.1 for a discussion of the different bootstrapping algorithms). It consists of learning two views (independent classifiers) called F and G, respectively, and it is assumed that both are *conditionally independent* given a target label. In practice, both views F and G are classifiers that learn with the same ML algorithms but with different feature set. The bootstrapping algorithm is an iterative process (figure 5.3 shows its pseudo-code) that learns an atomic rule for one of the views at each iteration alternatively. The algorithm needs as input a set of seeds, that is, an initial set of atomic rules with a very high precision, but low coverage. The result of the learning process consists of two classifiers, one for each view, which may be combined to classify new instances.

```
procedure Greedy Agreement
Input:  seed rules F, G
  loop
    foreach atomic rule H
      G' := G + H
      evaluate cost of (F, G')
      keep lowest-cost G'
    if G' is worse than G, quit
    swap F, G'
return F and G
end Greedy Agreement
```

Figure 5.3: The Greedy Agreement algorithm

Classifiers F and G induced from both views can abstain when predicting: $F(x) = \perp$

or $G(x) = \perp$. The cost function needed by the algorithm is defined in (Abney, 2002) as:

$$cost(F, G) = \frac{cost(F) + cost(G)}{2}$$

where

$$cost(F) = \frac{\delta}{\mu - \delta} \times lab(F) + unlab(F)$$

where $\delta = Pr[F \neq G|F, G \neq \perp]$, $\mu = min_u Pr[F = u|F \neq \perp]$, $lab(F)$ is the number of predictions of classifier F, and $unlab(F)$ is the number of times classifier F abstains. $cost(G)$ is computed in the same way.

Each atomic rule corresponds to test the value of a unique feature and emit a classification decision. This algorithm does not delete atomic rules. That is, an atomic rule could be selected more than one time. This fact allows to assign some kind of "weight" to the atomic rules (or features).

Finally, the combination of both views, in order to get a final classification, is not a trivial issue due to the fact that both views could abstain or predict a label. We have implemented several alternatives that are tested in the subsequent experiments. All variants we tried are summarised in table 5.6 together with a brief explanation of each strategy.

5.2.1 Setting

Four words of the DSO corpus have been selected. These words have the highest number of examples for their two most frequent senses. They are two nouns (*law* and *system*) and two verbs (*become* and *think*). Table 5.7 shows the two most frequent senses and the number of examples for each of these senses. The experiment has been performed taking into account only these two most frequent senses for each word. Abney (2002) applied its algorithm to Name Entity Recognition and for that task he had available a very large data set. In WSD there are not such a large annotated corpora. This is why we have selected only the two most frequent senses from those words with more examples.

Table 5.8 shows the set of features selected for each view and a short description of each feature. Considering the assumption of conditionally independence of the views, we have chosen three local features for view F and three topical features for view G.

A random partition of examples in 10 folds has been generated. These folds have been numbered from 0 to 9. Tables 5.9 and 5.10 show the results (precision, recall, F1 and coverage) calculated to serve as baselines for nouns and verbs respectively. "$0 \rightarrow (1 - 8)$" stands for fold 0 as training data and folds from 1 to 8 as test data. "$0 \rightarrow 9$" stands for

key	definition
STRICT_AGREE	the final classifier only predicts a label (as output) if both classifiers (F and G) have a prediction and the label is the same
RELAX_AGREE	the final classifier predicts a label e if both classifiers F and G predict the same label e or if one predicts e and the other abstains
PRIORITY_X	the final classifier predicts a label e if one or both predict e; but if both labels are different, it predicts that of view X; where $X \in \{F, G\}$
STRICT_VOTING	the final classifier only predicts a label when both classifiers have a prediction; then it gets the highest scoring label; else abstains
RELAX_VOTING	STRICT_VOTING but if one of the views abstains, the final classifier takes the label from the non abstaining view
ONLY_VOTING	the predicted label is the highest scoring label from both classifiers and all classes

Table 5.6: Description of the ways of combining both views

	senses		**examples**	
law.n	14:00	10:00	444	361
system.n	06:00	09:02	417	141
become.v	30:00	42:00	606	532
think.v	31:01	31:00	690	429

Table 5.7: Senses and number of examples for each word selected from DSO corpus

view	**feature**	**description**
F	bigsf	all word form bigrams in the sentence
F	fcomb	word forms of all pairs (consecutive or not) of the open-class-words in the sentence
F	top_f	bag of word forms of the sentence
G	f_loc	all word forms and its position in a ±3-word-window
G	form	word form of the target word
G	p_loc	part-of-speech of the relative words in a ±3-word-window together with their position to the target word

Table 5.8: Description of the features for each view

fold 0 as training data and fold 9 as test data. The distribution of the folds has been done to be comparable with the experiments in next subsection. In this experiment, fold 0 is used to generate the set of seeds; folds from 1 to 8 are used as unlabelled training set; and fold 9 as test set. *MFC*, *DLs*, and *LB* stands respectively for Most Frequent Classifier, Decision Lists, and LazyBoosting (see chapter 3 for a description of these algorithms). In previous chapters DLs had a coverage of 100% and now it is lower. This is due to the fact that we were using the Most Frequent Sense when *DL* abstained or ties were observed. Instead, in this experiment ties and abstentions are not solved. Here, we will be using DLs to generate the set of seeds. In all three cases the training set contains only the examples of fold 0.

	0 → (1 − 8)							
	law.n				**system.n**			
	prec	**rec**	**F1**	**cov**	**prec**	**rec**	**F1**	**cov**
MFC	55.16	55.16	55.16	100.00	74.73	74.73	74.73	100.00
DLs	59.87	42.40	49.64	70.81	60.59	46.19	52.42	76.23
LB	59.75	59.75	59.75	100.00	62.33	62.33	62.33	100.00
	0 → 9							
	law.n				**system.n**			
	prec	**rec**	**F1**	**cov**	**prec**	**rec**	**F1**	**cov**
MFC	55.16	55.16	55.16	100.00	74.73	74.73	74.73	100.00
DLs	52.73	38.16	44.28	72.37	71.43	51.28	59.70	71.79
LB	64.47	64.47	64.47	100.00	71.79	71.79	71.79	100.00

Table 5.9: Baseline accuracies for nouns (precision - recall - F1 - coverage) in %

The main conclusions that arise from this results are the following: The results are lower for nouns than for verbs. The classifiers learn the behaviour of verbs with only fold 0 as a training set. Finally, note that the accuracies obtained by DL for noun *law* and by both methods for noun *system* are under the most frequent .

5.2.2 Experimental Evaluation

We started a set of controlled experiments using the bootstrapping algorithm by measuring the effect of the threshold for selecting the initial set of seed rules. We have trained a Decision List on fold 0 to obtain the most confident rules as seed rules for the Abney's algorithm. Tables 5.11, 5.12 and 5.13 show the results (precision, recall, F1 and coverage) corresponding to three different sets of seeds resulting from thresholds 3.0, 2.5 and 2.0,

	0 → (1 − 8)							
	become.v				**think.v**			
	prec	**rec**	**F1**	**cov**	**prec**	**rec**	**F1**	**cov**
MFC	53.25	53.25	53.25	100.00	61.66	61.66	61.66	100.00
DLs	85.17	80.81	82.93	94.87	89.54	85.14	87.28	95.08
LB	87.24	87.24	87.24	100.00	87.15	87.15	87.15	100.00
	0 → 9							
	become.v				**think.v**			
	prec	**rec**	**F1**	**cov**	**prec**	**rec**	**F1**	**cov**
MFC	53.25	53.25	53.25	100.00	61.66	61.66	61.66	100.00
DLs	83.50	78.90	81.13	94.50	86.36	84.07	85.20	97.35
LB	85.32	85.32	85.32	100.00	86.73	86.73	86.73	100.00

Table 5.10: Baseline accuracies for verbs (precision - recall - F1 - coverage) in %

respectively. These values have been selected to evaluate the trade-off between accuracy and coverage during the bootstrapping process.

The first line of each table (Iterations) shows the total number of iterations applied. The bootstrapping algorithm stops when the addition of the best rule selected for the next iteration is not able to improve the cost function. For instance, the number of iterations for the noun *system* in table 5.13 (threshold 2.0) is 0 because for this threshold no further rules where included into the bootstrapping process. For comparison purposes, in all tables, section *seed* stands for the results of applying directly the seed rules learned from fold 0 on the test folds 1-8 and fold 9; and the *Abney* section for the results of applying the bootstrapping algorithm using folds 1-8 as bootstrapping examples on the same test folds. Thus, $0 \rightarrow (1-8) \rightarrow (1-8)$ stands for extracting the seed rules from fold 0, performing the bootstrapping process on folds (1-8) and testing also on folds (1-8); while $0 \rightarrow (1-8) \rightarrow 9$ provides the tests on fold 9.

As expected, we obtain very different results depending on the threshold selected. It seems that the Abney's algorithm heavily depends on the initial set of seeds as shown by the final number of iterations and the accuracy figures on both (1-8) and 9 test sets. That is, the optimal threshold appears to be different for each word.

We also obtain very different performances depending on the way of combining both views. The best results are obtained by *priority_F*, *priority_G* and *only_voting*. Also as expected, Abney's algorithm obtains generally better figures than the initial seed algorithm. This is not the case for *think.v* using threshold 3.0 and using as test folds 1-8, for both verbs using threshold 2.5 in both test sets and with the verb *become* using

	law.n	system.n	become.v	think.v
Iterations	71	63	93	79
Seed (0 → (1 − 8)) (pre. rec. F1 cov.)				
F	66.2 14.4 23.7 21.8	65.0 22.0 32.9 33.8	80.4 30.0 43.7 37.3	76.8 24.0 36.6 31.3
G	74.2 11.1 19.3 14.9	73.1 8.6 15.3 11.7	89.7 68.9 77.9 76.9	93.8 75.9 83.9 81.0
strict_agree	78.1 4.9 9.3 6.3	83.9 3.9 7.4 4.6	96.3 25.4 40.2 26.4	99.5 21.3 35.1 21.5
relax_agree	68.4 20.0 30.9 29.2	65.8 25.9 37.1 39.3	88.4 68.2 77.0 77.1	91.8 73.0 81.3 79.4
priority_F	67.5 20.1 31.0 29.8	65.7 26.3 37.6 40.1	83.7 69.0 75.7 82.4	86.4 73.5 79.4 85.1
priority_G	68.6 20.4 31.5 29.8	65.3 26.2 37.4 40.1	88.1 72.6 79.6 82.4	91.7 78.1 84.4 85.1
strict_voting	78.1 4.9 9.3 6.3	84.4 4.0 7.7 4.8	94.0 27.5 42.5 29.2	95.8 22.9 37.0 23.9
relax_voting	68.4 20.0 30.9 29.2	65.9 26.0 37.3 39.5	87.9 70.2 78.1 79.9	91.0 74.5 82.0 81.9
only_voting	68.4 20.0 30.9 29.2	65.8 25.9 37.1 39.3	88.0 70.5 78.3 80.0	91.0 74.5 82.0 81.9
Abney (0 → (1 − 8) → (1 − 8)) (pre. rec. F1 cov.)				
F	60.3 58.2 59.2 96.6	51.7 51.7 51.7 100	86.8 72.5 79.0 83.5	83.1 83.0 83.1 99.9
G	61.0 58.7 59.8 96.2	50.7 42.5 46.2 83.7	85.7 84.0 84.8 98.0	83.7 82.1 82.9 98.1
strict_agree	60.9 56.5 58.7 92.8	50.7 42.5 46.2 83.7	88.0 71.7 79.0 81.5	86.3 75.8 80.7 87.8
relax_agree	60.4 60.4 60.4 100	51.7 51.7 51.7 100	84.9 84.6 84.8 99.7	83.1 83.0 83.1 99.9
priority_F	60.4 60.4 60.4 100	51.7 51.7 51.7 100	84.7 84.6 84.7 99.9	83.1 83.0 83.1 99.9
priority_G	60.4 60.4 60.4 100	51.7 51.7 51.7 100	84.9 84.8 84.9 99.9	83.1 83.0 83.1 99.9
strict_voting	60.9 56.5 58.7 92.8	50.7 42.5 46.2 83.7	87.7 71.7 78.9 81.7	86.3 75.8 80.7 87.8
relax_voting	60.4 60.4 60.4 100	51.7 51.7 51.7 100	84.7 84.6 84.7 99.9	83.1 83.0 83.1 99.9
only_voting	60.4 60.4 60.4 100	51.7 51.7 51.7 100	84.8 84.6 84.7 99.8	83.1 83.0 83.1 99.9
Seed (0 → 9) (pre. rec. F1 cov.)				
F	58.8 13.2 21.5 22.4	80.0 30.8 44.5 38.5	91.3 38.5 54.2 42.2	68.6 21.2 32.4 31.0
G	76.9 13.2 22.5 17.1	83.3 12.8 22.2 15.4	83.3 64.2 72.5 77.1	93.4 75.2 83.3 80.5
strict_agree	83.3 6.6 12.2 7.9	100 7.7 14.3 7.7	100 29.4 45.4 29.4	91.7 19.5 32.1 21.2
relax_agree	63.6 18.4 28.6 29.0	79.4 34.6 48.2 43.6	86.1 67.9 75.9 78.9	87.5 74.3 80.4 85.0
priority_F	60.9 18.4 28.3 30.3	80.0 35.9 49.6 44.9	82.6 69.7 75.6 84.4	84.9 74.3 79.3 87.6
priority_G	65.2 19.7 30.3 30.3	77.1 34.6 47.8 44.9	84.8 71.6 77.6 84.4	87.9 77.0 82.1 87.6
strict_voting	83.3 6.6 12.2 7.9	100 7.7 14.3 7.7	97.1 31.2 47.2 32.1	92.0 20.4 33.3 22.1
relax_voting	63.6 18.4 28.6 29.0	79.4 34.6 48.2 43.6	85.4 69.7 76.8 81.7	87.6 75.2 81.0 85.8
only_voting	63.6 18.4 28.6 29.0	79.4 34.6 48.2 43.6	85.4 69.7 76.8 81.7	87.6 75.2 81.0 85.8
Abney (0 → (1 − 8) → 9) (pre. rec. F1 cov.)				
F	69.8 57.9 63.3 82.9	54.7 52.6 53.6 96.2	88.3 62.4 73.1 70.6	84.7 73.5 78.7 86.7
G	68.7 60.5 64.4 88.2	51.8 37.2 43.3 71.8	85.7 71.6 78.0 83.5	91.1 81.4 86.0 89.4
strict_agree	72.7 52.6 61.1 72.4	52.7 37.2 43.6 70.5	91.4 48.6 63.5 53.2	93.8 67.3 78.4 71.7
relax_agree	69.8 57.9 63.3 82.9	54.0 52.6 53.3 97.4	90.4 78.0 83.7 86.2	89.2 80.5 84.7 90.3
priority_F	68.1 61.8 64.8 90.8	54.0 52.6 53.3 97.4	88.2 82.6 85.3 93.6	83.6 81.4 82.5 97.4
priority_G	68.1 61.8 64.8 90.8	54.0 52.6 53.3 97.4	86.3 80.7 83.4 93.6	89.1 86.7 87.9 97.4
strict_voting	70.7 54.0 61.2 76.3	52.7 37.2 43.6 70.5	86.2 51.4 64.4 59.6	94.1 70.8 80.8 75.2
relax_voting	68.2 59.2 63.4 86.8	54.0 52.6 53.3 97.4	87.1 80.7 83.8 92.7	89.6 84.1 86.8 93.8
only_voting	68.2 59.2 63.4 86.8	54.0 52.6 53.3 97.4	88.9 80.7 84.6 90.8	87.9 83.2 85.2 94.7

Table 5.11: Results of Abney's algorithm with a seed threshold of 3 (in %)

	law.n	**system.n**	**become.v**	**think.v**
Iterations	115	114	120	94
Seed (0 → (1 − 8)) (pre. rec. F1 cov.)				
F	59.3 32.7 42.2 55.2	65.1 38.1 48.1 58.6	72.6 45.4 55.8 62.5	75.2 53.1 62.2 70.6
G	61.2 20.6 30.8 33.6	55.8 18.8 28.2 33.8	89.6 78.3 83.6 87.4	92.2 76.4 83.6 82.9
strict_agree	70.4 8.8 15.6 12.4	67.4 9.0 15.8 13.3	94.5 39.3 55.5 41.6	95.4 45.8 61.9 48.1
relax_agree	59.9 38.6 46.9 64.4	62.9 41.3 49.8 65.6	88.1 70.0 78.0 79.5	88.5 71.2 78.9 80.5
priority_F	58.3 41.0 48.2 70.4	61.8 44.7 51.9 72.4	76.4 71.8 74.0 93.9	78.3 72.7 75.4 93.0
priority_G	59.8 42.1 49.4 70.4	61.6 44.5 51.7 72.4	88.0 82.7 85.3 93.9	88.3 82.1 85.1 93.0
strict_voting	65.6 9.7 16.9 14.8	63.1 10.5 17.9 16.6	89.3 46.6 61.2 52.1	88.5 49.1 63.1 55.4
relax_voting	59.2 39.5 47.4 66.7	62.0 42.8 50.6 68.9	85.8 77.3 81.4 90.1	84.7 74.4 79.2 87.8
only_voting	59.5 39.6 47.6 66.7	62.1 42.3 50.3 68.2	87.0 77.3 81.9 88.9	85.9 74.8 79.9 87.0
Abney (0 → (1 − 8) → (1 − 8)) (pre. rec. F1 cov.)				
F	62.4 62.1 62.3 99.4	62.1 62.0 62.1 99.9	83.3 74.4 78.6 89.3	81.0 80.9 80.9 99.9
G	62.2 59.0 60.6 94.8	63.0 47.5 54.2 75.5	84.4 81.6 83.0 96.6	81.7 78.2 79.9 95.8
strict_agree	62.5 58.5 60.4 93.7	63.2 47.5 54.3 75.2	85.5 72.4 78.4 84.7	81.8 78.1 79.9 95.5
relax_agree	62.4 62.1 62.2 99.5	62.0 61.7 61.9 99.6	83.4 82.2 82.8 98.6	81.0 80.9 80.9 99.9
priority_F	62.5 62.5 62.5 100	62.1 62.0 62.1 99.9	82.3 82.2 82.3 99.9	81.0 81.0 81.0 100
priority_G	62.1 62.1 62.1 100	61.8 61.7 61.8 99.9	83.6 83.5 83.6 99.9	80.9 80.9 80.9 100
strict_voting	62.6 58.8 60.7 94.0	63.0 47.5 54.2 75.5	85.1 73.0 78.6 85.7	81.7 78.1 79.8 95.6
relax_voting	62.5 62.4 62.4 99.9	61.8 61.7 61.8 99.9	83.1 82.8 83.0 99.6	80.9 80.9 80.9 100
only_voting	62.2 62.1 62.1 99.9	61.8 61.7 61.8 99.9	83.4 82.9 83.2 99.4	80.9 80.9 80.9 100
Seed (0 → 9) (pre. rec. F1 cov.)				
F	42.2 25.0 31.4 59.2	70.2 51.3 59.3 73.1	82.1 50.5 62.5 61.5	77.0 59.3 67.0 77.0
G	57.1 21.1 30.8 36.8	68.0 21.8 33.0 32.1	82.6 69.7 75.6 84.4	94.6 77.9 85.4 82.3
strict_agree	50.0 7.9 13.6 15.8	91.7 14.1 24.4 15.4	93.6 40.4 56.4 43.1	93.6 51.3 66.3 54.9
relax_agree	46.7 27.6 34.7 59.2	70.4 48.7 57.6 69.2	85.2 68.8 76.1 80.7	88.8 77.0 82.5 86.7
priority_F	47.2 32.9 38.8 69.7	71.0 56.4 62.9 79.5	79.0 72.5 75.6 91.7	80.6 77.0 78.7 95.6
priority_G	47.2 32.9 38.8 69.7	64.5 51.3 57.1 79.5	83.0 76.2 79.4 91.7	89.8 85.8 87.8 95.6
strict_voting	50.0 10.5 17.4 21.1	93.8 19.2 31.9 20.5	85.5 43.1 57.3 50.5	88.4 54.0 67.0 61.1
relax_voting	46.9 30.3 36.8 64.5	72.4 53.9 61.8 74.4	81.3 71.6 76.1 88.1	85.7 79.7 82.6 92.9
only_voting	45.7 27.6 34.4 60.5	71.9 52.6 60.7 73.1	81.3 71.6 76.1 88.1	86.7 80.5 83.5 92.9
Abney (0 → (1 − 8) → 9) (pre. rec. F1 cov.)				
F	55.9 43.4 48.9 77.6	72.1 62.8 67.1 87.2	75.0 55.1 63.5 73.4	81.3 77.0 79.1 94.7
G	63.2 56.6 59.7 89.5	52.8 35.9 42.8 68.0	79.3 63.3 70.4 79.8	82.0 72.6 77.0 88.5
strict_agree	63.4 34.2 44.4 54.0	66.7 28.2 39.7 42.3	86.3 40.4 55.0 46.8	85.0 69.9 76.7 82.3
relax_agree	61.3 50.0 55.1 81.6	68.3 52.6 59.4 76.9	81.4 64.2 71.8 78.9	80.6 77.0 78.7 95.6
priority_F	56.8 55.3 56.0 97.4	68.9 65.4 67.1 94.9	75.3 69.7 72.4 92.7	80.2 78.8 79.5 98.2
priority_G	62.2 60.5 61.3 97.4	60.8 57.7 59.2 94.9	78.2 72.5 75.2 92.7	79.3 77.9 78.6 98.2
strict_voting	61.2 39.5 48.0 64.5	61.4 34.6 44.3 56.4	71.9 42.2 53.2 58.7	84.2 70.8 76.9 84.1
relax_voting	60.0 55.3 57.5 92.1	64.8 59.0 61.7 91.0	72.7 66.1 69.2 90.8	80.0 77.9 78.9 97.4
only_voting	60.9 55.3 57.9 90.8	66.2 57.7 61.6 87.2	81.3 67.9 74.0 83.5	80.0 77.9 78.9 97.4

Table 5.12: Results of Abney's algorithm with a seed threshold of 2.5 (in %)

	law.n	**system.n**	**become.v**	**think.v**
Iterations	93	0	137	101
Seed (0 → (1 − 8)) (pre. rec. F1 cov.)				
F	56.1 48.5 52.1 86.5	63.5 54.9 58.9 86.4	69.0 58.9 63.5 85.4	71.7 63.9 67.6 89.2
G	59.4 42.6 49.6 71.6	55.0 37.5 44.6 68.2	88.0 80.0 83.8 91.0	87.7 76.8 81.9 87.5
strict_agree	65.5 25.7 36.9 39.2	68.6 25.1 36.8 36.6	91.1 50.4 64.9 55.3	90.4 55.5 68.8 61.5
relax_agree	58.2 42.6 49.2 73.1	61.6 44.0 51.3 71.3	87.0 65.9 75.0 75.7	84.2 67.8 75.1 80.6
priority_F	55.5 53.3 54.4 96.0	61.3 58.0 59.6 94.6	71.2 70.0 70.6 98.4	71.9 70.4 71.2 97.9
priority_G	57.0 54.7 55.8 96.0	56.2 53.2 54.7 94.6	85.8 84.4 85.1 98.4	84.4 82.6 83.5 97.9
strict_voting	59.8 35.0 44.2 58.5	63.6 35.7 45.7 56.2	76.8 56.7 65.3 73.8	79.5 60.7 68.8 76.3
relax_voting	56.2 51.9 54.0 92.5	60.0 54.6 57.2 90.9	76.6 72.2 74.3 94.2	76.5 73.0 74.7 95.4
only_voting	57.8 51.5 54.4 89.1	60.0 52.8 56.2 87.9	82.4 75.8 79.0 91.9	80.1 74.4 77.2 92.9
Abney (0 → (1 − 8) → (1 − 8)) (pre. rec. F1 cov.)				
F	60.1 59.9 60.0 99.7	-	79.0 77.8 78.4 98.5	85.3 85.1 85.2 99.8
G	61.7 47.8 53.9 77.4	-	83.2 79.1 81.1 95.0	87.8 84.5 86.1 96.2
strict_agree	62.7 45.0 52.4 71.7	-	83.7 73.9 78.5 88.3	88.1 82.2 85.1 93.3
relax_agree	60.6 57.3 58.9 94.6	-	82.0 77.8 79.8 94.9	87.1 84.6 85.8 97.1
priority_F	60.1 60.1 60.1 100	-	78.8 78.8 78.8 100	85.4 85.3 85.3 99.9
priority_G	59.9 59.9 59.9 100	-	81.8 81.8 81.8 100	86.8 86.7 86.8 99.9
strict_voting	61.9 47.6 53.8 77.0	-	81.7 75.7 78.6 92.6	87.1 83.4 85.2 95.8
relax_voting	60.0 59.9 60.0 99.9	-	80.2 79.5 79.9 99.1	86.1 85.7 85.9 99.6
only_voting	60.3 59.6 60.0 98.8	-	81.0 80.2 80.6 98.9	86.6 86.0 86.3 99.3
Seed (0 → 9) (pre. rec. F1 cov.)				
F	55.7 51.3 53.4 92.1	73.5 64.1 68.5 87.2	76.8 67.0 71.6 87.2	75.5 65.5 70.1 86.7
G	71.2 48.7 57.8 68.4	63.0 43.6 51.5 69.2	82.1 71.6 76.5 87.2	91.8 79.7 85.3 86.7
strict_agree	77.8 27.6 40.8 35.5	79.3 29.5 43.0 37.2	88.9 51.4 65.1 57.8	93.9 54.9 69.3 58.4
relax_agree	64.2 44.7 52.7 69.7	76.4 53.9 63.2 70.5	84.6 70.6 76.9 83.5	89.4 74.3 81.2 83.2
priority_F	58.1 56.6 57.3 97.4	74.3 70.5 72.4 94.9	76.2 76.2 76.2 100	77.7 77.0 77.3 99.1
priority_G	62.2 60.5 61.3 97.4	64.9 61.5 63.2 94.9	81.7 81.7 81.7 100	88.4 87.6 88.0 99.1
strict_voting	66.0 40.8 50.4 61.8	72.3 43.6 54.4 60.3	83.1 58.7 68.8 70.6	82.7 59.3 69.1 71.7
relax_voting	60.3 57.9 59.1 96.1	72.6 68.0 70.2 93.6	81.0 78.0 79.4 96.3	81.7 78.8 80.2 96.5
only_voting	57.4 51.3 54.2 89.5	73.1 62.8 57.6 85.9	81.3 79.8 80.6 98.2	81.5 77.9 79.6 95.6
Abney (0 → (1 − 8) → 9) (pre. rec. F1 cov.)				
F	65.7 57.9 61.5 88.2	-	78.4 69.7 73.8 89.0	81.3 77.0 79.1 94.7
G	75.4 56.6 64.7 75.0	-	81.0 74.3 77.5 91.7	89.0 78.8 83.6 88.5
strict_agree	81.8 35.5 49.6 43.4	-	85.1 57.8 68.9 67.9	93.3 62.0 74.5 66.4
relax_agree	78.4 52.6 63.0 67.1	-	84.2 73.4 78.4 87.2	92.6 77.0 84.1 83.2
priority_F	66.2 61.8 64.0 93.4	-	79.8 79.8 79.8 100	82.3 82.3 82.3 100
priority_G	74.7 69.7 72.1 93.4	-	79.8 79.8 79.8 100	88.5 88.5 88.5 100
strict_voting	62.3 43.4 51.2 69.7	-	82.6 65.1 72.8 78.9	81.9 68.1 74.4 83.2
relax_voting	64.8 60.5 62.6 93.4	-	82.2 80.7 81.5 98.2	83.2 83.2 83.2 100
only_voting	66.7 60.5 63.5 90.8	-	84.1 82.6 83.3 98.2	85.6 84.1 84.8 98.2

Table 5.13: Results of Abney's algorithm with a seed threshold of 2 (in %)

threshold 2.0 in both test sets.

In fact, for three words (*law.n*, *become.v* and *think.v*) the greedy agreement algorithm obtains better figures than the three baselines (MFC, Decision Lists and LazyBoosting trained only using fold 0, see also tables 5.9 and 5.10). For instance, for noun *law* using threshold 3.0 and testing on fold 9 and using *priority_F* and *priority_G* achieves an F1 measure of 64.8, while the LazyBoosting baseline obtains 64.5. The same also holds for test on folds 1-8 and five combination rules achieving 60.4 instead of 59.8 F1 measure of the LazyBoosting baseline.

Regarding the best parameters for each word when testing on fold 9 (a new corpora not used during the bootstrapping), the best results for noun *law* are obtained when applying *priority_G* combination rule and threshold 2.0 achieving an F1 measure of 72.1, which represents an improvement of 7.6 points over the best baseline (LB). For the verb *become*, the best results are achieved when applying *priority_F* combination rule and threshold 3.0 obtaining the same figures than the best baseline (an F1 measure of 85.3 for LB). For the verb *think.v*, the best results are obtained when applying *priority_G* combination rule and threshold 2.0 achieving an F1 measure of 88.5, which represents an improvement of 1.8 points over the best baseline (also LB).

However, for the noun *system*, none of the parameter combinations obtain better performance than the best baseline. The best results in this case are obtained when using threshold 3.0 and *F* and *priority_F* rule combinations achieving an F1 measure of 67.1, which is 7.6 points below the best baseline for this word (MFC). In fact, the other two Machine Learning algorithms (DL and LB) are not able to beat the MFC (with an F1 measure of 74.7), which possibly denotes a biased sense distribution and a very difficult task. It could be also the case that the whole set of features (and also possibly the two views) do not characterise appropriately the examples of this word.

It should be also noted that these results are obtained covering almost the whole fold 9 (over 93% coverage).

Now, if we consider folds 1-8 for testing, only for the noun *law* the greedy agreement algorithm obtains better F1 figures than the three baselines.

Regarding the best parameters for each word when testing on folds 1-8 (the same corpora used for the bootstrapping), the best results for noun *law* are obtained when applying *priority_F* combination rule and threshold 2.5 achieving an F1 measure of 62.5, which represents an improvement of 2.7 points over the best baseline (LB obtains an F1 of 59.8). The best results for the noun *system* are obtained when using threshold 2.5 and *F* and *priority_F* combination rules achieving an F1 measure of 62.1, which is 12.6 points below the best baseline for this word (again the MFC), but only 0.2 points below the best

Machine Learning algorithm (LB with an F1 of 62.3). For the verb *become*, the best results are achieved when applying *priority_G* combination rule and threshold 3.0 achieving an F1 measure of 84.9, which is 2.3 points below the best baseline for this word (LB). Finally, for the verb *think*, the best results are achieved when applying *priority_G* combination rule and threshold 2.0 achieving an F1 measure of 86.8, which is 0.4 points below the best baseline for this word (LB).

Summarising, the threshold parameter affects directly the initial coverage and precision. Obviously, the higher is the threshold, the higher the initial precision, and lower the initial coverage, which not surprisingly has a direct relationship with the behaviour of the bootstrapping algorithm. The best results are obtained using *priority_F* or *priority_G* combination rules. When performing the test on fold 9 (not seen by the bootstrapping algorithm), three of the four words, the Abney's algorithm is able to obtain better performances than the baselines. However, when using folds 1-8 for testing (those used by the bootstrapping algorithm) only one word is able to outperform the baselines.

There are at least two points in which these experiments are different from those previously reported for the Named Entity Recognition task (Collins and Singer, 1999). The first one is the amount of unlabelled examples used for the bootstrapping. The experiments on Named Entity Recognition used around 90,000 unlabelled examples (two orders of magnitude higher than those reported here, which consist on about 500 unlabelled examples). The second substantial difference is the high precision rates of the initial seeds. The Collins and Singer's work report rules of almost 100% precision while we are reporting precisions in the range of 60% to 90%.

Thus, our next steps on this research will consist on evaluating the performances of non absolute thresholds. Instead, we plan to use relative thresholds associated to precision, recall, coverage or F1 measures. We also plan to use a larger number of unlabelled examples gathered from the British National Corpus. For instance, in BNC there are 19,873 examples for the noun "law", 40,478 examples for noun "system", 53,844 occurrences of the verb "become" and 86,564 examples for verb "think". After this initial set of experiments, we also plan to use a more realistic evaluation scenario. We plan to use all the senses of the words (7 for law.n, 8 for system.n, 4 for become.v and 8 for think.v) and their whole set of labelled examples from the DSO (around 1,500) as a bootstrapping corpora (750 examples) and as test corpora (750 examples).

5.3 Conclusions and Current/Further Work

The use of unlabelled data to train classifiers for Word Sense Disambiguation is a very challenging line of research in order to develop a really robust, complete and accurate Word Sense Tagger (Agirre et al., 2005). Looking at the results of the experiments of the first section of this chapter, we can conclude that, at this moment, the use of unlabelled examples with the *transductive approach* is not useful for Word Sense Disambiguation. At this point, there is a line of research to explore; to further study on the differences and similarities between Text Categorisation and Word Sense Disambiguation models, in order to develop a Word Sense Disambiguation model "compatible" with the *transductive* approach.

The second section of this chapter explores the application of the Greedy Agreement algorithm of Abney (2002) to Word Sense Disambiguation. It consists of the previous use of Decision Lists in order to obtain the seeds. The atomic rules of Decision Lists are similar to those of Abney's algorithm. We have applied it to only 2 senses of 4 words of the DSO corpus in order to deal with a big amount of examples per sense. Then we have compared the results to a naive Most Frequent Classifier, Decision Lists and Lazy Boosting, and obtained comparable or higher results (in accuracy). This approximation seems to be useful in order to apply bootstrapping to the Word Sense Disambiguation data.

Since the bootstrapping algorithm on this setting is providing interesting but not conclusive results, we plan to continue the study of the behaviour of this approach on a more realistic scenario. It should consist of:

- testing the framework with all word senses (not only with 2 of them),
- increasing the feature sets of the views with those described in appendix B[1],
- optimising the threshold see extraction for each word by applying cross-validation,
- and finally, testing the Greedy Agreement algorithm using large sets of unlabelled data extracted from the British National Corpus; enlarging both the training set and the set used for the extraction of the seed rules.

[1]One of the difficulties when applying the greedy agreement algorithm is the use of a limited number of features in the views. So, the results of the views are far below than when using the complete feature set.

Chapter 6

Evaluation in Senseval Exercises

This chapter is devoted to explain our participation in the international evaluations on Word Sense Disambiguation. Concretely, we have participated in the English Lexical Sample task of both Senseval-2 (Kilgarriff, 2001) and Senseval-3 (Mihalcea et al., 2004) events. We start this chapter with a description of the English Lexical Sample task data for both Senseval-2 and Senseval-3. Section 6.2 explains the system presented at Senseval-2 and section 6.3 shows the results of a comparative study of the best six methods on Senseval-2 English Lexical Sample task. Section 6.4 is devoted to present the architecture of the system presented at Senseval-3 and section 6.3.2 shows further experiments on features with the system.

6.1 Senseval Corpora

This section explains in detail the Senseval English Lexical Sample data[1]. Senseval-2 corpus consists of 8,611 training and 4,328 test examples of 73 words. These examples were extracted from the British National Corpus[2]. Figure 6.1 shows an example from the Senseval-2 corpus. It corresponds to the instance *art.40012*, which has been labelled with two senses: *P* and *arts%1:09:00::*. Most of the examples have only a sense but there are exceptions with two or three senses. The word to be disambiguated is marked by the <*head*> label and is always located in the last sentence of the context. The Senseval-2 sense inventory is a prerelease of WordNet 1.7. Table 6.1 shows the noun *art* in that inventory. For each word the corpus has some extra "sense labels": label *P* for proper-names; and label *U* for unassignable. Unassignable stands for examples in which human annotators

[1] http://www.senseval.org
[2] http://www.natcorp.ox.ac.uk

had problems in labelling. Nouns and adjectives can be also monosemous multiwords –such as *art_class%1:04:00::* or *art_collection%1:14:00::* for the word art–; and verbs can be polysemous phrasal verbs –such as *work_on%2:41:00::* for the verb *work*. Phrasal verbs have been tagged. So, they should be treated as independent words (see figure 6.2 for an example). Senseval providers referred to the sense labels of the examples as *keys* and the labels provided by the participants as *system votes*.

sense	**word**	**pos**	**descriptor**
art%1:04:00::	art	noun	creation
art%1:06:00::	art	noun	artworks
art%1:09:00::	art	noun	skill
art%1:10:00::	art	noun	inabook

Table 6.1: Sense Inventory for the noun *art* in Senseval-2 corpus

Senseval-3 corpus consists of 7,860 training and 3,944 test examples of 57 words. Nouns and adjectives of Senseval-3 data have been annotated using the WordNet 1.7.1 sense inventory, and verbs with labels based on Wordsmyth definitions[3] (see appendix C for Web references to both resources). Senseval-3 sense inventory has neither multiwords, nor phrasal verbs, nor proper-nouns. That is, the senses used are only the senses like those of table 6.1 and the unassignable *U* label. This corpus has a large number of examples with more than one sense assigned. Figure 6.3 shows and example from Senseval-3 corpus.

6.2 TALP System at Senseval-2

This section describes the architecture and results of the TALP system presented at the Senseval-2 exercise for the English Lexical Sample task. The system was developed on the basis of LazyBoosting (Escudero et al., 2000c), the boosting-based approach for Word Sense Disambiguation explained in section 3.4. In order to better fit the Senseval-2 domain, some improvements were made, including: (1) features that take into account domain information, (2) an specific treatment of multiwords, and (3) a hierarchical decomposition of the multiclass classification problem, similar to that of Yarowsky (2000). All these issues will be briefly described in the following sections.

[3] A mapping between Wordsmyth and WordNet verb senses was provided.

```
<instance id="art.40012" docsrc="bnc_A0E_130">
<answer instance="art.40012" senseid="P"/>
<answer instance="art.40012" senseid="arts%1:09:00::"/>
<context>
Titles include Lars von Trier's EUROPA, shown in competition at
Cannes: RECOLLECTIONS OF THE YELLOW HOUSE, which won a Silver
Leopard at Venice and Claude Berri's film starring Gerard
Depardieu, URANUS, screened in competition at Berlin.
Also screening are films from Japan and Korea: WHY DID BODHI DHARMA
LEAVE FOR THE EAST?: the re-issued Kurosawa classic DERSU UZALA and
CIRCUS BOYS, by talented young director Kaizo Hayashi.
DEATH IN BRUNSWICK is an Australian film starring Sam Neill, which
Derek Malcolm named as one of the Best Ten Films in the Edinburgh
Film Festival this year.
We are screening several of those titles, see if you can work out
which ones they are!
Festivals are one of the few opportunities in the UK for screening
short films, and we will be presenting not just this year's BFI New
Directors shorts, but also programmes of short films from film
schools in the UK and Canada, from the <head>Arts</head> Council
funded schemes and films funded by the filmmakers themselves.
</context>
</instance>
```

Figure 6.1: Example of the word *art* from Senseval-2 corpus

6.2.1 Feature Set

Three kinds of information were used to describe the examples and to train the classifiers. These features refer to local and topical contexts, and domain labels. These features are $SetA$ plus the topic features contained in $SetB$ (see section 3.2.2) and the domain information detailed below.

Domain Information: we enriched the basic set of features by adding semantic information in the form of domain labels. These domain labels were computed during a preprocessing step using the 164 domain labels linked to the nominal part of WordNet 1.6 (Magnini and Cavaglia, 2000). For each training example (e), a program gathered, from its context, all nouns and their synsets with the attached domain labels, and scores each domain d according to,

```
<instance id="work.097">
<answer instance="work.097" senseid="work_on%2:41:00::"/>
<context>
How could anyone know what to do with an assortment like that ? ?
Perhaps he had better have someone help him put up the pegboard and
build the workbench - someone who knew what he was about .
Then at least he would have a place to hang his tools and something
to <head sats="work_on.097:0">work</head>
<sat id="work_on.097:0">on</sat> .
</context>
</instance>
```

Figure 6.2: Example of phrasal verb *work on* from Senseval-2 corpus

```
<instance id="argument.n.bnc.00066601" docsrc="BNC">
<answer instance="argument.n.bnc.00066601"
senseid="argument%1:09:00::"/>
<answer instance="argument.n.bnc.00066601"
senseid="argument%1:10:02::"/>
<answer instance="argument.n.bnc.00066601" senseid="U"/>
<context>
The whole system was ill - conceived from the outset . That some very
large companies should have lost a great deal of money on their
little experiment shows only how unwise their decision to pursue the
technology was .  Look at the <head>arguments</head> . Since calls
must be made close to a node ,  then why not simply find a
telephone kiosk  must work these days and the cost will only be at
the standard rate .  Then there is the absurd lack of ability to
receive calls .
</context>
</instance>
```

Figure 6.3: Example of noun *argument* from Senseval-3 corpus

$$Weight(d) = \frac{ContextDomains(d)}{DomainFreqWN(d)} \cdot LabelsDomain$$

where *LabelsDomain* is the ratio of the total number of labelled WordNet synsets over the total number of domain labels; *DomainFreqWN(d)* is the frequency of the domain d on WordNet; and *ContextDomains(d)* is,

$$ContextDomains(d) = \sum_{w \in nouns(e)} \frac{WordDomains(w,d)}{SumWeights(w)}$$

where

$$WordDomains(w,d) = \sum_{\{s \in Synsets(w) | s \in d\}} \left(1 - \frac{Position(w,s) - 1}{|Synsets(w)|}\right)$$

and

$$SumWeights(w) = \sum_{s \in Synsets(w)} |Domains(s)| \cdot \left(1 - \frac{Position(w,s) - 1}{|Synsets(w)|}\right)$$

where *Position(w,s)* is the frequency position relative to WordNet of the synset s in the word w (e.g., 2 stands for the second most frequent sense of the word); $|Synsets(w)|$ stands for the number of synsets of the word w; and $|Domains(s)|$ is referred to the number of domains to which synset s is connected.

This function assigns to each example a set of domains labels depending on the number of domain labels assigned to each noun and their relative frequencies in the whole WordNet. Each domain label of the example is ranked by a weight, and the list of domains is pruned by a threshold. That is, the result of this procedure is the set of domain labels that achieve a score higher than a threshold of 0.04, which are incorporated as regular features for describing the example. Figure 6.4 shows a text example and the domains extracted applying the algorithms with their corresponding weights[4]. We used this information to built a bag of domains.

[4]The example has been applied using not only nouns, but also verbs, adjectives and adverbs. At Senseval-2 time, there were available domains for nouns only.

Example
A new **art** fair will take place at the G-Mex_Centre , Manchester from 18 to 21 March . **Paintings** , **drawings** and **sculpture** from every **period** of **art** during the last 350 **years** will be on **display** ranging from a Tudor **portrait** to contemporary British **art** .

domain	weight
painting	7.07
art	6.79
time_period	1.64
anthropology	1.39
sociology	1.21
quality	0.25
factotum	0.16

Figure 6.4: Text example with its corresponding domains

6.2.2 Preprocessing and Hierarchical Decomposition

As an initial experiment we selected a representative sample, containing the most frequent words of the Senseval-2 training data, and applied the LazyBoosting system (see section 3.4) straightforwardly on this sample. The results achieved after a 10-fold cross-validation procedure were extremely bad, mainly due to the fact that most of the words contain too many senses and too few examples per sense to induce reliable classifiers. With the aim of improving the performance of the learning algorithm, two different simplifications were carried out. Firstly, training examples consisting of a multiword have been processed separately. During training, multiwords have been saved into a separate file. At test time, all examples found in this multiword file are automatically tagged as multiwords. As an example, the word *bar* appears in the training set with 22 labels. But only the 10 senses showed in the left table of figure 6.5 are single words. The remaining 12 are multiwords which are considered unambiguous. In the official competition, labels 'U' and 'P' were completely ignored by our system. Thus, the examples labelled with these classes have not been considered during the training, and no test examples have been tagged with them.

Secondly, we have reduced the sense granularity, by hierarchically decomposing the learning process into two steps, following the work of Yarowsky (2000). At the first level, the learning algorithm was trained to classify among the labels corresponding to the WordNet semantic files, and, additionally, the semantic file labels with less than 10 training examples were automatically discarded. If less than two senses remain, no training is performed and the *Most-frequent-sense Classifier* is applied at testing time. As an example, for the word *'bar'*, in this first step the system is trained to classify between the labels of the top-right table of figure 6.5[5]. Note that senses *bar%1:04*, *bar%1:23* and *bar%1:17* have been discarded out because there are not enough training examples. At the second level, one classifier is trained for each of the resulting semantic-file labels of the first

[5]Numbers 06, 14, 10, 04, 23, and 17 correspond to the WordNet Lexicographical Files.

Full senses

Senses	Exs.
bar%1:06:04::	127
bar%1:06:00::	29
bar%1:06:05::	28
bar%1:14:00::	17
bar%1:10:00::	12
bar%1:06:06::	11
bar%1:04:00::	5
bar%1:06:02::	4
bar%1:23:00::	3
bar%1:17:00::	1

1st level

Senses	Exs.
bar%1:06	199
bar%1:14	17
bar%1:10	12

2nd level bar%1:06

Senses	Exs.
04::	127
00::	29
05::	28
06::	11

Figure 6.5: Sense treatment for word 'bar'

step in order to distinguish between its particular senses. Note that the same simplifying rules of the previous level are also applied. For instance, the bottom-right table of figure 6.5 shows the labels for *bar%1:06*, where *02::* has been rejected (few examples). When classifying a new test example, the classifiers of the two levels are applied sequentially. That is, the semantic-file classifier is applied first. Then, depending on the semantic-file label output by this classifier, the appropriate 2nd level classifier is selected. The resulting label assigned to the test example is formed by the concatenation of both previous labels.

Despite the simplifying assumptions and the loss of information, we observed that all these changes together significantly improved the accuracy on the training set. Next section includes further evaluation about this issue.

6.2.3 Evaluation

Figure 6.6 graphically shows the official results for the fine-grained accuracy of the Senseval-2 English Lexical Sample Task[6]. Table 6.2 shows the fine- and coarse-grained results of the 6 best systems on the same exercise. The evaluation setting corresponding to these results contains all the modifications explained in the previous sections, including the hierarchical approach to all words. The best system obtained 64.2% precision on fine-grained evaluation and 71.3% precision on coarse-grained. The TALP system was the

[6]This graphic has been downloaded from the official Senseval Web Page. See appendix C for web references.

sixth of 21 supervised systems and the difference from the first was of only 4.8% and 4.2% in fine-grained and coarse-grained precision, respectively.

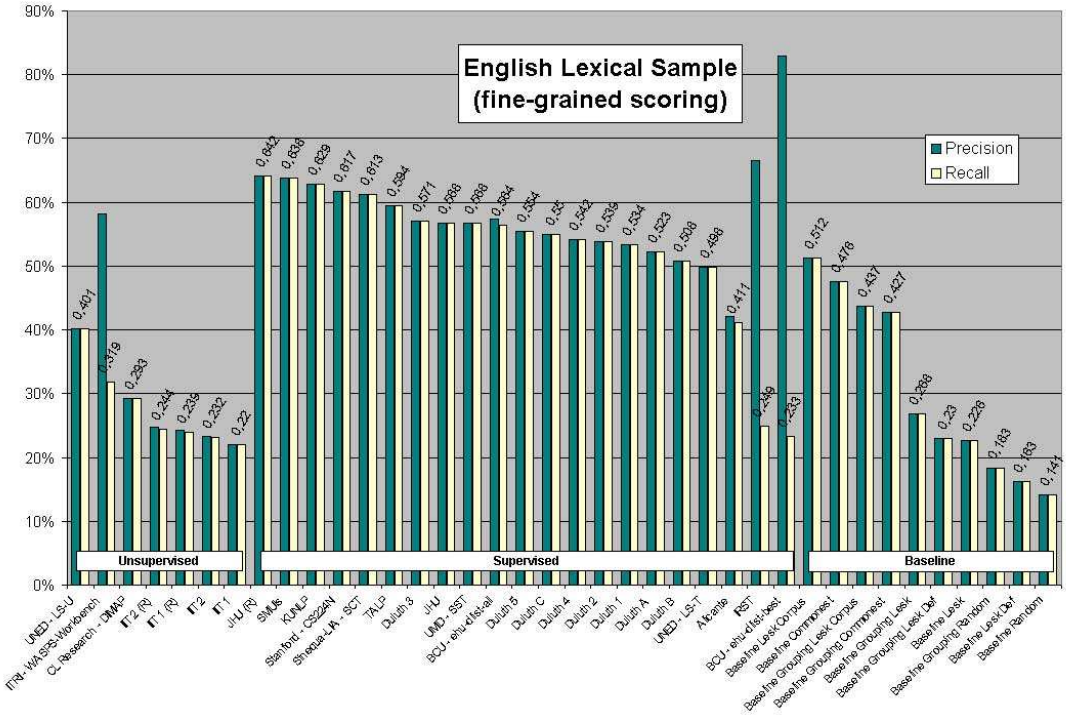

Figure 6.6: Senseval-2 Results

After the Senseval-2 event, we added a very simple Named-entity Recogniser to the part-of-speech tagger that was not finished at the time of the event, but the system continued completely ignoring the 'U' label. We also evaluated which parts of the system contributed most to the improvement in performance. Figure 6.7 shows the accuracy results of the four combinations resulting from using (or not) domain-label features and hierarchical decomposition. These results have been calculated over the test set of Senseval-2.

On the one hand, it becomes clear that enriching the feature set with domain labels systematically improves the results in all cases, and that this difference is especially noticeable in the case of nouns (over 3 points of improvement). On the other hand, the use of the hierarchies is unexpectedly useless in all cases. Although it is productive in some particular words (3 nouns, 12 verbs and 5 adjectives) the overall performance is significantly lower. A fact that can explain this situation is that the first-level classifiers do not succeed at classifying semantic-file labels with high precision levels (the average accuracy of first-level classifiers is only slightly over 71%) and that this important error is

system	fine	coarse
JHU(R)	64.2	71.3
SMUIs	63.8	71.2
KUNLP	62.9	69.7
CS224N	61.7	68.9
Sinequa	61.3	68.2
TALP	59.4	67.1

Table 6.2: Senseval-2 Global Results (in %)

dramatically propagated to the second-level, not allowing the greedy sequential application of classifiers. A possible explanation of this fact is the way semantic classes are defined in WordNet. Consider for instance work#1 (activity) and work#2 (production), they seem quite close but a system trying to differentiate among semantic fields will try to distinguish between these two senses. On the other extreme, such a classifier should group house#2 (legislature) with house#4 (family), which are quite different. Joining both situations makes a pretty difficult task.

Regarding multiword preprocessing (not included in figure 6.7), we have seen that is slightly useful in all cases. It improves the non hierarchical scheme with domain information by almost 1 point in accuracy. By part-of-speech, the improvement is about 1 point for nouns, 0.1 for verbs and about 2 points for adjectives.

In conclusion, the best results obtained by our system on this test set correspond to the application of multiword preprocessing and domain-labels for all words, but no hierarchical decomposition at all, achieving a fine-grained accuracy of 61.51% and a coarse-grained accuracy of 69.00% (from figure 6.7). We know that it is not fair to consider these results for comparison, since the system is tuned over the test set. Instead, we only wanted to fully inspect the TALP system and to know which parts were useful for future Word Sense Disambiguation systems.

6.3 From Senseval-2 to Senseval-3

This section is devoted to study the results of the best six systems of Senseval-2 English Lexical Sample task; trying to understand the behaviour of their components (e.g., if they use an Name-Entity Recogniser and how it works). After this study, we show the architecture of a new system built taking into account this new information. This system outperforms all previous six systems.

	nouns			
	no dom.		**with dom.**	
	fine	**coar.**	**fine**	**coar.**
no hier.	64.3	72.4	**67.9**	**75.6**
hier.	63.0	71.1	64.3	71.5
	verbs			
	no dom.		**with dom.**	
	fine	**coar.**	**fine**	**coar.**
no hier.	51.6	61.6	**52.1**	**62.6**
hier.	50.3	60.8	51.1	62.0
	adjectives			
	no dom.		**with dom.**	
	fine	**coar.**	**fine**	**coar.**
no hier.	66.2	66.2	**68.9**	**68.9**
hier.	65.4	65.4	68.2	68.2

Figure 6.7: Fine/coarse-grained evaluation (in %) for different settings and part-of-speech

6.3.1 System Decomposition

The Senseval-2 sense labels could be classified in different groups. These groups of labels could be tackled with different modules.

Table 6.3 shows the overall results of the six best systems of Senseval-2: *jhu(r)* (Yarowsky et al., 2001), *smuis* (Mihalcea and Moldovan, 2001), *kunlp* (Seo et al., 2001), *cs224n* (Ilhan et al., 2001), *sinequa* (Crestan et al., 2001) and *talp* (Escudero et al., 2001). The columns stands, respectively, for the fine-grained accuracy, the number of correct labelled examples in the test sample (in fine-grained evaluation), the coarse-grained accuracy and the number of examples correctly classified. There are 4,328 test examples in the corpus. All examples were treated by all systems (coverage = 100%). So, precision and recall are the same. Throughout all this study we have used the scorer of Senseval-2[7]. A difference of only 5 points in fine-grained and of 4 points in coarse-grained accuracy can be observed between the first and the last system.

Table 6.4 shows the agreement and kappa values between every pair of systems. Cells above the diagonal are the agreement between every pair of systems and those below the diagonal contain the kappa values. The maximum difference between pairs of systems is of 8% of agreement (from 0.60 to 0.68). The low agreements (65%) of all pairs show

[7]Available at the web site (see appendix C).

system	fine	ok	coarse	ok
JHU(R)	64.2	2,780	71.3	3,086
SMUIs	63.8	2,763	71.2	3,080
KUNLP	62.9	2,724	69.7	3,016
CS224N	61.7	2,670	68.9	2,981
Sinequa	61.3	2,653	68.2	2,953
TALP	59.4	2,571	67.1	2,903

Table 6.3: Senseval-2 Global Results (in %)

the big differences in the outputs of the systems. The kappa values differ more than simple agreement (from 0.22 to 0.35). This measure shows evidence of a large difference in performance among some of the systems.

	jhu	smuis	kunlp	cs224n	sinequa	talp
jhu	-	0.62	0.67	0.65	0.64	0.61
smui	0.34	-	0.66	0.60	0.64	0.60
kunlp	0.35	0.34	-	0.68	0.67	0.64
cs224n	0.29	0.22	0.31	-	0.67	0.65
sinequa	0.28	0.31	0.31	0.25	-	0.64
talp	0.27	0.27	0.30	0.25	0.26	-

Table 6.4: Agreement (above diagonal) and kappa (below diagonal) values between every pair of systems

Now we try to decompose the systems output (outcomes) in order to understand their results and the source of the differences among them. First, we clustered the test examples as follows: *P* stands for Proper-names, *mw* for Multiwords, *pv* for Phrasal Verbs, *U* for Unassignable and *ml* for the set of examples with regular senses (e.g., sense *bar%1:06:04::* of the noun *bar*).

Table 6.5 shows the performance of the six systems taking into account only the *P* label. To do that, all files (both keys and system votes) were pruned, and only *P* labels remained for testing. There are 146 *P* test examples in the corpus. This table shows, respectively, precision (pr), recall (rc), F1, number of correct labelled examples (ok), and number of examples attempted (att). Both fine-grained and coarse-grained results are the same, due to the fact that label *P* has no hierarchy. Two of these systems (*kunlp* and *talp*) had no *Name-Entity Recogniser*. Two of them detected proper nouns with high precision but a very poor recall, Sinequa and CS2224N (probably they tackled proper

nouns as another label of the learning problem). And finally, the two best systems had a component to preprocess proper nouns. SMUIs had the best .

system	pr	rc	F1	ok	att
JHU(R)	54.8	34.9	42.6	51	93
SMUIs	81.6	48.6	60.9	71	87
KUNLP	N/A	0	N/A	0	0
CS224N	66.7	1.4	2.7	2	3
Sinequa	88.9	5.5	10.4	8	9
TALP	N/A	0	N/A	0	0

Table 6.5: Senseval-2 results on Proper-Names (in %)

Table 6.6 shows multiword treatment and has the same format of table 6.5. There are 170 *multiword* test examples in the corpus. Multiwords appeared only for nouns and adjectives, but not for verbs. In this group there is again a significant difference between the two best system and the rest. They had the best multiword processor.

system	pr	rc	F1	ok	att
JHU(R)	85.8	74.7	79.9	127	148
SMUIs	88.8	68.0	77.0	151	222
KUNLP	53.9	48.8	51.2	83	154
CS224N	78.4	34.1	47.5	58	74
Sinequa	90.3	54.7	68.1	93	103
TALP	89.3	64.1	74.6	109	122

Table 6.6: Senseval-2 results on Multiwords (in %)

Table 6.7 has the same format as the previous two. It shows the results on the set of phrasal verbs. There are 182 *phrasal verbs* test examples in the corpus. Phrasal verb labels were in a hierarchy. For this reason this table contains fine-grained and coarse-grained results. Again, there are big differences. They ranged from 43.0% to 66.5% on precision and from 26.9% to 65.4% of recall.

Next, table 6.8 shows the behaviour on the *U*-labelled examples. This label has no hierarchy. Only 3 systems treated this label, and they achieved very low recall results. There are 160 *U* test examples in the corpus.

Finally, the results on the core *ml* set of examples are shown in table 6.9. There are 3.733 *ml* test examples in the corpus. The results for nouns (1,412 examples), verbs

	fine				coarse				
system	**pr**	**rc**	**F1**	**ok**	**pr**	**rc**	**F1**	**ok**	**att**
JHU(R)	66.5	65.4	66.0	119	71.5	70.3	70.9	128	179
SMUIs	58.0	57.7	57.9	105	65.7	65.4	65.6	119	181
KUNLP	49.8	58.8	53.9	107	53.5	63.2	58.0	115	215
CS224N	43.0	26.9	33.1	49	50.9	31.9	39.2	58	114
Sinequa	59.5	42.9	49.9	78	65.6	47.3	55.0	86	131
TALP	45.5	39.0	42.0	71	55.1	47.3	50.9	86	156

Table 6.7: Senseval-2 results on Phrasal Verbs (in %)

system	**pr**	**rc**	**F1**	**ok**	**att**
JHU(R)	70.4	23.7	35.5	38	54
SMUIs	N/A	0	N/A	0	0
KUNLP	N/A	0	N/A	0	0
CS224N	55.8	15.0	23.6	24	43
Sinequa	61.5	10.0	17.2	16	26
TALP	N/A	0	N/A	0	0

Table 6.8: Senseval-2 results on the U label examples (in %)

(1,630 examples), and adjectives (691 examples) are presented by tables 6.10, 6.11 and 6.12, respectively.

From these results, the first system in the competition, JHU(R), is not the best system (from the machine learning perspective) for nouns and verbs -and nouns and verbs represented more than the half part of the test set. The main reason for obtaining the best results resides in performing a better preprocessing of examples and treatment of U, phrasal verbs, multiwords and proper-nouns cases. That is, it treated better than the rest multiword, phrasal verbs, proper-nouns and the *U* label; maintaining a competitive performance on the *ml* set. There are no overall *ml* significant differences. SMUIs and KUNLP are better in verbs, and SMUIs in nouns; but JHU(R) is extremely good with adjectives.

The interaction between all components of the systems makes difficult to compare the Machine Learning algorithms. For this reason, we have extracted the intersection set of all the examples in which every system have output a regular sense. This set has 3,520 examples. The results are shown in table 6.13. Here, we can see that the four best systems have very similar performance being KUNLP slightly better than the rest.

	fine				coarse				
system	**pr**	**rc**	**F1**	**ok**	**pr**	**rc**	**F1**	**ok**	**att**
JHU(R)	60.1	62.1	61.1	2,317	68.1	70.3	69.2	2,623	3,854
SMUIs	60.3	62.0	61.1	2,315	68.5	70.4	69.4	2,629	3,838
KUNLP	59.1	62.6	60.8	2,338	66.4	70.5	68.4	2,630	3,959
CS224N	57.0	62.5	59.6	2,334	64.5	70.7	67.5	2,640	4,094
Sinequa	55.8	60.6	58.1	2,263	63.2	68.7	65.8	2,565	4,059
TALP	54.3	59.0	56.6	2,201	62.2	67.5	64.7	2,521	4,050

Table 6.9: Senseval-2 results on the *ml* set of examples (in %)

	fine				coarse				
system	**pr**	**rc**	**F1**	**ok**	**pr**	**rc**	**F1**	**ok**	**att**
JHU(R)	61.5	65.9	63.6	930	69.3	74.3	71.7	1,049	1,513
SMUIs	63.8	67.8	65.7	957	72.5	77.1	74.7	1,088	1,501
KUNLP	58.0	67.4	62.4	951	65.4	75.9	70.3	1072	1,639
CS224N	59.1	69.3	63.8	978	66.1	77.5	71.4	1,094	1,655
Sinequa	57.0	65.9	61.1	931	64.4	74.6	69.1	1,053	1,634
TALP	55.6	65.5	60.2	925	63.4	74.7	68.6	1,055	1,664

Table 6.10: Senseval-2 results on the nouns of the *ml* set of examples (in %)

Summarising, the best system did not only focus on the learning algorithm. But it had the best components treating all the special labels. Thus, preprocessors were very important to construct a competitive WSD systems at Senseval-2.

6.3.2 The TalpSVM System

After the previous study we developed a system (hereinafter *TalpSVM*) taking into account several points to improve the TALP system at competition time:

- First, new features have been generated to improve the performance of the system. After extracting the features, the features that appears in less than 4 examples have been eliminated.
- Second, Support Vector Machines have been chosen as learning algorithm, due to its better performance in the presence of small training corpora (see chapter 3). Again, we have used SVM^{light} implementation of Support Vector Machines. The learner

	fine				coarse				
system	**pr**	**rc**	**F1**	**ok**	**pr**	**rc**	**F1**	**ok**	**att**
JHU(R)	54.9	54.8	54.9	894	66.4	66.3	66.4	1,081	1,627
SMUIs	55.5	55.3	55.4	902	66.8	66.6	66.7	1,085	1,625
KUNLP	58.3	56.9	57.6	927	69.0	67.4	68.2	1,098	1,591
CS224N	52.4	54.4	53.4	887	63.7	66.1	64.9	1,077	1,692
Sinequa	52.4	53.9	53.1	878	63.2	64.9	64.0	1,058	1,675
TALP	51.4	52.0	51.7	848	62.9	63.7	63.3	1,038	1,650

Table 6.11: Senseval-2 results on the verbs of the *ml* set of examples (in %)

	fine				coarse				
system	**pr**	**rc**	**F1**	**ok**	**pr**	**rc**	**F1**	**ok**	**att**
JHU(R)	69.0	71.3	70.1	493	69.0	71.3	70.1	493	714
SMUIs	64.0	66.0	65.0	456	64.0	66.0	65.0	456	712
KUNLP	63.1	66.6	64.8	460	63.1	66.6	64.8	460	729
CS224N	62.8	67.9	65.3	469	62.8	67.9	65.3	469	747
Sinequa	60.5	65.7	63.0	454	60.5	65.7	63.0	454	750
TALP	58.2	61.9	60.0	428	58.2	61.9	60.0	428	736

Table 6.12: Senseval-2 results on the adjectives of the *ml* set of examples (in %)

takes as input all the examples of the training set that are not tagged as P, U, multiword neither phrasal verb, and label U has not been considered in the training step neither in the test step.

- Third, the addition of new examples from other corpora have been tried to compensate for the small number of examples of Senseval-2 corpora.

- And finally, we have tried to improve all preprocessing components.

Each example is labelled with 3 linguistic processors[8]: the FreeLing Part-of-Speech tagger (Atserias et al., 2006), from which the system extracts the lemmas of the words and the Part-of-Speech tag; the Minipar parser (Lin, 1998), from which the syntactical relations are extracted as features for the learners; and, finally, the TnT tagger (Brants, 1997), from which the system extract the *P* tag (it is the target word tagged as Named-Entity by the tagger).

[8]See appendix C for web references of these 3 linguistics processors.

	accuracy	
Methods	**fine**	**coarse**
JHU(R)	65.5	72.9
SMUIS	65.2	72.9
KUNLP	66.0	73.4
CS224N	65.7	73.4
Sinequa	63.4	70.8
TALP	61.6	69.6

Table 6.13: Comparison of the learning component of the best performing systems at Senseval-2 (accuracy in %)

Features

We intended to test the impact of different feature groups. The set of features was incrementally increased by adding big groups of related features. All the features included are binary. These feature groups are:

- **forms:** includes features about: the form of the target word, the part-of-speech and the location of the words in a ±3-word-window around the target word, the form and the location of the words in a ±3-word-window around the target word, collocations of part-of-speech and forms is a ±3-word-window, a bag of word forms of the text, a bag of word forms in a ±10-word-window, bigrams of the text, bigrams in a ±10-word-window, pairs of open-class-words in the text, pairs of open-class-words in a ±10-word-window, the form of the first noun/verb/adjective/adverb on the left/right, and finally, the form of the first noun/verb/adjective/adverb on the left/right in a ±10-word-window.
- **lemmas:** the same as *forms*, but taking the lemmas instead of word forms.
- **both:** the union set of both *forms* and *lemmas*.
- **domains:** a bag of domains extracted with the algorithm of section 6.2.1, but using all nouns, verbs, adjectives and adverbs instead of only nouns.
- **sumo:** the same type but using **sumo** ontology labels[9](Niles and Pease, 2001) instead of those of Magnini and Cavaglia (2000).

[9] `http://www.ontolgyportal.org`

- **minipar:** a bag of syntactical relations extracted from the Minipar parser. We considered also a feature that only has the syntactical relations of the target word.

Table 6.14 shows the result of the system using the different sets of features. As we can see, all features sets seems to improve the performance of the system.

system	fine	coarse
forms	62.6	69.7
lemmas	63.9	70.5
both	63.8	70.5
both+sumo	63.8	70.7
both+domains	64.3	71.0
both+minipar	63.7	70.7
both+all	**64.6**	**71.2**

Table 6.14: TalpSVM results using different feature sets (in %)

New Examples

Table 6.16 shows the effect of the addition of training examples from other corpora. **wne** stands for the examples in WordNet glosses; **wng** for the definitions in WordNet glosses; **sc** stands for sentences in the Semcor corpus[10]; and **se2** represents the original Senseval-2 English Lexical Sample task corpus. Table 6.15 shows the amount of examples extracted from each of these corpora.

source	acr.	number of examples
WN Examples	*wne*	549
WN Glosses	*wng*	3,115
Semcor	*sc*	7,193

Table 6.15: Number of examples extracted from other corpora

Surprisingly, the addition of examples from other sources implies a decrease in accuracy results (see table 6.16). This could be caused by the difference in the format of the examples in the corpora.

[10]See appendix C for Web references.

system	fine	coarse
se2	63.6	70.5
se2+wne	63.4	70.3
se2+wng	62.3	69.5
se2+sc	62.4	70.2
all	62.0	70.0

Table 6.16: TalpSVM results adding different example sets (accuracy in %)

Preprocess

Table 6.17 shows the results of the proper-name component. Every row represents respectively: the use of the *P* label, the use of the *ml* corresponding label, and the use of both labels for each example. The best option corresponds to *ml* that shows the lack of utility of the component and evincing that a better Name-Entity Recogniser is needed.

system	fine	coarse
P	64.6	71.2
ml	64.8	71.7
P+ml	64.7	71.4

Table 6.17: TalpSVM results using a Name-Entity Recogniser (accuracy in %)

The multiword component takes the information provided by Senseval-2 organisers (a list of the multiwords of appearing in the corpus plus its corresponding sense). This multiword list have been indexed and compared with the test examples in order to detect them. Table 6.18 shows the behaviour of this component. **mw** stands for the use of multiwords component labels and *ml* for the Machine Learning classifier. The use of this component increases the fine-grained accuracy in 3.6% (from 61.6 to 64.8). The last row represents the vote of both the multiword and that of the classifier.

system	fine	coarse
mw	64.8	71.7
ml	61.6	68.6
mw+ml	63.2	70.2

Table 6.18: TalpSVM and multiword (in %)

Phrasal verbs have been tagged with the most frequent sense of the training set; or,

if they did not appear in training set, with the most frequent sense of WordNet. The behaviour of the system including this component is shown in table 6.19. An increase of 2% in accuracy is observed when using the phrasal verbs component.

system	**fine**	**coarse**
pv	64.8	71.7
ml	62.7	69.3
pv+ml	63.2	69.8

Table 6.19: TalpSVM and phrasal verbs (in %)

6.3.3 Global Results

After the development, *TalpSVM* has been compared with the six best systems of Senseval-2 competition. Tables 6.20 and 6.21 contain the results of the system to be compared with those in tables from 6.3 to 6.13. *P* stands for the use of *P* label in test and *noP* for ignoring it, respectively. *TalpSVM.P* outperforms all six systems in both accuracy, fine and coarse grained (table 6.2). The same applies for *TalpSVM.noP*. Also, it is the most similar (tables 6.4 and 6.21) to *JHU(R)*, the winner, and to *KUNLP*, the best learning system. Table 6.20 (proper names) shows a high recall of the *TalpSVM* in tagging *P* labels but a very low precision, that means that the proper-names component is overassigning *P* labels to test examples. This component is counterproductive. The multiword component works very well (tables 6.6 and 6.20). It achieves the best precision and a very high recall. Phrasal verbs have been tagged with the most frequent sense. This component works as the average of the systems. It is one of the candidates to be improved. And finally, tables from 6.10 to 6.13 show that *TalpSVM.noP* has the best precision on nouns, verbs, and overall. This, seems to be caused by the Support Vector Machines algorithm. It works pretty well with small sets of training examples and is very robust in front of different kind of features. We must remark, that the final system outperforms all the systems presented at Senseval-2.

6.4 Senseval-3

This section describes the TALP system on the English Lexical Sample task of the Senseval-3 event (Mihalcea et al., 2004). The system is fully supervised and relies also on Support Vector Machines. It does not use extra examples than those provided by

	fine				coarse					
subcorpus	**pr**	**rc**	**F1**	**ok**	**pr**	**rc**	**F1**	**ok**	**att**	**table**
global.P	64.6	–	–	2,794	71.1	–	–	3,078	4,328	6.3
global.noP	**64.8**	–	–	2,806	**71.7**	–	–	3,105	4,328	6.3
proper-names	54.3	82.9	65.6	121	–	–	–	–	223	6.5
multiwords	92.4	85.9	89.0	146	–	–	–	–	158	6.6
phrasal-verbs	52.5	51.1	51.8	93	61.6	59.9	60.7	109	177	6.7
unassignable	N/A	0	N/A	0	–	–	–	–	0	6.8
ml.all.P	62.2	62.8	62.5	2,346	69.5	70.2	69.9	2,622	3,770	6.9
ml.all.noP	60.3	64.5	62.3	2,408	67.3	72.0	69.6	2,689	3,993	6.9
ml.nouns.P	64.1	65.9	65.0	931	71.1	73.2	72.1	1,033	1,453	6.10
ml.nouns.noP	59.6	69.1	64.0	975	66.1	76.6	71.0	1,082	1,637	6.10
ml.verbs.P	58.6	58.6	58.6	955	69.3	69.3	69.3	1,129	1,629	6.11
ml.verbs.noP	58.6	58.6	58.6	955	69.3	69.3	69.3	1,129	1,629	6.11
ml.adjs.P	66.9	66.6	66.8	460	66.9	66.6	66.8	460	688	6.12
ml.adjs.noP	65.7	69.2	67.4	478	65.7	69.2	67.4	478	727	6.12
learner	**66.8**	–	–	–	**73.6**	–	–	–	3,520	6.13

Table 6.20: TalpSVM performance on Senseval-2 data, comparison to the best performing systems is included by referencing to previous tables of results [last column]

Senseval-3 organisers, though it uses external tools and ontologies to extract part of the representation features.

Three main characteristics have to be highlighted from the system architecture. The first thing is the way in which the multiclass classification problem is addressed using binary SVM classifiers. Two different approaches for binarising multiclass problems have been tested: *One vs All* and *Constraint Classification*. In a cross-validation experimental setting the best strategy has been selected at word level.

The second characteristic is the rich set of features used to represent training and test examples; extending the feature set of the previous section. *Topical* and *local context* features are used as usual, but also syntactic relations and semantic features indicating the predominant semantic classes in the example context are now taken into account.

And finally, since each word represents a learning problem with different characteristics, a per-word feature selection has been applied.

	agree.	kappa
jhu	0.70	0.41
smui	0.66	0.36
kunlp	0.71	0.40
cs224n	0.69	0.29
sinequa	0.67	0.29
talp	0.68	0.37

Table 6.21: Agreement and kappa values between TalpSVM and the best six systems of Senseval-2

6.4.1 Learning Framework

One of the issues when using SVM for WSD is how to binarise the multiclass classification problem. The two approximations tested in the TALP system are: (1) the usual *one-versus-all* and (2) the recently introduced *constraint-classification* framework (Har-Peled et al., 2002).

The **one-versus-all** approach decompose the problem in as many binary problems as classes has the original problem; then, one classifier is trained for each class trying to separate the examples of that class (positives) from the examples of all other classes (negatives). This method assumes the existence of a separator between each class and the set of all other classes. When classifying a new example, all binary classifiers predict a class and the one with highest confidence is selected (*winner-takes-all* strategy).

Constraint classification (Har-Peled et al., 2002) is a learning framework that generalises many multiclass classification and ranking schemes. It consists of labelling each example with a set of binary constraints indicating the relative order between pairs of classes. For WSD, these binary constraints consists of a sense to which the example belongs and a sense to which the example does not belong. For instance, if we have 4 possible senses {1, 2, 3, 4} and a training example with labels 2 and 3, the constraints corresponding to the example are {(2>1), (2>4), (3>1), and (3>4)}. The aim of the methodology is to learn a classifier consistent with the partial order defined by the constraints. Note that here we are not assuming that perfect separators can be constructed between each class and the set of all other classes. Instead, the binary decisions imposed are more conservative.

Using Kesler's construction for multiclass classification, each training example is expanded into a set of (longer) binary training examples. Finding a vector-based separator in this new training set is equivalent to find a separator for each of the binary constraints imposed by the problem. The construction is general, so we can use SVMs directly on

the expanded training set to solve the multiclass problem. See Har-Peled et al. (2002) for details.

6.4.2 Features

The basic feature set is that detailed in section 6.3.2, with a few extensions. Local and topic features are the same. Knowledge-based features have been extended by using the formula in section 6.2.1 to other semantic resources into the Multilingual Central Repository (MCR) (Atserias et al., 2004) of the MEANING Project: Sumo labels, WordNet Lexicographic Files, and EuroWordNet Top Ontology. Also, it has been extended with features codifying the output of Yarowsky's dependency pattern extractor[11].

Appendix B details part of the output of the feature extractor for an example of Senseval-2 corpus.

Since the set of features is highly redundant, a selection process has been applied. It is detailed in the next section.

6.4.3 Experimental Setting

For each binarisation approach, we performed a feature selection process consisting of two consecutive steps:

- **Feature selection by PoS**: Using the Senseval-2 corpus, an exhaustive selection of the best set of features for each particular Part-of-Speech has been performed. These feature sets are taken as the initial sets in the feature selection process of Senseval-3. Table 6.22 shows which features have been selected for each Part-of-Speech (see appendix B for an explanation of the labels of the first two columns of the table).
- **Feature selection by word**: We applied a forward(selection)-backward(deletion) two step procedure to obtain the best feature selection per word. For each word, the process starts with the best feature set obtained in the previous step according to its Part-of-Speech. In the selection part, we consider those features not selected during feature selection by PoS, adding all features which produce some improvement. During deletion, we consider only those features selected during feature selection by PoS, removing all features whose deletion produces some improvement. Although this addition-deletion procedure could be iterated until achieving no further improvements, we only performed a unique iteration because its computational overhead.

feature	**feature group**	**nouns**	**verbs**	**adjs**
form	*form*	X	X	-
all	*locat*	X	X	X
pos	*coll*	-	X	X
form	*coll*	X	-	-
lemma	*coll*	X	X	X
all	*coll2*	X	X	X
form noun/verb	*first sent.*	-	X	-
form adj	*first sent.*	-	X	X
lemma noun/verb/adv	*first sent.*	X	X	X
lemma adj	*first sent.*	X	-	X
form noun/verb/adv	*first ±10w*	X	X	X
form adj	*first ±10w*	X	X	-
lemma	*first ±10w*	X	X	X
form	*topic sent.*	-	-	X
lemma	*topic sent.*	X	X	-
all	*topic ±10w*	-	-	X
form	*sbig sent.*	-	X	X
lemma	*sbig sent.*	-	-	X
all	*sbig ±10w*	X	X	X
form	*comb sent.*	-	X	-
form	*comb ±10w*	-	X	-
lemma	*comb ±10w*	X	X	X
-	*f_sumo*	X	X	X
-	*a_sumo*	X	X	X
-	*f_semf*	-	X	-
-	*a_semf*	X	X	X
-	*f_tonto*	X	-	-
-	*a_tonto*	X	X	X
-	*f_magn*	X	X	X
-	*a_magn*	X	X	X
-	*tgt_mnp*	X	X	X
-	*rels_mnp*	X	X	X
all	*yar_**	X	X	X

Table 6.22: Selected Features per Part-of-Speech.

The result of this preprocess is the selection of the best binarisation approach with its best feature set for each word.

Regarding the implementation details of the system, we used again SVM^{light} (Joachims, 2002). A simple lineal kernel with a regularisation C value of 0.1 was applied. This parameter was empirically decided on the basis of our previous experiments on the Senseval-2 corpus. Again, previous tests using non-linear kernels did not provide better results. The selection of the best feature set and the binarisation scheme per word described above, have been performed using 5-fold cross-validation on the Senseval-3 training set. The five partitions of the training set were obtained maintaining, as much as possible, the initial distribution of examples per sense (using the stratified sampling scheme).

After several experiments considering the U label as an additional regular class, we found that we obtained better results by simply ignoring it. Then, if a training example was tagged only with this label, it was removed from the training set. If the example was tagged with this label and others, the U label was also removed from the learning example. In that way, the TALP system do not assigns U labels to the test examples. Due to lack of time, the TALP system presented at the competition time did not include a complete model selection for the constraint classification binarisation setting. More precisely, 14 words were processed within the complete model selection framework, and 43 were adjusted with only the one-vs-all approach with feature selection but not the constraint classification approach. After the competition was closed, we implemented the constraint classification setting more efficiently and we reprocessed again the data.

6.4.4 Evaluation

Table 6.23 shows the accuracy obtained in the training set by 5 fold cross-validation. *OVA(base)* stands for the results of the one-versus-all approach on the starting feature set; *CL(base)* for the constrain-classification on the starting feature set; and *OVA(best)* and *CL(best)* mean one-versus-all and constraint-classification with their respective feature selection. *Final* stands for the full architecture on the training set and *SE3* for the results obtained at competition time[12].

Globally, feature selection consistently improves the accuracy in 3 points (both in OVA and CL). Constraint-classification is slightly better than one-vs-all approach; but the Final (the full architecture) results shows that it is not true for all the words. Finally, the combined feature selection increases the accuracy in half a point.

[11] This software was kindly provided by David Yarowsky's group, from Johns Hopkins University.
[12] Only 14 words where processed at competition time due to lack of time.

method	acc.
OVA(base)	72.38
CL(base)	72.28
OVA(best)	75.27
CL(best)	75.70
SE3	75.62
Final	76.02

Table 6.23: Results on the Senseval-3 5 fold cross-validation training set (accuracy in%)

Table 6.24 shows the official results of the system. *SE3* means the official result[13], *Final* means the complete architecture, and *Best* means the winner of the competition.

measure	SE3	Final	Best
fine	71.3	71.6	72.9
coarse	78.2	78.2	79.3

Table 6.24: Official Senseval-3 Results (accuracy in%)

However, when testing the complete architecture on the official test set, we obtained an accuracy decrease of more than 4 points. Nevertheless, the system obtained high competitive results: ranking in the 7th team position, at only 1.3 points on fine-grained accuracy and by 1.1% on coarse-grained accuracy below the winner.

6.4.5 Extending the Feature Selection process

This section presents the results of the application of three new feature selection methods using the Senseval-3 corpora.

The first method is a greedy iterative process (hereinafter *bf*) in which the best feature (in accuracy) is selected at each step. The initial features of this method is the empty set. This approximation is local and greedy. Sometimes, a pair of features would be selected in order to improve the overall. And once a feature is selected it can not be deleted. A variant of this method consisting of using as initial set those in table 6.22 has been called *wbf*.

The third process, hill climbing (hereinafter *hc*), aims to avoid the problem of selecting a bad feature. It is also a greedy iterative process. First and second steps are like the

[13]Only 14 words were processed with the full architecture.

previous one. That is, the process selects the best two features in two iterations. From the third step further, each step looks for the best feature (in accuracy) to add and the best to delete (if it exists a feature that increases the accuracy in being deleted). The initial set of this method is also the empty set.

Evaluation

Table 6.25 shows the result of the three feature selection methods on the Senseval-3 corpora for nouns, verbs, adjectives and overall. The first method (*wrd*) is the one we presented to the Senseval-3 event, achieving a fine-grained accuracy of 71.6 and a coarse one of 78.2. Although every procedure achieves the best performance on some words, the best overall system is *wrd*. That is, none of feature selection methods improved the original results.

	pos	**wrd**	**bf**	**hc**	**wbf**
nouns	71.9	72.1	70.8	69.2	72.0
verbs	72.3	73.0	72.1	72.3	72.8
adjectives	50.3	49.1	49.7	44.0	49.1
all	71.2	71.6	70.6	69.8	71.5

Table 6.25: Fine-grained Results.

The basic aim of table 6.26 is showing the impact on accuracy of the feature selection procedures for the seven first words of Senseval-3 English Lexical Sample corpus. *ova-wrd* stands for one versus all binarisation plus wrd feature selection procedure, *cc-wrd* for constraint classification plus wrd selection, and so on. *top*, *loc*, *dom* and *syn* stands respectively for topical, local, domain and syntactical features. So, the four first columns of the table show the number of words for which the method has selected some of those features. The last column (*avg*) shows the average number of features selected by the method.

It seems that generally one versus all selects more features than constrain classification. *wrd* and *wbf* feature selection procedures gather much more features (around 50 types of them) than the others (less than 20). While *wrd* and *wbf* methods start with a large feature set (approximately 40 feature types depending of the part-of-speech), *bf* and *hc* begin with the empty set. This explains the differences between them. Furthermore, *bf* selects the double number of features than *hc* because *hc* deletes features in its iterative process.

Regarding the 4 clusters of features, all methods have selected local features for all the

	loc	top	dom	syn	avg
ova-wrd	7	7	7	7	49.7
cc-wrd	7	7	7	7	48.6
ova-bf	7	7	4	4	16.1
cc-bf	7	4	4	3	9.9
ova-hc	7	6	5	4	17.7
cc-hc	7	2	3	3	10.4
ova-wbf	7	7	7	7	48.6
cc-wbf	7	7	7	7	49.9

Table 6.26: Features Selection Information.

words. The domain features from the MCR are equally selected as topical features and selected more often than syntactical features.

6.5 Conclusions

We have participated in the English Lexical Sample task of last two editions of Senseval international evaluation exercises. Our systems have achieved a very high position in both competitions. That is, they are part of the State-of-the-Art in supervised WSD systems.

At Senseval-2, we proposed a system based on LazyBoosting, an improved version of the AdaBoost.MH. We enriched the typical local and topical context with domain information. After the exercise, we have studied the outputs of the six best system and proposed a new system based on SVM that outperformed all the six systems presented at the competition.

At Senseval-3, we proposed and tested a supervised system in which the examples are represented through a very rich and redundant set of feature (using the information content integrated within the Multilingual Central Repository of the Meaning project), and which performs some specialised selection of features and multiclass binarisation process for each word. All these features lead to a better system able to be automatically configured for every word. The main drawback of the approach presented in this work is the high computational overhead of the feature selection process for every word.

The system based on Support Vector Machines and a rich set of features has a very high competitive performance. The features based on domains bring new complementary and useful information.

This work suggests some promising lines to be studied in detail: 1) Searching new ways of representing the examples: for instance, the learning algorithms could use the output

information of unsupervised systems as features; and 2) improving the per-word feature selection: for instance, some other procedures should be tested, such as descendent hill climbing[14].

Finally, there are some open questions: 1) Which set of features better represent each sense of each word? 2) How do we have to binarise the multiclass problem? And 3) How do we have to treat examples with more than one possible sense (multilabel assignments)?

[14]That is, starting from the full feature set and deleting features at each iteration.

Chapter 7

Conclusions

7.1 Summary

Word Sense Disambiguation is one of the tasks for dealing with NLP semantics. First references on Lexical Ambiguity are sixty years old. Despite the work devoted to the problem, it can be said that no large-scale broad-coverage accurate WSD system has been built up to date (Kilgarriff and Rosenzweig, 2000). Therefore, WSD is one of the most important open problems in NLP, with current state-of-the-art accuracy in the 60-70% range. Machine Learning classifiers are effective, but, due to the knowledge acquisition bottleneck, they will not be feasible until reliable methods for acquiring large sets of training examples with a minimum human annotation effort will be available. These automatic methods for helping in the collection of examples should be robust to noisy data and changes in sense frequency distributions and corpus domain (or genre). The adequate ML algorithms used to induce the WSD classifiers should be also noise-tolerant (both in class-label and feature values), easy to adapt to new domains, robust to overfitting, and efficient for learning thousands of classifiers in very large training sets and feature spaces.

Many NLP researchers consider word sense ambiguity as a central problem for several Human Language Technology applications, (e.g., Machine Translation, Information Extraction and Retrieval, etc.). This is also the case for most of the associated sub-tasks, for instance, reference resolution and parsing. For this reason, many international research groups have been actively working on WSD during last years, using a wide range of approaches. This can be noticed throughout the recent publications in the main NLP conferences and journals (including special issues on the WSD task), the organisation of specialised conferences on this topic, a number of current NLP research projects including WSD among their goals, and the increasing number of participants in the different editions

of the Senseval international competitions.

The Senseval international evaluation exercises (organised by the ACL SIGLEX) introduced two main evaluation approaches on Word Sense Disambiguation. The *Lexical Sample* proposes to disambiguate only a word in a sentence. Typically these words are the most polysemous and difficult to solve and the Senseval organisers provide a set of training examples for each word. The *All Words* task proposes the evaluation on all the open-class words of a text, and the Senseval organisers do not provide training examples.

The Best Systems at the *English Lexical Sample* task on the last Senseval editions obtained accuracy results lower than 75%. Obviously, this figures are not useful for practical purposes. We think that the best way to improve the current systems is to exploit the representation of the examples with linguistic knowledge that is not currently available in wide-coverage lexical knowledge bases. We refer, for instance, to sub-categorisation frames, syntactic structure, selectional preferences, and domain information. That is, an in depth study of the features is needed (the better we represent examples, the better learning systems will be obtained). One of the biggest problems of this approach is the dependency of the classifiers on the training corpora. We need an in depth study on the dependency on the corpora and a way to develop general corpora or effective and simple ways to adapt to new application domains.

The Best Systems at the *All-words* task on last Senseval edition obtained accuracy results near 65%. Following the supervised approach, we need a large amount of labelled corpora to build a broad coverage Word Sense Tagger; which are very expensive. The most promising line to tackle this problem is the application of bootstrapping techniques to profit from unlabelled examples. This is one of the interesting current research lines on Word Sense Disambiguation field and other NL-related tasks. Moreover, future WSD systems will need to automatically detect and collapse spurious sense distinctions, as well as to discover, probably in an on-line learning setting, occurrences of new senses in running text.

We also think that it might be very useful to develop more complex systems dealing with Semantics from several points of view, like systems based on Semantic Parsing.

Finally, we think that there are some general outstanding problems within Word Sense Disambiguation dealing with semantics. The most important one is the demonstration, on a real application, of the usefulness of Word Sense Disambiguation. Another open issue is the utility of WSD in semantics to develop Natural Language Understanding Systems.

7.2 Contributions

This section summarises the main contributions of the thesis. We have grouped them based on the same chapters of this document:

Comparison of supervised ML algorithms for WSD: First, we have performed a direct comparison of the most used algorithms by the community. As a complementary issue, we have clarified some confusing information on the literature and developed some improvements on the adaptation (representation of the information and efficiency) of some algorithms. We have empirically showed that the best systems on Word Sense Disambiguation data are those based on margin maximisation. This evaluation is based not only on accuracy values, but also on agreement and Kappa statistics.

Cross-corpora evaluation and adaptation to new domains: The experiments of chapter 4 have empirically shown the dependency of the classifiers based on Machine Learning to the training corpora. This issue has to be solved in order to develop a real and useful Supervised Word Sense tagger. This is an important issue because until knowing the way of building full general corpora, training less dependant classifiers, or adapting classifiers to new domains; the only taggers we can build are those based on particular domains.

Data scarceness, bootstrapping: The contributions here are the application of Transductive Support Vector Machines and the Greedy Agreement bootstrapping algorithm to Word Sense Disambiguation. The first approach does not seem to work on this problem appropriately. Instead, the Greedy Agreement algorithm shows better performance. The first experiments seem to indicate a promising future line of research.

International evaluation exercises: We have participated to the English Lexical Sample task of both Senseval-2 and 3 competitions obtaining fairly good results. At Senseval-2 our system got the fifth position at 4.8% of accuracy from the first one, and at Senseval-3 it got the seventh position at only 1.6% of accuracy. In both cases our system was within the set of best performing systems. That is, our systems are among the State-of-the-Art systems on Word Sense Disambiguation. Improving the systems involved in Senseval exercises have produced: a comparative study of the six best systems from Senseval-2, and some improvements of the learning systems presented at competition time: 1) the addition of new useful features, such as those based on domain information; 2) the

application of feature selection procedures, and the comparison between them; and 3) the application of different ways of binarising the multiclass problem when using binary classifiers, such as SVM.

7.3 Publications

This section lists the publications ordered by the contribution topics of the previous section:

Comparison of supervised ML algorithms for WSD:

G. Escudero, L. Màrquez, and G. Rigau. Boosting Applied to Word Sense Disambiguation. In *Proceedings of the 12th European Conference on Machine Learning, ECML*, 2000. Explained in section 3.4.

G. Escudero, L. Màrquez, and G. Rigau. Naive Bayes and Exemplar-Based Approaches to Word Sense Disambiguation Revisited. In *Proceedings of the 14th European Conference on Artificial Intelligence, ECAI*, 2000. Explained in section 3.3.

L. Màrquez, G. Escudero, D. Martínez, and G. Rigau. *Machine Learning Methods for WSD.* chapter of the book *Word Sense Disambiguation: Algorithms, Applications, and Trends.* E. Agirre and P. Edmonds, editors. Kluwer, 2006. Explained in section 3.5.

Cross-corpora evaluation and adaptation to new domains:

G. Escudero, L. Màrquez, and G. Rigau. A Comparison between Supervised Learning Algorithms for Word Sense Disambiguation. In *Proceedings of the 4th Computational Natural Language Learning Workshop, CoNLL*, 2000. Explained in chapter 4.

G. Escudero, L. Màrquez, and G. Rigau. An Empirical Study of the Domain Dependence of Supervised Word Sense Disambiguation Systems. In *Proceedings of the joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora, EMNLP/VLC*, 2000. Explained in chapter 4.

Data scarceness: bootstrapping:

G. Escudero and L. Màrquez. *Transductive Approach using Support Vector Machines on Labelled and Unlabelled Data*. Deliverable WP6.5 of the MEANING Project (IST-2001-34460), 2003. Explained in section 5.1.

International evaluation exercises:

G. Escudero, L. Màrquez, and G. Rigau. Using Lazy Boosting for Word Sense Disambiguation. In *Proceedings of the 2nd International Workshop an Evaluating Word Sense Disambiguation Systems, Senseval-2*, 2001. Explained in section 6.2.

G. Escudero, L. Màrquez, and G. Rigau. TALP System for the English Lexical Sample Task. In *Proceedings of the 3rd International Workshop an Evaluating Word Sense Disambiguation Systems, Senseval-3*, 2004. Explained in section 6.4.

Lexical/semantic resources for NLP:

L. Villarejo, J. Atserias, G. Escudero and G. Rigau. First Release of the Multilingual Central repository of MEANING. In *Proceedings of the SEPLN*, 2003.

L. Benítez, S. Cervell, G. Escudero, M. López, G. Rigau and M. Taulé. Methods and tools for building the Catalan WordNet. In *LREC Workshop: Language Resources for European Minority Languages*, 1998.

L. Benítez, S. Cervell, G. Escudero, M. López, G. Rigau and M. Taulé. Web WordNet Interface (Un sistema para el desarrollo de WordNets a través de la Web). In *Proceedings of the SEPLN*, 1998.

7.4 Further Work

This section is devoted to explain the current work and further research lines that we plan to address:

- First, the representation of the examples. Which is the best way to represent the examples? Which is the best feature selection procedure?

- The second research line is the application of bootstrapping techniques to Word Sense Disambiguation in order to increase, in a simple way, the amount of training examples; alleviating the data scarceness problem. We also are interested in *on-line* learning classifiers able to adapt to new domains.
- The third line is the integration of Semantic Parsing models within Word Sense Disambiguation classifiers in order to develop a better and complementary hybrid system to deal with both tasks.
- And finally, we want to continue participating in next editions of Senseval competitions, in both Lexical Sample and All-word tasks, for English, Catalan and Spanish languages.

Bibliography

S. Abney. Bootstrapping. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics, ACL*, 2002.

S. Abney. Understanding the Yarowsky Algorithm. *Computational Linguistics*, 30(3), 2004.

S. Abney, R. E. Schapire, and Y. Singer. Boosting Applied to Tagging and PP–attachment. In *Proceedings of the SIGDAT Conference on Empirical Methods and Very Large Corpora, EMNLP/VLC*, 1999.

E. Agirre, O. Lopez de Lacalle Lekuona, and D. Martínez. Exploring feature spaces with SVD and unlabeled data for Word Sense Disambiguation. In *Proceedings of the International Conference: Recent Advances in Natural Language Processing, RANLP*, 2005.

E. Agirre and P. Edmonds, editors. *Word Sense Disambiguation: Algoristhms, Applications, and Trends*. Kluwer, 2006.

E. Agirre and D. Martínez. Exploring Automatic Word Sense Disambiguation with Decision Lists and the Web. In *Proceedings of the COLING workshop: Semantic Annotation and Intelligent Annotation*, 2000.

E. Agirre and D. Martínez. Knowledge Sources for WSD. In *Proceedings of the International TSD Conference*, 2001.

E. Agirre and D. Martínez. Smoothing and Word Sense Disambiguation. In *Proceedings of España for Natural Language Processing (EsTAL)*, 2004a.

E. Agirre and D. Martínez. The Basque Country University system: English and Basque tasks. In *Proceedings of the International Workshop on the Evaluation of Systems for the Semantic Analysis of Text, Senseval–3*, 2004b.

E. Agirre and D. Martínez. Unsupervised WSD based on automatically retrieved examples: The importance of bias. In *Proceedings of the International Conference on Empirical Methods in Natural Language Processing, EMNLP*, 2004c.

E. Agirre and G. Rigau. A Proposal for Word Sense Disambiguation using Conceptual Distance. In *Proceedings of the International Conference: Recent Advances in Natural Language Processing, RANLP*, 1995.

E. Agirre and G. Rigau. Word Sense Disambiguation Using Conceptual Density. In *Proceedings of the International Conference on Computational Linguistics, COLING*, 1996.

H. Alshawi and D. Carter. Training and scaling preference functions for disambiguation. *Computational Linguistics*, 20(4), 1994.

R. K. Ando and T. Zhang. A High–Performance Semi–Supervised Learning Method for Text Chunking. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics, ACL*, 2005.

S. Argamon-Engelson and I. Dagan. Committee–based Sample Selection for Probabilistic Classifiers. *Journal of Artificial Intelligence Research*, 11, 1999.

J. Atserias, B. Casas, E. Comelles, M. González, L. Padró, and M. Padró. FreeLing 1.3: Syntactic and semantic services in an open-source NLP library. In *Proceedings of the 5th International Conference on Language Resources and Evaluation, LREC*, 2006.

J. Atserias, S. Climent, J. Farreras, G. Rigau, and H. Rodríguez. Combining Multiple Methods for the Automatic Construction of Multilingual WordNets. In *Proceedings of Conference on Recent Advances in Natural Language Processing, RANLP*, 1997.

J. Atserias, L. Villarejo, G. Rigau, E. Agirre, J. Carroll, B. Magnini, and P. Vossen. The MEANING Multilingual Central Repository. In *Proceedings of the International WordNet Conference*, 2004.

Y. Bar-Hillel. The present status of automatic translation of languages. In F. L. Alt, editor, *Advances in Computers*. Academic Press, 1960.

E. Bauer and R. Kohavi. An Empirical Comparison of Voting Classification Algorithms: Bagging, Boosting and Variants. *Machine Learning Journal. Special issue on IMLM for Improving and Scaling Machine Learning Algorithms*, 1999.

A. Berger, S. Della Pietra, and V. Della Pietra. A Maximum Entropy Approach to Natural Language Processing. *Computational Linguistics*, 22(1), 1996.

A. Blum and T. Mitchell. Combining labeled and unlabeled data with co–training. In *Proceedings of the Workshop on Computational Learning Theory, COLT*, 1998.

B. Boser, I. Guyon, and V. Vapnik. A Training Algorithm for Optimal Margin Classifiers. In *Proceedings of the Workshop on Computational Learning Theory, COLT* , 1992.

T. Brants. Internal and external tagsets in part-of-speech tagging. In *Proceedings of EuroSpeech*, 1997.

L. Breiman. Arcing Classifers. *The Annals of Statistics*, 26(1), 1998.

E. Brill. Transformation–based Error–Driven Learning and Natural Language Processing: A Case Study in Part–of–Speech Tagging. *Computational Linguistics*, 21(4), 1995.

E. Briscoe and J. Carroll. Automatic extraction of subcategorization from corpora. In *Proceedings of the ACL Conference on Applied Natural Language Processing*, 1997.

E. J. Briscoe. Lexical issues in natural language processing. In E. H. Klein and F. Veltman, editors, *Natural Language and Speech Proceedings of the Symposium on Natural Language and Speech*. Springer–Verlag, Berlin, 1991.

R. F. Bruce and J. M. Wiebe. Decomposable Modeling in Natural Language Processing. *Computational Linguistics*, 25(2), 1999.

C. Cabezas, P. Resnik, and J. Stevens. Supervised Sense Tagging using Support Vector Machines. In *Proceedings of the International Workshop on Evaluating Word Sense Disambiguation Systems*, 2001.

C. Cardie and R. J. Mooney. Guest Editors' Introduction: Machine Learning and Natural Language. *Machine Learning (Special Issue on Natural Language Learning)*, 34, 1999.

J. Carletta. Assessing Agreement on Classification Tasks: the Kappa Statistic. *Computational Linguistics*, 22(2), 1996.

M. Carpuat, W. Su, and D. Wu. Augmenting Ensemble Classification for Word Sense Disambiguation with a Kernel PCA Model. In *Proceedings of the International Workshop on the Evaluation of Systems for the Semantic Analysis of Text, Senseval–3*, 2004.

M. Carpuat and D. Wu. Word Sense Disambiguation vs. Statistical Machine Translation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics, ACL*, 2005.

M. Castillo, F. Real, J. Atserias, and G. Rigau. The TALP Systems for Disambiguating WordNet Glosses. In *Proceedings of the International Workshop on the Evaluation of Systems for the Semantic Analysis of Text, Senseval–3*, 2004.

S. F. Chen. *Building Probabilistic Models for Natural Language*. PhD thesis, Center for Research in Computing Technology. Harvard University, 1996.

T. Chklovski and R. Mihalcea. Building a Sense Tagged Corpus with Open Mind Word Expert. In *Proceedings of the ACL workshop: Word Sense Disambiguation: Recent Successes and Future Directions*, 2002.

J. Cohen. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20, 1960.

M. Collins. Parameter Estimation for Statistical Parsing Models: Theory and Practice of Distribution–Free Methods. In *To appear as a book chapter*, 2002. Available at the web address: www.ai.mit.edu/people/mcollins.

M. Collins and Y. Singer. Unsupervised models for named entity classification. In *Proceedings of the joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora, EMNLP/VLC*, 1999.

C. Connine. Effects of sentence context and lexical knowledge in speech processing. In G. T. Altmann, editor, *Cognitive models in speech processing*. MIT Press, 1990.

S. Cost and S. Salzberg. A Weighted Nearest Neighbor Algorithm for Learning with Symbolic Features. *Machine Learning*, 10, 1993.

J. Cowie, J. Guthrie, and L. Guthrie. Lexical disambiguation using simulated annealing. In *Proceedings of the International Conference on Computational Linguistics, COLING*, 1992.

E. Crestan, M. El-Bèze, and C. de Loupy. Improving WSD with Multi–Level View of Context Monitored by Similarity Measure. In *Proceedings of the International Workshop on evaluating Word Sense Disambiguation Systems*, 2001.

N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines*. Cambridge University Press, 2000.

W. Daelemans and V. Hoste. Evaluation of Machine Learning Methods for Natural Language Processing Tasks. In *Proceedings of the International Conference on Language Resources and Evaluation, LREC*, 2002.

W. Daelemans, A. van den Bosch, and J. Zavrel. Forgetting Exceptions is Harmful in Language Learning. *Machine Learning (Special Issue on Natural Language Learning)*, 34, 1999.

W. Daelemans, J. Zavrel, K. van der Sloot, and A. van den Bosch. *Timbl: Tilburg memory based learner, version 4.0, reference guide.* Technical Report, University of Antwerp, 2001.

I. Dagan, Y. Karov, and D. Roth. Mistake–Driven Learning in Text Categorization. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing, EMNLP*, 1997.

S. Dasgupta, M. Littman, and D. McAllester. PAC generalization bounds for co–training. In *Advances in Neural Infomation Processing Systems, NIPS*, 2001.

J. Daudé, L. Padró, and G. Rigau. Mapping Multilingual Hierarchies using Relaxation Labelling. In *Proceedings of the Joint SIGDAT Conference on Empirical Methods in natural Language Processing and Very Large Corpora, EMNLP/VLC*, 1999.

J. Daudé, L. Padró, and G. Rigau. Magpping WordNets using Structural Information. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics, ACL*, 2000.

J. Daudé, L. Padró, and G. Rigau. A complete WN1.5 to WN1.6 Mapping. In *Proceedings of the NAACL Workshop: WordNet and Other Lexical Resources: Applications, Extensions and Customizations*, 2001.

R. Lopez de Mántaras. A Distance–Based Attribute Selection Measure for Decission Tree Induction. *Machine Learning*, 6(1), 1991.

B. Decadt, V. Hoste, W. Daelemans, and A. van den Bosch. GAMBL, Genetic Algorithm Optimization of Memory–Based WSD. In *Proceedings of the International Workshop an Evaluating Word Sense Disambiguation Systems, Senseval–3*, 2004.

T. G. Dietterich. Machine Learning Research: Four Current Directions. *AI Magazine*, 18 (4), 1997.

T. G. Dietterich. Aproximate Statistical Tests for Comparing Supervised Classification Learning Algorithms. *Neural Computation*, 10(1), 1998.

T. G. Dietterich. An Experimental Comparison of Three Methods for Constructing Ensembles of Decision Trees: Bagging, Boosting, and Randomization. *Machine Learning*, 40(2), 2000.

R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*. John Wiley & Sons, 1973.

P. Edmonds and S. Cotton. Senseval–2: Overview. In *Proceedings of the International Workshop on Evaluating Word Sense Disambiguation Systems*, 2001.

S. Engelson and I. Dagan. Minimizing Manual Annotation Cost in Supervised Training from Corpora. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics, ACL*, 1996.

G. Escudero and L. Màrquez. Transductive Approach using Support Vector Machines on Labeled and Unlabeled Data. Technical report, Deliverable WP6.5 of the MEANING Project (IST–2001–34460), 2003.

G. Escudero, L. Màrquez, and G. Rigau. A Comparison between Supervised Learning Algorithms for Word Sense Disambiguation. In *Proceedings of the Computational Natural Language Learning Workshop, CoNLL*, 2000a.

G. Escudero, L. Màrquez, and G. Rigau. An Empirical Study of the Domain Dependence of Supervised Word Sense Disambiguation Systems. In *Proceedings of the joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora, EMNLP/VLC*, 2000b.

G. Escudero, L. Màrquez, and G. Rigau. Boosting Applied to Word Sense Disambiguation. In *Proceedings of the European Conference on Machine Learning, ECML*, 2000c.

G. Escudero, L. Màrquez, and G. Rigau. Naive Bayes and Exemplar–Based Approaches to Word Sense Disambiguation Revisited. In *Proceedings of the European Conference on Artificial Intelligence, ECAI*, 2000d.

G. Escudero, L. Màrquez, and G. Rigau. Using Lazy Boosting for Word Sense Disambiguation. In *Proceedings of the International Workshop an Evaluating Word Sense Disambiguation Systems, Senseval–2*, 2001.

G. Escudero, L. Màrquez, and G. Rigau. TALP System for the English Lexical Sample Task. In *Proceedings of the International Workshop an Evaluating Word Sense Disambiguation Systems, Senseval–3*, 2004.

C. Fellbaum, editor. *WordNet. An Electronic Lexical Database*. The MIT Press, 1998.

R. Florian, S. Cucerzam, C. Schafer, and D. Yarowsky. Combining Classifiers for Word Sense Disambiguation. *Natural Language Engineering, Special Issue on Evaluating Word Sense Disambiguation Systems*, 8(4), 2003.

R. Florian and D. Yarowsky. Modelling consensus: Classifier combination for word sense disambiguation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing, EMNLP*, 2002.

W. N. Francis and H. Kucera. *Frequency Analysis of English Usage: Lexicon and Grammar*. Houghton Mifflin Company, 1982.

Y. Freund and R. E. Schapire. Experiments with a New Boosting Algorithm. In *Proceedings of the 13th International Conference on Machine Learning, ICML*, 1996.

Y. Freund and R. E. Schapire. A Decision–theoric Generalization of On–line Learning and an Application to Boosting. *Journal of Computer and System Sciences*, 55(1), 1997.

A. Fujii, K. Inui, T. Tokunaga, and H. Tanaka. Selective Sampling for Example-Based Word Sense Disambiguation. *Computational Linguistics*, 24(4), 1998.

W. Gale, K. W. Church, and D. Yarowsky. Estimating Upper and Lower Bounds on the Performance of Word Sense Disambiguation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics, ACL*, 1992a.

W. Gale, K. W. Church, and D. Yarowsky. One Sense Per Discourse. In *Proceedings of the DARPA Speech and Natural Language Workshop*, 1992b.

W. Gale, K. W. Church, and D. Yarowsky. A Method for Disambiguating Word Senses in a Large Corpus. *Computers and the Humanities*, 26(5), 1993.

G. Grefenstette. Evaluation Techniques for Automatic Semantic Extraction: Comparing Syntatic and Window–based Approaches. In *Proceedings of the ACL SIGLEX Workshop on Acquisition of Lexical Knowledge from Text*, 1993.

C. Grozea. Finding optimal parameter settings for high performance word sense disambiguation. In *Proceedings of the International Workshop on the Evaluation of Systems for the Semantic Analysis of Text, Senseval–3*, 2004.

J. Guthrie, L. Guthrie, Y. Wilks, and H. Aidinejad. Subject–dependent Cooccurrence and Word Sense Disambiguation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics, ACL*, 1991.

S. Har-Peled, D. Roth, and D. Zimak. Constraint Classification for Multiclass Classification and Ranking. In *Proceedings of the Workshop on Neural Information Processing Systems*, 2002.

S. M. Harabagiu and D. I. Moldovan. Knowledge Processing on an Extended WordNet. In C. Fellbaum, editor, *WordNet: An Electronic Lexical Database and some of its Applications*. MIT Press, 1998.

V. Hoste, W. Daelemans, I. Hendrickx, and A. van den Bosch. Evaluating the Results of a Memory–Based Word–Expert Approach to Unrestricted Word Sense Disambiguation. In *Proceedings of the Workshop on Word Sense Disambiguation: Recent Successes and Future Directions*, 2002a.

V. Hoste, I. Hendrickx, W. Daelemans, and A. van den Bosch. Parameter Optimization for Machine–Learning of Word Sense Disambiguation. *Natural Language Engineering*, 8(4), 2002b.

V. Hoste, A. Kool, and W. Daelemans. Classifier Optimization and Combination in the English All Words Task. In *Proceedings of the International Wordshop on Evaluating Word Sense Disambiguation Systems*, 2001.

N. Ide and J. Véronis. Introduction to the Special Issue on Word Sense Disambiguation: The State of the Art. *Computational Linguistics*, 24(1), 1998.

H. T. Ilhan, S. D. Kamvar, D. Klein, C. D. Manning, and K. Toutanova. Combining Heterogeneous Classifiers for Word–Sense Disambiguation. In *Proceedings of the International Workshop on evaluating Word Sense Disambiguation Systems*, 2001.

T. Joachims. Text Categorization with Support Vector Machines: Learning with Many Relevant Features. In *Proceedings of the European Conference on Machine Learning, ECML*, 1998.

T. Joachims. Transductive Inference for Text Classification using Support Vector Machines. In *Proceedings of the Internacional Conference on Machine Learning, ICML*, 1999.

T. Joachims. *Learning to Classify Text Using Support Vector Machines*. Kluwer, 2002.

A. Kaplan. An Experimental Study of Ambiguity and Context. *Mechanical Translation*, 2(2), 1950.

A. Kilgarriff. What is Word Sense Disambiguation Good For? In *Proceedings of NLP Pacific Rim Symposium*, 1997.

A. Kilgarriff. Gold Standard Datasets for Evaluating Word Sense Disambiguation Programs. *Computer Speech and Language, Special Issue on Evaluation*, 12(4), 1998a.

A. Kilgarriff. SENSEVAL: An Exercise in Evaluating Word Sense Disambiguation Programs. In *Proceedings of the 1st Conference on Language Resources and Evaluation, LREC*, 1998b.

A. Kilgarriff. English Lexical Sample Task Description. In *Proceedings of the International Workshop on Evaluating Word Sense Disambiguation Systems, Senseval–2*, 2001.

A. Kilgarriff and J. Rosenzweig. English SENSEVAL: Report and Results. In *Proceedings of the International Conference on Language Resources and Evaluation, LREC*, 2000.

U. S. Kohomban and W. S. Lee. Learning semantic classes for word sense disambiguation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics, ACL*, 2005.

R. Krovetz and W. B. Croft. Lexical Ambiguity and Information Retrieval. *ACM Transactions on Information Systems*, 10(2), 1992.

T. Kudo and Y. Matsumoto. Chunking with Support Vector Machines. In *Proceedings of the Meeting of the North American Chapter of the Association for Computational Linguistics, NAACL*, 2001.

C. Leacock, M. Chodorow, and G. A. Miller. Using Corpus Statistics and WordNet Relations for Sense Identification. *Computational Linguistics*, 24(1), 1998.

C. Leacock, G. Towell, and E. Voorhes. Towards Building Contextual Representations of Word Sense Using Statistical Models. In *Proceedings of SIGLEX Workshop on Acquisition of Lexical Knowledge from Text*, 1993.

Y. K. Lee and H. T. Ng. An Empirical Evaluation of Knowledge Sources and Learning Algorithms for Word Sense Disambiguation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing, EMNLP*, 2002.

Y. K. Lee, H. T. Ng, and T. K. Chia. Supervised Word Sense Disambiguation with Support Vector Machines and Multiple Knowledge Sources. In *Proceedings of the International Workshop on the Evaluation of Systems for the Semantic Analysis of Text, Senseval–3*, 2004.

D. B. Lenat and V. G. Ramanathan, editors. *Building Large Knowledge–based Systems.* Addison–Wesley, Reading, MA, US, 1990.

M. Lesk. Automated sense disambiguation using machine–readable dictionaries: How to tell a pine cone from an ice cream cone. In *Proceedings of the SIGDOC Conference*, 1986.

D. Lewis and W. Gale. Training Text Classifiers by Uncertainty Sampling. In *Proceedings of the International ACM Conference on Research and Development in Information Retrieval*, 1994.

D. Lin. Dependency-based Evaluation of MiniPar. In *Proceedings of Workshop on hte Evaluation of Parsing Systems*, 1998.

B. Magnini and G. Cavaglia. Integrating Subject Field Codes into WordNet. In *Proceedings of the International Conference on Language Resources and Evaluation, LREC*, 2000.

B. Magnini, C. Strapparava, G. Pezzulo, and A. Gliozzo. Using Domain Information for Word Sense Disambiguation. In *Proceedings of the International Workshop on Evaluating Word Sense Disambiguation Systems, Senseval–2*, 2001.

C. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing*. The MIT Press, 1999.

L. Màrquez. *Part–of–Speech Tagging: A Machine–Learning Approach based on Decision Trees*. PhD thesis, Departament de Llenguatges i Sistemes Informàtics. Universitat Politècnica de Catalunya, 1999.

L.. Màrquez, G. Escudero, D. Martínez, and G. Rigau. *Word Sense Disambiguation: Algorithms, Applications and Trends*, chapter Machine Learning Methods. Kluwer, 2006.

L. Màrquez, M. Taulé, A. Martí, N. Artigas, M. García, F. Real, and D. Férres. Senseval–3: The Spanish lexical sample task. In *Proceedings of the International Workshop on the Evaluation of Systems for the Semantic Analysis of Text, Senseval–3*, 2004a.

L. Màrquez, M. Taulé, A. Martí, M. García, F. Real, and D. Férres. Senseval–3: The Spanish lexical sample task. In *Proceedings of the International Workshop on the Evaluation of Systems for the Semantic Analysis of Text, Senseval–3*, 2004b.

D. Martínez. *Supervised Word Sense Disambiguation: Facing Current Challenges*. PhD thesis, Lengoaia eta Sistema Informatikoak Saila. Euskal Herriko Univertsitatea, 2004.

D. Martínez and E. Agirre. One Sense per Collocation and Genre/Topic Variations. In *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora, EMNLP/VLC*, 2000.

D. Martínez, E. Agirre, and L. Màrquez. Syntactic Features for High Precision Word Sense Disambiguation. In *Proceedings of International Conference on Computational Linguistics, COLING*, 2002.

R. Mihalcea. Bootstrapping Large Sense Tagged Corpora. In *Proceedings of the International Conference on Languages Resources and Evaluation, LREC*, 2002a.

R. Mihalcea. Instance Based Learning with Automatic Feature Selection Applied to Word Sense Disambiguation. In *Proceedings of the International Conference on Computational Linguistics, COLING*, 2002b.

R. Mihalcea. Co–training and Self–training for Word Sense Disambiguation. In *Proceedings of the Conference on Natural Language Learning, CoNLL*, 2004.

R. Mihalcea, T. Chklovski, and A. Kilgarriff. The Senseval–3 English Lexical Sample Task. In *Proceedings of the International Workshop on the Evaluation of Systems for the Semantic Analysis of Text, Senseval–3*, 2004.

R. Mihalcea and E. Faruque. SenseLearner: Minimally Supervised Word Sense Disambiguation for All Words in Open Text. In *Proceedings of the International Workshop on the Evaluation of Systems for the Semantic Analysis of Text, Senseval–3*, 2004.

R. Mihalcea and D. Moldovan. Pattern Learning and Active Feature Selection for Word Sense Disambiguation. In *Proceedings of the International Workshop on Evaluating Word Sense Disambiguation Systems, Senseval–2*, 2001.

R. Mihalcea and I. Moldovan. A Method for Word Sense Disambiguation of Unrestricted Text. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics, ACL*, 1999a.

R. Mihalcea and I. Moldovan. An Automatic Method for Generating Sense Tagged Corpora. In *Proceedings of the Conference of the American Association for Artificial Intelligence, AAAI*, 1999b.

G. A. Miller, R. Beckwith, C. Fellbaum, D. Gross, and K. Miller. Five Papers on WordNet. *Special Issue of International Journal of Lexicography*, 3(4), 1990.

G. A. Miller, M. Chodorow, S. Landes, C. Leacock, and R. G. Thomas. Using a semantic concordance for sense identification. In *Proceedings of the ARPA Human Language Technology Workshop*, 1994.

G. A. Miller, C. Leacock, R. Tengi, and R. T. Bunker. A Semantic Concordance. In *Proceedings of the ARPA Workshop on Human Language Technology*, 1993.

T. M. Mitchell. *Machine Learning.* McGraw Hill, 1997.

R. J. Mooney. Comparative Experiments on Disambiguating Word Senses: An Illustration of the Role of Bias in Machine Learning. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing, EMNLP*, 1996.

M. Murata, M. Utiyama, K. Uchimoto, Q. Ma, and H. Isahara. Japanese Word Sense Disambiguation Using the Simple Bayes and Support Vector Machine Methods. In *Proceedings of the International Workshop an Evaluating Word Sense Disambiguation Systems, Senseval–2*, 2001.

H. T. Ng. Exemplar-Base Word Sense Disambiguation: Some Recent Improvements. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing, EMNLP*, 1997a.

H. T. Ng. Getting Serious about Word Sense Disambiguation. In *Proceedings of the ACL SIGLEX Workshop: Standardizing Lexical Resources*, 1997b.

H. T. Ng and H. B. Lee. Integrating Multiple Knowledge Sources to Disambiguate Word Sense: An Exemplar–based Approach. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics, ACL*, 1996.

H. T. Ng, C. Y. Lim, and S. K. Foo. A Case Study on Inter–Annotator Agreement for Word Sense Disambiguation. In *Proceedings of the ACL SIGLEX Workshop: Standardizing Lexical Resources*, 1999.

G. Ngai and R. Florian. Transformation–based Learning in the Fast Lane. In *Proceedings of the Meeting of the North American Chapter of the Association for Computational Linguistics, NAACL*, 2001.

I. Niles and A. Pease. Towards a Standard Upper Ontology. In *Proceedings of the 2nd International Conference on Formal Ontology in Information Systems, FOILS'2001*, 2001.

T. Pedersen. A Decision Tree of Bigrams is an Accurate Predictor of Word Sense. In *Proceedings of the North American Chapter of the Association for Computational Linguistics, NAACL*, 2001.

T. Pedersen and R. Bruce. A New Supervised Learning Algorithm for Word Sense Disambiguation. In *Proceedings of the National Conference on Artificial Intelligence, AAAI*, 1997.

T. Pedersen and R. Bruce. Knowledge Lean Word–Sense Disambiguation. In *Proceedings of the Conference of the American Association for Artificial Intelligence, AAAI*, 1998.

M. Popescu. Regularized Least–Squares Classification for Word Sense Disambiguation. In *Proceedings of the International Workshop on the Evaluation of Systems for the Semantic Analysis of Text, Senseval–3*, 2004.

P. Procter. *Longman's Dictionary of Contemporary English.* Longmans, Harlow, 1973.

J. R. Quinlan. *C4.5 : Programs for Machine Learning.* Morgan Kaufman, 1993.

J. R. Quinlan. Bagging, Boosting and C4.5. In *Proceedings of the 13th American Conference on Artificial Intelligence, AAAI*, 1996.

P. Resnik. Disambiguating noun groupings with respect to WordNet. In *Proceedings of the ACL Workshop on Very Large Corpora, VLC'95*, 1995.

P. Resnik and D. Yarowsky. A Perspective on Word Sense Disambiguation Methods and their Evaluation. In *Proceedings of the SIGLEX Workshop: Tagging Text with Lexical Semantics: Why, What and How?*, 1997.

F. Ribas. *On Acquiring Appropriate Selectional Restrictions from Corpora Using a Semantic Taxonomy.* PhD thesis, Departament de Llenguatges i Sistemes Informàtics. Universitat Politècnica de Catalunya, 1995.

S. Richardson. *Determining Similarity and Inferring Relations in a Lexical Knowledge Base.* PhD thesis, The City University of New York, New York, 1997.

G. Rigau. *Automatic Acquisition of Lexical Knowledge from MRDs.* PhD thesis, Departament de Llenguatges i Sistemes Informàtics. Universitat Politècnica de Catalunya, 1998.

G. Rigau, J. Atserias, and E. Agirre. Combining Unsupervised Lexical Knowledge Methods for Word Sense Disambiguation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics, Joint ACL/EACL*, 1997.

R. L. Rivest. Learning Decision Lists. *Machine Learning*, 2, 1987.

G. Salton. *Automatic Information Retrieval.* McGraw Hill, 1968.

G. Salton and M. McGill. *Introduction to Modern Information Retrieval.* McGraw Hill, 1983.

K. Samuel. Lazy Transformation–Based Learning. In *Proceedings of the 11th International Florida AI Research Symposium Conference*, 1998.

R. E. Schapire. The Boosting Approach to Machine Learning: An Overview. In *MSRI Workshop on Nonlinear Estimation and Classification*, 2002.

R. E. Schapire and Y. Singer. Improved Boosting Algorithms Using Confidence–rated Predictions. *Machine Learning*, 37(3), 1999.

R. E. Schapire and Y. Singer. Boostexter: A Boosting–based System for Text Categorization. *Machine Learning*, 39(2/3), 2000.

H. Schütze. Dimensions of Meaning. In *Proceedings of Supercomputing*, 1992.

H. Schütze and J. Pedersen. Information retrieval based on word senses. In *Proceedings of SDAIR*, 1995.

S. Sekine. The Domain Dependence of Parsing. In *Proceedings of the 5th of Applied Natural Language Processing, ANLP*, 1997.

S. Seneff. TINA, A natural language system for spoken language applications. *Computational Linguistics*, 18(1), 1992.

H. Seo, S. Lee, H. Rim, and H. Lee. KUNLP system using Classification Information Model at Senseval–2. In *Proceedings of the International Workshop on evaluating Word Sense Disambiguation Systems, Senseval–2*, 2001.

B. Snyder and M. Palmer. The English All–Words Task. In *Proceedings of the International Workshop on the Evaluation of Systems for the Semantic Analysis of Text, Senseval–3*, 2004.

R. Sproat, J. Hirschberg, and D. Yarowsky. A corpus–based synthesizer. In *Proceedings of the International Conference on Spoken Language Processing*, 1992.

M. Stevenson and Y. Wilks. The Interaction of Knowledge Sources in Word Sense Disambiguation. *Computational Linguistics*, 27(3):321–349, 2001.

C. Strapparava, A. Gliozzo, and C. Giuliano. Pattern Abstraction and Term Similarity for Word Sense Disambiguation: IRST at Senseval–3. In *Proceedings of the International Workshop on the Evaluation of Systems for the Semantic Analysis of Text, Senseval–3*, 2004.

A. Suárez. *Resolución de la ambigüedad semántica de las palabras mediante modelos de probabilidad de máxima entropía.* PhD thesis, Departamento de Lenguajes y Sistemas Informáticos. Universidad de Alicante, 2004.

A. Suárez and M. Palomar. A Maximum Entropy–based Word Sense Disambiguation System. In *Proceedings of the International Conference on computational Linguistics, COLING*, 2002.

M. Sussna. Word sense disambiguation for gree–text indexing using a massive semantic network. In *Proceedings of the International Conference on Information and Knowledge Base Management, CIKM*, 1993.

G. Towell and E. M. Voorhees. Disambiguating Highly Ambiguous Words. *Computational Linguistics*, 24(1), 1998.

V. N. Vapnik. *Statistical Learning Theory.* John Wiley, 1998.

S. Vázquez, R. Romero, A. Suárez, A. Montoyo, M. García, M. T. Martín, M. Á. García, and L. A. Urena. The R2D2 team at Senseval–3. In *Proceedings of the International Workshop on the Evaluation of Systems for the Semantic Analysis of Text, Senseval–3*, 2004.

J. Véronis. A study of polysemy judgements and inter–annotator agreement. In *Programme and advanced papers of the Senseval workshop*, 1998.

E. Viegas, K. Mahesh, S. Nirenburg, and S. Beale. Semantics in Action. In P. Saint-Dizier, editor, *Predicative Forms in Natural Language and in Lexical Knowledge Bases.* Kluwer Academic Press, 1999.

L. Villarejo, L. Màrquez, E. Agirre, D. Martínez, B. Magnini, C. Strapparava, D. McCarthy, A. Montoyo, and A. Suárez. The "Meaning" system on the English all–words task. In *Proceedings of the International Workshop on the Evaluation of Systems for the Semantic Analysis of Text, Senseval–3*, 2004.

E. Voorhees. Using WordNet to disambiguate word senses for text retrieval. In *Proceedings of the Annual International ACM SIGDIR Conference on Research and Development in Information Retrieval*, 1993.

P. Vossen, editor. *EuroWordNet. A Multilingual Database with Lexical Semantic Networks.* Kluwer Academic Publishers, 1998.

W. Weaver. Translation. *Machine Translation of Languages*, 1955.

Y. Wilks, D. Fass, C. Guo, J. McDonald, T. Plate, and B. Slator. Providing Machine Tractable Dictionary Tools. In J. Pustejowsky, editor, *Semantics and the Lexicon.* Dordrecht, Kluwer Academic Publishers, 1993.

Y. Wilks and M. Stevenson. The Grammar of Sense: Is word–sense tagging much more than part–of–speech tagging? Technical Report CS-96-05, University of Sheffield, Sheffield, UK, 1996.

T. Winograd. *Understanding Natural Language.* Academic Press, 1972.

D. Yarowsky. Word sense disambiguation using statistical models of Roget's categories trained on large corpora. In *Proceedings of the International Conference on Computational Linguistics, COLING*, 1992.

D. Yarowsky. Decision Lists for Lexical Ambiguity Resolution: Application to Accent Restoration in Spanish and French. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics, ACL*, 1994.

D. Yarowsky. *Three Machine Learning Algorithms for Lexical Ambiguity Resolution*. PhD thesis, Department of Computer and Information Sciences. University of Pennsylvania, 1995a.

D. Yarowsky. Unsupervised Word Sense Disambiguation Rivaling Supervised Methods. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics, ACL*, 1995b.

D. Yarowsky. Homograph Disambiguation in Text–to–speech Synthesys. In J. P. Olive J. T. H. van Santen, R. Sproat and J. Hirschberg, editors, *Progress in Speech Synthesis*. Springer–Verlag, 1997.

D. Yarowsky. Hierarchical Decision Lists for Word Sense Disambiguation. *Computers and the Humanities*, 34(2), 2000.

D. Yarowsky, S. Cucerzan, R. Florian, C. Schafer, and R. Wicentowski. The Johns Hopkings Senseval–2 System Descriptions. In *Proceedings of the International Workshop on evaluating Word Sense Disambiguation Systems, Senseval–2*, 2001.

V. Yngve. Syntax and the Problem of Multiple Meaning. *Machine Translation of Languages*, 1955.

D. Yuret. Some Experiments with a Naive Bayes WSD System. In *Proceedings of the International Workshop an Evaluating Word Sense Disambiguation Systems, Senseval–3*, 2004.

Appendix A

Statistical Measures

This appendix contains the description and formulae of the statistics and measures used in this work.

A.1 Evaluation Measures

All measures of this section have been applied for comparison purposes within different classifiers on different data.

Given the predictions of a classifier F, where N denotes then total number of examples, V the number of examples with prediction and A the number of examples with a correct prediction:

$$precision = \frac{A}{V}$$

$$recall = \frac{A}{N}$$

$$coverage = \frac{V}{N}$$

$$F1 = \frac{2 \times precision \times recall}{precision + recall} = \frac{2 \times A}{V + N}$$

When $V = N$ (when a classifier gives an output prediction for each example and not abstains) we talk about **accuracy** instead of precision and recall, because they are the same.

A.2 Significance Tests

We have applied two significance tests for complementing the measures of the previous section. Given two classifiers F and G:

- the **paired Student's t-test** with a confidence value of $t_{9,0.975} = 2.262$ or equivalent (depending on the number of folds) have been used; when using a cross-validation framework:
$$\frac{\overline{p} \times \sqrt{n}}{\sqrt{\frac{\sum_{i=1}^{n}(p(i)-\overline{p})^2}{n-1}}} > 2.262$$
where n is the number of folds, $\overline{p} = \frac{1}{n}\sum_{i=1}^{n} p(i)$, $p(i) = p_F(i) - p_G(i)$, and $p_X(i)$ is the number of examples misclassified by classifier X (where $X \in \{F, G\}$).
- and, the **McNemar's test** with a confidence value of $\chi^2_{1,0.95} = 3.842$, when no cross-validation framework has been applied:
$$\frac{(|A-B|-1)^2}{A+B} > 3.842$$
where A is the number of examples misclassified by classifier F but no by G, and B is the number of examples misclassified by G but not by F.

A.3 Agreement and Kappa statistic

Given the predictions of two classifiers F and G. The **agreement** (observed) is computed as:
$$Pa = \frac{A}{N}$$
where A denotes then number of examples for which both classifiers have output the same prediction and N is the total number of examples.

And the **kappa** statistic as:
$$kappa = \frac{Pa - Pe}{1 - Pe}$$
where Pe (the *expected agreement*) is computed as:
$$Pe = \sum_{i=1}^{ns} \frac{P_F(i) \times P_G(i)}{N^2}$$
where ns is the number of senses, and $P_X(i)$ is the number of predictions of classifier X (with $X \in \{F, G\}$) on the sense i.

Appendix B

Example of the Feature Extractor

The following example has been extracted from the test set of the English Lexical Task corpus of Senseval-2. It is followed by some of the features of the output of our feature extractor component used in this document. The target word is *'Pop Art'*[1] and its sense is *'pop_art%1:06:00::'*:

```
What is clearly a dealers' market is often signalled
by the invention of a brand name to group together
a variety of material, perhaps rather disparate.
Pop  Art# is an example.
## pop_art%1:06:00:: art.40003
```

We have divided the features of the system in 4 categories: local, topical, knowledge-based and syntactic features. Table B.1 shows a short description of the **local features**. The basic aim of theses features is to model the information of the surroundings words of the target word. All this features are extracted from a ±3-word-window centred on the target word. The features also contain the position of all its components. To obtain Part-of-Speech and lemma for each word, we used FreeLing[2]. Most of these features have been doubled for lemma and word form. Next, some local features generated from the previous example are shown:

- **form (form):** form of the target word (e.g. `form Art`).
- **p_loc (locat)**: part-of-speech and position in a ±3-word-window (e.g. `p_loc +1 VBZ`).

[1]The target words is always in the last sentence of the example, and followed by the symbol '#'.
[2]`http://www.lsi.upc.es/~freeling`

feat.	description
form	form of the target word
locat	all part-of-speech/forms/lemmas in the local context
coll	all collocations of two part-of-speech/forms/lemmas
coll2	all collocations of a form/lemma and a part-of-speech (and the reverse)
first	form/lemma of the first noun/verb/adjective/adverb on the left/right of the target word

Table B.1: Local Feature Set

- **f_loc (locat)**: form and position in a ±3-word-window (i.e. `f_loc +1 is`).
- **l_loc (locat)**: lemma and position in a ±3-word-window (e.g. `l_loc +1 be`).
- **ppcol (coll)**: part-of-speech collocations of every two words in a ±3-word-window (e.g. `ppcol -1 NN +2 DT`).
- **ffcol (coll)**: form collocations of every two words (continuous or not) in a ±3-word-window (e.g. `ffcol -1 Pop +1 is`).
- **fpcol (coll2)**: form–part-of-speech collocations of every two words (continuous or not) in a ±3-word-window (e.g. `fpcol -1 Pop +1 VBZ`).
- **pfcol (coll2)**: part-of-speech–form collocations of every two words (continuous or not) in a ±3-word-window (e.g. `pfcol +1 VBZ +3 example`).
- **f_fln (first)**: the form of the first noun on the left of the target word (e.g. `f_fln Pop`).
- **l_frv (first)**: the lemma of the first verb on the right of the target word (e.g. `l_frv be`).

Three types of **topical features** are shown in table B.2. Topical features try to obtain non-local information from the words of the context. For each type, two overlapping sets of redundant topical features are considered: one extracted from a ±10-word-window and another considering all the example. Next, it is shown some topical features generated from previous example:

- **f_top (topic)**: bag of forms of the open-class-words in the example (e.g. `f_top market`).

feat.	description
topic	bag of forms/lemmas
sbig	all form/lemma bigrams of the example
comb	forms/lemmas of consecutive (or not) pairs of the open-class-words in the example

Table B.2: Topical Feature Set

- **l_top (topic)**: bag of lemmas of the open-class-words in the example (e.g. `l_top signal`).
- **fsbig (sbig)**: collocations of forms of every two consecutive words in the example (e.g. `fsbig of material`).
- **lcomb (comb)**: collocations of lemmas of every two open-class-words (consecutive or not) in the example (e.g. `lcomb signal invention`).

Table B.3 presents the **knowledge-based features**. These features have been obtained using the knowledge contained into Multilingual Central Repository (MCR) (Atserias et al., 2004) of MEANING Project. For each example, the program obtains, from each context, all nouns, all their synsets and their associated semantic information: Sumo labels, domain labels, WordNet Lexicographic Files, and EuroWordNet Top Ontology. We also assign to each label a weight which depend on the number of labels assigned to each noun and their relative frequencies in the whole WordNet (see section 6.2.1 for a more detailed description). For each kind of semantic knowledge, summing up all these weights, the program finally selects those semantic labels with higher weights. Next, it is shown some knowledge-based features generated from previous example:

- **f_mag (f_magn)**: the first domain label applying the formula previously explained in section 6.2.1 to the IRST Domains (e.g. `f_mag quality`).
- **a_mag (f_magn)**: bag of domain labels applying the formula previously explained in section 6.2.1 to the IRST Domains (e.g. `a_mag psychology`).
- **f_sum (f_sumo)**: the first SUMO label applying the formula previously explained in section 6.2.1 to the SUMO Labels (e.g. `f_sum capability`).
- **f_ton (f_tonto)**: the first EuroWordNet Top Ontology label applying the formula previously explained in section 6.2.1 to the Top Ontology (e.g. `f_ton Property`).

feat.	description
f_sumo	first sumo label
a_sumo	all sumo labels
f_semf	first wn semantic file label
a_semf	all wn semantic file labels
f_tonto	first ewn top ontology label
a_tonto	all ewn top ontology labels
f_magn	first domain label
a_magn	all domain labels

Table B.3: Knowledge-based Feature Set

feat.	description
tgt_mnp	syntactical relations of the target word from minipar
rels_mnp	all syntactical relations from minipar
yar_noun	NounModifier, ObjectTo, SubjectTo for nouns
yar_verb	Object, ObjectToPreposition, Preposition for verbs
yar_adjs	DominatingNoun for adjectives

Table B.4: Syntactical Feature Set

- **f_smf (f_semf)**: the first WordNet Semantic File label applying the formula previously explained in section 6.2.1 to the Semantic Files (e.g. `f_smf 42`).

Finally, table B.4 describes the **syntactic features** which contains features extracted using two different tools: Dekang Lin's Minipar and Yarowsky's dependency pattern extractor. Next, it is shown some local features generated from previous example:

- **mptgt (tgt_mnp)**: minipar dependency relation to or from the target word (e.g. `mptgt s:be`).
- **mprls (rels_mnp)**: minipar dependency relations of the example (e.g. `mprls dealer:nn:market`).
- **NounModifier (yar_noun)**: noun modifier (e.g. `mosd NounModifier Pop/NN`).
- **SubjectTo (yar_noun)**: subject to (e.g. `mosd SubjectTo is/VBZ`).

Appendix C

Resources and Web References

1. Altavista Web Search Engine
   http://www.altavista.com
2. British National Corpus
   http://www.natcorp.ox.ac.uk
3. DSO Corpus
   http://www.ldc.upenn.edu/Catalog/LDC97T12.html
4. EuroWordNet
   http://www.illc.uva.nl/EuroWordNet/
5. FreeLing
   http://garraf.epsevg.upc.es/freeling
6. Global WordNet Association
   http://www.globalwordnet.org
7. hard, line and serve corpora
   http://www.d.umn.edu/~tpederse/data.html
8. interest corpus
   http://crl.nmsu.edu/cgi-bin/Tools/CLR/clrcat#I9
9. LazyBoosting
   http://www.lsi.upc.es/~escudero
10. Linguistic Data Consortium (LDC)
    http://www.ldc.upenn.edu

11. Magnini Domains
    http://wndomains.itc.it
12. Meaning Project
    http://www.lsi.upc.es/∼meaning
13. Minipar
    http://www.cs.ualberta.ca/∼lindek/minipar.htm
14. Open Mind Word Expert
    http://www.teach-computers.org/word-expert.html
15. Ripper Software
    http://www.wcohen.com
16. Romanseval
    http://www.lpl.univ-aix.fr/projectes/romanseval
17. Semcor corpus
    http://www.cs.unt.edu/∼rada/downloads.html#semcor
18. Senseval
    http://www.senseval.org
19. SUMO
    http://www.ontolgyportal.org
20. SVM-light
    http://svmlight.joachims.org
21. TiMBL Software
    http://ilk.kub.nl/sofware.html
22. TNT
    http://www.coli.uni-saarland.de/∼thorsten/tnt
23. NLP on Wikipedia
    http://en.wikipedia.org/wiki/Natural_language_processing
24. WordNet
    http://wordnet.princeton.edu
25. WordSmyth
    http://www.wordsmyth.net

Appendix D

List of Acronyms

AB	*AdaBoost.MH*
AI	*Artificial Intelligence*
BC	*Brown Corpus*
cc	*constraint classification*
CL	*Computational Linguistics*
DL	*Decision Lists*
EB	*Exemplar-based*
IR	*Information Retrieval*
kNN	*k Nearest Neighbour*
LB	*LazyBoosting*
LDC	*Linguistic Data Consortium*
MBL	*Memory Base Learning*
MFC	*Most Frequent Sense Classifier*
MFS	*Most Frequent Sense*
ML	*Machine Learning*
MT	*Machine Translation*
MVDM	*Modified Value Difference Metric*
NB	*Naive Bayes*
NL	*Natural Language*
NLP	*Natural Language Processing*

NLU	*Natural Language Understanding*
ova	*one versus all*
P	*Proper name* label in Senseval corpora
PEB	*Positive Exemplar-based*
PNB	*Positive Naive Bayes*
PoS	*Part-of-Speech*
SVM	*Support Vector Machines*
TC	*Text Categorisation*
TSVM	*Transductive Support Vector Machines*
U	*Unassignable* label in Senseval corpora
VSM	*Vector Space Model*
WSD	*Word Sense Disambiguation*
WST	*Word Sense Tagging*
WSJ	*Wall Street Journal* corpus