High-performance Multilingual Semantic Role Labeling

Anders Björkelund Love Hafdell

Examensarbete för 30 hp
Institutionen för datavetenskap, Naturvetenskapliga fakulteten, Lunds Universitet

Thesis for a diploma in computer science, 30 ECTS credits
Department of computer science, Factulty of science, Lund University

Abstract

In this thesis, we have explored automatic semantic role labeling in a multilingual setting. Essentially, we aim to interpret natural language text along the lines of questions such as "*who* did *what* to *whom*". Additionally, we aim to answer questions of the type *when* and *where*, i.e. resolving more general characteristics surrounding events. This is modeled by considering the event as the core structure that defines a set of logical relations, *semantic roles*, that describe the relation between event and participant.

We propose an automatic and language-independent method that utilizes statistical classifiers to extract these roles. The proposed system has been implemented and took part in an international evaluation, the CoNLL 2009 Shared Task, which targeted seven languages: Catalan, Chinese, Czech, English, German, Japanese, and Spanish. Our system achieved the second best average semantic score, and the top results in Chinese and German.

Sammanfattning

I den här uppsatsen har vi utforskat automatisk semantisk rolletikettering i en flerspråkig miljö. Väsentligen försöker vi tolka naturligt språk med avseende på frågor av typen "*vem* gör *vad* för *vem*". Vidare försöker vi även besvara frågor såsom *när* och *var*, d.v.s. att förstå mer generella begrepp som beskriver händelser. Detta modelleras genom att betrakta händelsen som den centrala struktur som definierar en uppsättning logiska relationer, *semantiska roller*, som beskriver relationen mellan händelse och deltagare.

Vi föreslår en automatisk och språkoberoende metod som nyttjar statistiska klassificerare för att extrahera dessa roller. Det föreslagna systemet har implementerats och deltagit i en internationell utvärdering, the CoNLL 2009 Shared Task, som avsåg sju språk: engelska, japanska, katalanska, kinesiska, spanska, tjeckiska, och tyska. Vårt system uppnåde det näst bästa genomsnittliga semantiska resultatet, och de bästa resultaten i kinesiska och tyska.

Contents

1 Introduction **1**

1.1 An Introductory Example 1

1.2 Outline 2

2 Background **3**

2.1 Corpora and Annotation 3

2.1.1 Morphology 3

2.1.2 Voice 4

2.1.3 Parts of Speech 4

2.2 Syntactic Structures 4

2.2.1 Syntactic Dependency Trees 5

2.3 Semantic Dependencies 6

2.3.1 Predicate-Argument Structures 7

2.3.2 Semantic Lexicons 7

2.3.3 Alternative Role-representations 8

2.3.4 Semantic Roles and Dependency Grammars 9

2.3.5 Sentences with Multiple Predicates 9

2.4 Evaluation Measures 9

2.5 Machine Learning Tools 10

2.5.1 Linear Classifiers and Support Vector Machines 10

2.5.2 LibLinear 11

3 Corpora and Evaluation **13**

3.1 The CoNLL-2009 Shared Task 13

3.1.1 Variations of the Task 13

3.1.2 Previous Tasks 14

3.2 Corpus Content 14

3.2.1 Differences in Semantic Annotation 14

3.2.2 Semantic Lexicons 15

3.2.3 Data Format 15

4 Automatic Semantic Role Labeling **19**
4.1 Previous Work . . . 19
4.1.1 Global Models . . . 19
4.2 The Baseline System . . . 20
4.2.1 Predicate Disambiguation . . . 20
4.2.2 Argument Identification . . . 22
4.2.3 Argument Classification . . . 22
4.2.4 Separating Classifiers over POS tags . . . 22
4.3 Features . . . 23
4.3.1 Features Used . . . 23
4.3.2 Constructing the Features . . . 25
4.3.3 Constructing Pairs . . . 26
4.3.4 Feature Selection . . . 26

5 Introducing a Global Model **29**
5.1 Generating Candidate Propositions . . . 29
5.2 The Global Reranker . . . 30
5.3 The Reranker Feature Space . . . 31
5.3.1 Global Features . . . 32
5.3.2 Using Voice . . . 32
5.4 Weighting the Models . . . 33
5.5 Reranker Potential . . . 33
5.6 Tuning the Parameters . . . 33

6 Results **35**
6.1 Baseline Results . . . 36
6.2 Applying the Reranker . . . 37
6.3 The CoNLL 2009 Shared Task Evaluation . . . 38

7 Discussion **41**
7.1 Conclusion . . . 41
7.2 Reranker Potential . . . 41
7.3 Further Work . . . 42

A Features Used **45**

References **51**

Chapter 1

Introduction

In this thesis, we have explored automatic semantic role labeling (SRL) in a multilingual setting. Essentially, we aim to interpret natural language[1] text along the lines of questions such as "*who* did *what* to *whom*". From this point of view, *what* can be seen as an event taking place, whereas *who* and *whom* correspond to participants of the event. Additionally, we aim to answer questions of the type *when* and *where*, i.e. resolving more general characteristics surrounding an event.

During recent years, automatic semantic analysis of natural language text has received much attention by the natural language processing (NLP) community. With the advent of semantically annotated corpora and corresponding semantic lexicons such as FrameNet and PropBank, this type of automatic analysis has become feasible.

The SRL problem, as defined by Baker et al. (1998), models this by considering the event as the core structure that defines a set of logical relations, *semantic roles*, that describe the relation between a participant and an event. Various types of representation for these semantic roles exist, ranging from using a set of universal semantic roles, such as AGENT, PATIENT, and INSTRUMENT, to more specific types of roles that are specifically defined with respect to a certain event or class of events.

We have developed a generic system that carries out SRL on top of a dependency-based syntactic representation. The system took part in an international evaluation, the CoNLL 2009 Shared Task, and received the second best semantic score among twenty participants. Our participation in the task was essential to this thesis as it provided us with annotated corpora for training and evaluation in seven languages: Catalan, Chinese, Czech, English, German, Japanese, and Spanish.

1.1 An Introductory Example

In Fig. 1.1, we present how the sentence *Warren handed him the metal box and disappeared* is represented syntactically (above the text) and semantically (below the text). The syntactic structure describes binary relations, *syntactic dependencies*, between words. Each word is thought to depend on another, and each word in the sentence directly or transitively depends on an artificial *root* word. Dependencies point from *head* to *dependent*; for instance in Fig. 1.1, *Warren* depends on *handed*.

[1] Natural languages as opposed to formal languages.

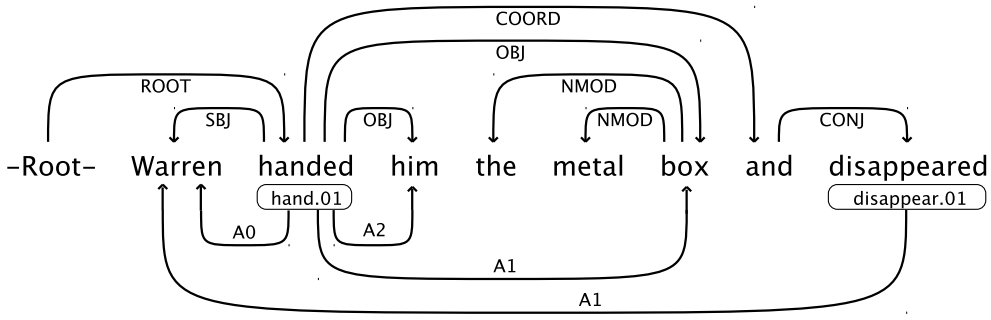

Figure 1.1: An example sentence with semantic frames from the PropBank lexicon.

The semantic representation below the text features two events: *hand* and *disappear*. The participants in these events are pointed out by the *semantic dependencies*, i.e. the arcs originating from *hand.01* and *disappear.01*. The labels attached to these dependencies correspond to semantic roles specified in a lexicon with respect to each event. Indeed, A1 is listed as "thing given" in the *hand.01* entry whereas it corresponds to "the disappeared" in the *disappear.01* entry.

Although the A1 dependency of *hand.01* points towards *box*, the actual participant is in fact *the metal box*. This follows from the syntactic dependencies originating from *box*, i.e. *the* and *metal* depend on *box*, and we consider the complete phrase as the participant of the handing.

Given this brief example, semantic role labeling can be considered finding and labeling the semantic dependencies in the figure.

1.2 Outline

The rest of this thesis is structured as follows. In *Chapter 2* we give an introduction to corpus linguistics and the syntactic and semantic formalisms used throughout this thesis. We also introduce the evaluation measures and machine-learning tools we have used. *Chapter 3* describes the CoNLL-2009 Shared Task and the corpora we have used. In *Chapter 4* we review previous work and present the baseline system that conducts SRL using a pipeline of statistical classifiers. In *Chapter 5* the baseline system is extended with a global model to offset the local decisions made by each step in the pipeline. We present various measurements of the system performance, as well as our contribution to the CoNLL 2009 Shared Task, in *Chapter 6*. The thesis is concluded by *Chapter 7* which gives a summary of the project, and suggests possible improvements to our system.

Chapter 2

Background

In this chapter, we give an introduction to natural language processing (NLP) and the tools and measures we use. The first section describes some basic grammatical tools and the notion of corpus linguistics. In Sect. 2.2, we go on and introduce the syntactic formalism that we use throughout this thesis. We then describe semantic dependencies in Sect. 2.3, which is the main target of this thesis. We define the measures we have used to evaluate performance in Sect. 2.4. The last section is devoted to the machine-learning tools we have utilized. Again, this is merely an introduction to define the concepts we use in our system and not an exhaustive survey.

2.1 Corpora and Annotation

Our approach to NLP is based on analyzing large samples of "real world" text, called *corpora* (*corpus* in singular). In its most simple form, a corpus consists of plain text. Aside from words, a corpus contains all kinds of special symbols such as parentheses and punctuation marks. We consider all these words or symbols as individual units, *tokens*, and require our corpora to be *tokenized*, i.e. sentences are split up into sequences of tokens. Once tokenized, a corpus is commonly annotated with linguistic meta-information that describe individual tokens. The tedious job of annotation is carried out by linguists with knowledge of the particular language in question. The manual annotation is considered correct and this is commonly referred to as the *gold standard* annotation.

2.1.1 Morphology

Morphology is the branch of linguistics that deals with analysis and description of the structure of words. For example, consider the English the words *runs*, *ran*, and *running*. They all derive from a common base word, or *lemma*, *run*, typically the word that one would look for in a dictionary. The various forms of *run* carry different information about grammatical categories such as *case*, *number*, *person*, and *tense*. Each category corresponds to meanings from some conceptual domain and are mutually exclusive. In English, for example, the *number* of a word can be either "singular" or "plural", but not both.

2.1.2 Voice

A particularly interesting morphological feature of verbs is called grammatical *voice*. In English there are two voices, *active* and *passive*. Roughly, a verb can be said to be in its active voice if the subject is the agent of the verb, i.e. the one that performs the action described by the verb. In passive voice, however, roles are reversed and the patient, i.e. the beneficiary of the action, is grammatically the subject of the verb. Consider the following example

$$\textit{John bathed the dog} - (\textit{active})$$

$$\textit{The dog was bathed by John} - (\textit{passive})$$

2.1.3 Parts of Speech

The *parts of speech* (POS) of tokens (such as nouns, verbs, adjectives etc.) is commonly also included in the annotation of a corpus. In order to be more useful, the traditional POS categories are usually further subdivided into more specialized ones. We use labels, or *tags*, to denote the different categories. For example the category of nouns, commonly denoted by the tag N, could be further divided into singular nouns, plural nouns, and proper nouns, denoted by the tags NN, NNS, and NNP, respectively. Moreover, words such as determiners, punctuation marks, and additional specialized words, can also be assigned their own category. See Fig. 2.1.3 for an example. In some corpora, these extensions of the set of POS categories correspond to morphological features of the words, for instance the subdivision of nouns into singular and plural.

Warren	handed	him	the	metal	box	and	disappeared	.
warren	hand	him	the	metal	box	and	disappear	.
NNP	VBD	PRP	DT	NN	NN	CC	VBD	.

Figure 2.1: Example sentence where each word is annotated with its lemma (middle) and POS-tag (bottom-most).

In addition to annotation on the token level, a corpus can also contain information on the structure of its sentences syntactic (Sect. 2.2) and semantic (Sect. 2.3) content.

2.2 Syntactic Structures

There are different ways to describe the syntactic structure of a sentence. Most previous work on semantic role labeling have utilized *constituent-based grammars*, in which a sentence is divided into a hierarchy of constituents. Examples of constituents are noun phrases and verb phrases which in turn can consist of constituents or single words. Intuitively this fits rather well with the problem of labeling semantic roles, as a participant not necessarily is denoted by a single word, but rather a complete phrase. For instance, *the metal box* in Fig. 1.1, makes up a single noun-phrase constituent using this type of grammar.

Although constituent-based grammars have successfully been used to perform SRL before, this has primarily been done in English, a language whose syntax can rather successfully be described using this type of grammar. However, languages with a more free word order, e.g. Slavic languages, are significantly harder to describe using this grammar (Mel'čuk, 1988) and for our multilingual setting we require a more versatile grammar.

A different approach to syntactic structure is *dependency-based grammars* which perceives connections between pairs of words. This grammar was introduced by Tesnière (1959) and has lately received much attention from the NLP community. Moreover, Johansson (2008) investigated the difference in performance between a constituent-based and a dependency-based SRL system that both were developed to be as similar as possible, and found that they perform roughly equally well in English.

2.2.1 Syntactic Dependency Trees

Formally, a *syntactic dependency tree* is defined as a *directed acyclic graph* (DAG) on which various constraints (e.g. rootedness) are imposed. Although some constraints and definitions can be altered, resulting in slightly different types of DAGs, in this thesis we restrict ourselves to the trees we have used. For simplicity, we give an informal introduction by means of an example given below. The DAGs we have used can (using appropriate definitions) be proved to be directed and rooted trees with edges originating from the root. For a more theoretical introduction we refer the reader to textbooks such as Hudson (1984).

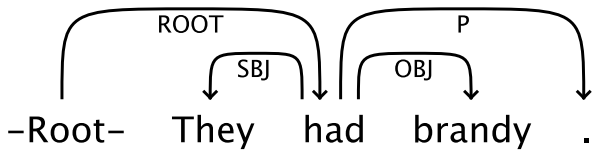

Figure 2.2: A sentence with four dependencies. *They*, *brandy*, and . depend on the main verb *had*, which in turn depends on the root token.

In a syntactic dependency tree, binary relations between tokens are assumed. Linguistically, each token is assumed to be governed by another token, and each token in the sentence is directly or transitively governed by an artificial *root* token. The tree is created in such a way that the tokens of all subtrees create phrases (i.e. they can function as single units in the syntax of the sentence similar to constituents in constituent grammars). A parent word is called *head*. A head-child relation is called a *dependency* and is drawn as an arc pointing from the head to the child.

The set of tokens in a subtree is called the *yield* of its root token and we say that a token *dominates* all the words in its yield. The head word determines the grammatical behavior of a dependency while the child word specifies its parent. A *labeled* syntactic dependency tree also has *labels* attached to each dependency. The labels give information about the grammatical function of the dependency, such as subject or object.

In the example sentence in Fig. 2.2, the *root* is the parent of the main verb *had*. *had* is in turn the head of three dependents: the dependency with the grammatical function SBJ (subject), where

the child word is *They*; the dependency with the grammatical function OBJ (object) with the child *brandy*; and the period sign with the function P (punctuation).

In fact, as a token determines its head, which in turn specifies its head in a transitive manner, we see that a parent, in a dependency, is actually specified by the whole phrase formed by the child token plus all tokens in the yield of the child. This makes syntactic dependency trees practical since they can represent complex multi-word word grammatical structures, similar to constituent grammars, while only defining binary relations between words. Since these are binary relations between tokens, these word-to-word dependencies can be kept intact even if the word order changes.

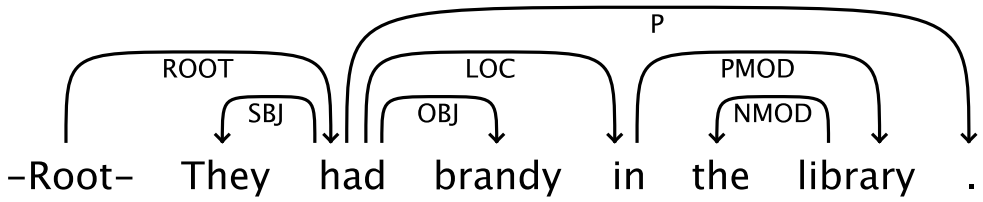

Figure 2.3: The resulting sentence after adding the adjunct *in the library* to the sentence in Fig. 2.2

If the sentence in Fig. 2.2 is extended by adding the adjunct *in the library*, we obtain the sentence in Fig. 2.3. One new dependency is created between *had* and *in*. The function of this new dependency is denoted by LOC and corresponds to locative adjunct. In this case it is the whole phrase *in the library* that acts as a locative adjunct rather than the word *in* alone, although there is only a direct dependency between *had* and *in*. The internal structure of the subtree rooted in *in* is independent of the already existing dependencies and the already existing dependencies are not affected by its addition.

The exact meaning of how a word *determines the grammatical behavior* of a phrase, or for a phrase to *specify its parent word*, is not completely clear and we will not address this topic here nor will we describe the set of labels used for the dependencies. We will simply assume that it is possible to create such a dependency tree, that a suitable set of labels exists, and that the labeled syntactic trees for each sentence is being created in a uniform way in each corpus.

2.3 Semantic Dependencies

The semantic interpretation we will take on in this thesis relies on the assumption that the meaning of a complex expression is determined by its structure and the meanings of its constituents. Although this approach ignores the way that the meaning of larger expressions seem to depend on additional factors, such as the rhetorical context in which the statement was made, it provides a suitable framework for automatic semantic analysis (Johansson, 2008). Again for simplicity, we introduce the central concepts of predicate-argument structures and how they represent information informally. This example is given using the PropBank **??** formalism and lexicon (described more thoroughly later in this section). For a more theoretical introduction, the reader is referred

to Bornkessel et al. (2006).

2.3.1 Predicate-Argument Structures

We distinguish between *events* and *participants* and refer to them as *predicates* and *arguments*, respectively. Each predicate defines a set of *core* arguments semantically central to that predicate. In addition to the core arguments of a predicate, there are also adjuncts, or *noncore* arguments. These noncore arguments give information such as time and location. In this way, we can view a predicate as having a number of abstract *roles* for possible arguments (core and noncore). A complete predicate-argument structure is referred to as a *proposition*, and a proposition without any participants, i.e. without arguments, is referred to as an *empty proposition*.

An example of a predicate is the event indicated by the verb *have* in Fig. 2.4. The predicate is referred to as *have.03* to differentiate it from other meanings of the verb *have*. The core arguments defined by the PropBank lexicon for the predicate *have.03* is the "owner" (A0) and the "possession" (A1).

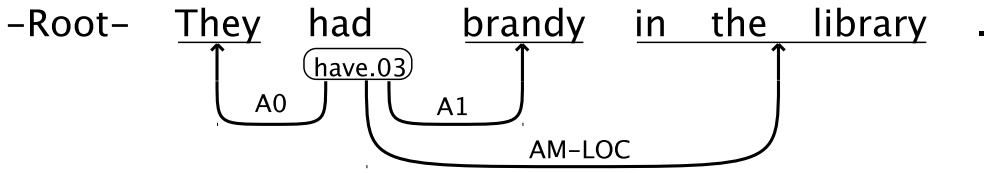

Figure 2.4: A sentence including the representation of the event *have* and its semantic dependencies

In addition to the core arguments, an event may be further described by non-core arguments that carry information such as degree, duration, manner, location, or time. For instance, the sentence in Fig. 2.4 describes not only *who* and *what*, but also *where*, which is denoted by the AM-LOC label on the semantic dependency.

Nonverbal predicates

In the previous example, the predicate is a verb. Events can also be described by noun, i.e. a *nominal* predicate. For example the sentences *He napped* and *He took a nap* describe the same event. In the latter sentence however, the noun *nap* is considered a predicate.

To a certain degree, nominal and verbal predicates do not necessarily suffice to catch the complete semantic information of a sentence. This is essentially a linguistic issue and is in our thesis governed by the conventions used when the corpora were annotated. For instance, the Czech corpus (Hajič et al., 2006) comprises a vast variety of predicates.

2.3.2 Semantic Lexicons

The corpora we have used in this thesis are all annotated with semantic dependencies drawn from semantic lexicons defined specifically for each language. For instance in English, the nominal

predicates are drawn from the NomBank (Meyers et al., 2004) lexicon, and the verbal predicates from PropBank (Palmer et al., 2005) lexicon. They provide a large amount of semantic *frames* that define the set of possible core arguments of various predicates. In the English corpus, these core arguments are denoted by labels A0 through A5, and each frame specifically defines the meaning of the core labels.

Although the meaning of the core labels are defined for each frame, the labels A0-A5 roughly correspond to the grammatical proximity of objects with respect to the predicate, i.e. A0 often refers to subjects, A1 to direct objects and so on.

As words can have multiple meanings depending on context, the same surface word, in fact the same lemma, can relate to different semantic frames. For instance, the predicate *had* in Fig. 2.4 corresponds to the frame called *have.03*. However, in the sentence *Mary had John killed*, *had* semantically has another meaning. Using the PropBank frames, this other frame is denoted *have.04*. We refer to these frame names as the *sense label*, or just *sense*, of a predicate.

However, not all predicates are ambiguous and proper frames can usually be deduced directly from the lemma. The English lexicons simply assign a numbered suffix to the lemma of the predicate starting at 01.

2.3.3 Alternative Role-representations

It can be argued that describing each predicate specifically in this manner is a bit too detailed, and one can fairly wonder whether it is not possible to find more general types of relations between events and participants. To a certain degree this is what the basic grammar taught in school models, i.e. subjects, direct and indirect objects, adjuncts and so on. However, the granularity of these concepts are not detailed enough. Moreover, as described in the example involving grammatical voice in Sect. 2.1, the subject of a passive verb does not necessarily constitute the performer, or *agent*, of an event, but rather the *patient*, which in active voice is normally an object.

Designing a universal set of roles that is general enough to catch all types of relations, yet precise enough to give sufficient amount of details is a linguistic problem that has to be dealt with by constructors of semantic lexicons. The VerbNet (Levin, 1993) lexicon solves this by grouping verbs into different classes and defining roughly 20 roles such as AGENT, INSTRUMENT, EXTENT, and PATIENT, and imposes restrictions as to what roles may be assigned to certain groups of verbs.

Other Lexicons

Intuitively there may be several words describing the same type of event. Consider the predicate *sleep*, as in *He sleeps during night*. This predicate is closely related to other predicates such as *nap* and *slumber*. This fact is exploited by the FrameNet lexicon (Baker et al., 1998), which defines a single frame for all these words that consequently share the same set of core arguments. According to FrameNet, the semantic frame corresponding to *sleep* represents events where “A sleeper stays in an altered state of consciousness known as sleep”, and a total of 23 different predicates belongs to this frame.

2.3.4 Semantic Roles and Dependency Grammars

When a sentence that is annotated with syntactic dependencies is also annotated with semantic information, the *semantic dependencies* are used to represent the arguments of the predicates. An argument is represented by an arc pointing from the predicate to a word in the sentence. The yield of the target word constitutes the argument. This rule comes with an exception in the case where the predicate itself is part of the yield of the argument. In these cases, the argument is the words in the yield of the arguments target word *minus* all the words in the yield of the predicate.

2.3.5 Sentences with Multiple Predicates

Although the sentence in Fig. 2.4 only features one predicate, a sentence can certainly have more than one predicate. In these cases, the arguments of different predicates may overlap partially or completely. The example given in the introduction, Fig. 1.1, shows a sentence with two predicates representing two different events (an object being handed and somebody disappearing). *Warren* participates in both events, and is consequently an argument of both predicates.

2.4 Evaluation Measures

We use a number of measures to evaluate the performance of our system. Moreover, as we use automatically parsed dependency trees that might not be entirely correct as input for our system, we are also interested in measuring the accuracy of these. To evaluate syntactic dependencies we use the *labeled attachment score* (LAS). This is simply defined as the percentage of words for which a system has predicted the correct head and syntactic label and is the most popular measure on correctness of dependency graphs.

In order to score predicate-argument propositions, we compare the semantic dependencies output by our system to the gold standard. We consider the proposition as a shallow tree with a root dependency from the predicate to the root of the sentence (these dependencies are omitted in the figures), and a number of dependencies from the predicate to the arguments. The label of the root dependency corresponds to the sense of the predicate, e.g. *have.03*. The labels of the argument dependencies are the ones given in the figures. Although there are various ways to measure correctness of semantic dependencies, we are primarily concerned with the *labeled semantic* F_1 ($SemF_1$) which takes several kinds of mistakes into consideration.

To calculate the SemF_1 we define two measures called *precision* and *recall* as:

$$precision = \frac{\#correct\ dependencies}{\#dependencies\ guessed},$$

$$recall = \frac{\#correct\ dependencies}{\#dependencies\ in\ gold\ standard}.$$

The SemF_1 measure is the precision and recall combined using the harmonic mean which is defined as:

$$SemF_1 = 2 \times \frac{precision \times recall}{precision + recall}.$$

This measure, which is commonly used to evaluate performance of SRL systems, is constructed in such a way that if a system assigns the incorrect predicate sense, it still receives some points for the arguments correctly assigned. However, as our implemented system performs frame assignment and argument identification and classification as separate steps, we require more precise measures that indicate how well we perform in each step. By considering argument or predicate dependencies exclusively, we can employ the SemF_1 measure with respect to either arguments or predicates. We refer to these measures as ArgF_1 and PredF_1, respectively.

2.5 Machine Learning Tools

The kind of machine learning we have used in this thesis is called *supervised learning*. The machine is trained by the use of *training examples*. The machine works on two sets; X the set of legal input objects and Y the set of possible outputs. The training examples are pairs (x_i, y_i), where $x_i \in X$ and $y_i \in Y$ is the desired output corresponding to x_i. The machine is called a *classifier* and it predicts a class label for the input objects.

2.5.1 Linear Classifiers and Support Vector Machines

A *support vector machine* (SVM) is a classifier where $X \subseteq \mathbb{R}^n$ (real valued n-dimensional vectors) and $Y = \{-1, +1\}$. That is, its inputs are vector representations of the input objects.[1] The input part, x_i, of the training examples is partitioned in to two sets *positive* and *negative*. This is done in such a way that x_i belongs to *positive* if its corresponding $y_i = +1$ and *negative* otherwise. The SVM finds a hyperplane that separates the vector space X so that members of *positive* and *negative* end up on different sides. The margin between $x_i \in X$ and the hyperplane is the shortest distance between them. The constructed hyperplane, in addition to separating *positive* and *negative*, is also required to maximize the smallest margins of any point in either set (Cristianini and Shawe-Taylor, 2000).

There are no guarantees that linear separation is possible. To address this problem a nonlinear transformation can be made from the original nonlinear observations into a higher-dimensional space where linear separation is possible. This is known as the kernel trick, and makes a linear classification in the new space equivalent to nonlinear classification in the original space. However, it comes at the expense of transforming all points between these spaces, both during training and classification (Burges, 1998).

As previously mentioned, a SVM only allows binary classification. In a lot of cases it is, however, desirable to use more than two possible classes, i.e. a *multiclass* classifier. There are

[1] In case there are categorical attributes in the input, we first have to convert them into numeric data e.g. a three-category attribute such as red, green, blue can be represented as (0,0,1), (0,1,0), and (1,0,0).

three major ways of performing this: *one-against-all*, *one-against-one*, and the *directed acyclic graph method*. The simplest of these methods is one-against-all, in which one regression classifier is trained for each class assuming that all examples belonging to its corresponding class are to be considered as positive and all other examples are to be considered negative. When predicting a class for a given input, all classifiers are tested and the class corresponding to the classifier with the highest certainty is taken as output (Hsu and Lin, 2001).

2.5.2 LibLinear

In the system we have implemented, we have exclusively used the LIBLINEAR library (Fan et al., 2008). This is an efficient, freely available[2] linear classifier. It supports linear SVMs as well as *logistic regression*, a model commonly used in statistics to predict binary probabilities.

The logistic model can be extended for multiclass problems. In computational linguistics, this type of classifiers are referred to as *maximum entropy* (ME) classifiers, and are often used for large-scale classification problems. Maximum entropy classifiers are fast and can classify a large set of instances much faster than SVMs with non-linear kernels (Lin et al., 2008).

These classifiers proved to be useful as they allowed us to do arithmetic with the probabilities obtained in each classification step. The downside of this model is its inability to deal with data that is not linearly separable. We addressed this by constructing pairs of features and will get back to this in Sect. 4.3.3.

[2] Available at http://www.csie.ntu.edu.tw/~cjlin/liblinear/.

Chapter 3

Corpora and Evaluation

We evaluated our system on seven languages that were all given in a unified data format. The languages were Catalan and Spanish (Taulé et al., 2008), Chinese (Palmer and Xue, 2009), Czech (Hajič et al., 2006), English (Surdeanu et al., 2008), German (Burchardt et al., 2006) and Japanese (Kawahara et al., 2002). The training and testing data, i.e. corpora, we have used was provided by the CoNLL 2009 Shared Task. However since the corpora originally come from different sources the scheme of annotation, particularly the semantic annotation, does not follow the same pattern in all languages. In this chapter, we describe the CoNLL 2009 Shared Task and the corpora we have used.

3.1 The CoNLL-2009 Shared Task

The Conference on Natural Language Learning (CoNLL) is an annual conference on the topic of natural language learning. Each year the conference comes with a Shared Task that provides a basis for a global comparison between teams all over the world. The 2009 Shared Task[1] consisted of extracting syntactic and semantic dependencies in all seven languages (Hajič et al., 2009).

3.1.1 Variations of the Task

The task is divided into a *closed* and an *open* challenge. In the closed task, only the official training data provided by the the organizers were allowed to be used, whereas in the open challenge usage of external resources was allowed. Hence the closed task gives a fair environment for all participating teams, whereas there is room for improvement in the open challenge for teams that have additional resources at hand. Teams were allowed to participate in the closed challenge, the open challenge, or in both.

Due to the vastness of the task, teams could optionally participate in the SRL-only track of the task and skip the syntactic parsing. For that purpose predicted dependencies generated by state-of-the-art dependency parsers were provided by the organizers.

[1] Official website at http://ufal.mff.cuni.cz/conll2009-st/.

3.1.2 Previous Tasks

The 2009 task is a direct extension of the monolingual (English only) 2008 task which was also dedicated to syntactic and semantic dependencies (Surdeanu et al., 2008). The 2004 and 2005 Shared Tasks (Carreras and Màrquez, 2004, 2005) were also devoted to SRL in English, although in these tasks the training and testing corpora were much smaller, the syntactic representation was constituent-based, and SRL corresponded to labeling constituents. To our knowledge, the 2009 task was the first time multilingual SRL was evaluated in a fair environment.

3.2 Corpus Content

Aside from the lexical form of each token in the corpora, additional linguistic features were annotated including the the lemma and the POS tag of the token. Moreover, in all languages but Chinese and English, a set of additional morphological features were provided. These are generally referred to as FEATS, and we will make subsequent use of this name in later chapters. To a certain degree, these features weigh up the less detailed POS tags in these languages. I.e. Catalan, Czech, and Spanish feature relatively few POS tags, whereas English and Chinese have a richer set of POS tags. As mentioned in Sect. 2.1, these fine-grained POS tags often carry morphological information such as tense, case, or number, i.e. the same type of information that is provided by the FEATS.

The corpora had both gold standard and predicted values of the attributes mentioned above. The predicted values were generated by the Shared Task organizers using state-of-the-art NLP tools in each language. In the final test set the gold standard annotation was not to be used and participants consequently had to resort to the predicted values. This makes the results comparable to what can be achieved in any practical application, i.e. when no gold standard annotation is available. For exact details on the tools used and the contents of the FEATS, we refer to the Shared Task description paper (Hajič et al., 2009).

One additional bit of information that was provided both during training and testing was whether a token is a predicate or not. This corresponds to the SRL subproblem of identifying predicates and we get back to this in the next chapter. This essentially simplifies the problem and implies a higher PredF_1 score.

3.2.1 Differences in Semantic Annotation

A number of corpora statistics taken on the training corpus in each language are shown in Table 3.1. It gives the size of the corpora, the number of labels used for syntactic and semantic dependencies as well as number of POS tags. The figures give a rough idea to the differences between the corpora.

The semantic annotation is more exhaustive in some languages than others. In Chinese and German, only verbal predicates are annotated. Moreover, the German annotation is particularly sparse not covering all verbs that intuitively qualify as predicates. English, as well as Japanese, features nominal predicates. The last row of Table 3.1 shows what POS tags appear as predicates. The '*' is used to denote prefixes in languages that had a richer set of POS tags. V denotes verbs, N nouns, and A and JJ denote adjectives (in Catalan and Spanish, and Japanese, respectively).

The Czech corpus is by far the most exhaustive containing a large amount of predicates of all kinds of POS tags, however almost half appear to have no arguments at all (cf. Table 3.1). To get an idea of how the number of predicates relate to the number of arguments, we define the argument-predicate ratio (APR) simply by

$$APR = \frac{\#argument\ dependencies\ in\ gold\ standard}{\#predicate\ dependencies\ in\ gold\ standard}.$$

The APR, shown in Table 3.1, turns out to be less than one in Czech. This implies that in the Czech corpus there are more predicate than argument dependencies, resulting in a SemF_1 score biased towards disambiguation of predicates.

	Catalan	**Chinese**	**Czech**	**English**	**German**	**Japanese**	**Spanish**
# sentences	13,200	22,277	38,727	39,279	36,020	4,393	14,329
# tokens	390,302	609,060	652,544	958,167	648,677	112,555	427,442
# POS-tags	12	41	12	48	56	40	12
# dependency labels	50	41	49	69	46	4	49
# argument labels	38	36	46 (60)	53	10	41 (66)	43
# predicate dependencies	37,431	102,813	414,237	179,014	17,400	25,712	43,824
# empty propositions	7	10058	209,733	483	76	0	27
# argument dependencies	84,367	231,869	365,255	393,699	34,276	43,957	99,054
APR	2.25	2.26	0.88	2.20	1.97	1.71	2.26
Avg. tokens/sentence	29.57	27.34	16.85	24.39	18.00	25.62	29.83
Avg. predicates/sentence	2.84	4.62	10.69	4.56	0.48	5.85	3.06
Predicate POS-tags	A,V	V*	*	N*,V*	V*	JJ,N*,S*,V*	A,V

Table 3.1: Training corpus statistics.

The Czech and Japanese annotations featured multiple dependencies between a predicate and a single token. This was modeled in the data format simply by considering a double dependency as a single one, and creating artificial labels such as A|B, where A and B are the respective labels of the two dependencies. We considered these composite labels as unique labels, increasing the number of labels in these languages. Table 3.1 shows the number of argument labels in each language. The numbers in parentheses for Czech and Japanese correspond to this extended number of labels.

3.2.2 Semantic Lexicons

As mentioned in Section 2.3.2, predicates are normally defined by a semantic lexicon associating an abstract *frame* of arguments to a predicate. Semantic lexicons were provided in all languages but Japanese, however they did not follow the same standard nor contain equivalent information. Moreover, the Japanese predicates were not ambiguous and the predicate sense labels corresponded to the lemma of the token.

3.2.3 Data Format

The corpora we have used were provided as UTF-8 encoded text files. Each line corresponds to a token, and includes a tab-separated list of annotation. An example of a sentence from the Spanish

training corpus (Taulé et al., 2008) is given in Table 3.2. Each token is assigned an index (ID) listed in the first column. The second column corresponds to the surface form of the tokens. The following six columns contain gold standard and predicted Lemma, POS, and Feats. Columns beginning with a bold **P** correspond to predicted values. Underscores correspond to null values.

The head column denotes the ID of the syntactic head of the token. The ID 0 corresponds to the root token. Deprel denotes the label of the syntactic dependency to the head of the token. FillPred indicates whether the token is a predicate. The Sense column contains the sense label of the predicate.

The rest of the columns are used to denote argument dependencies. The first column (APred1) corresponds to the first predicate in the sentence, the second column (APred2) to the second predicate, and so on. As the number of predicates varies among sentences, so does the number of APred columns. A token with a nonnull value in an APred column, is an argument of the corresponding predicate. For instance, the token with ID 2, *acuerdo*, is an argument of the predicate *suponer.c2* with the label ARG1-TEM.

During training of classifiers, all columns are available. The SRL-only track of the CoNLL 2009 Shared Task consisted of assigning sense labels to the predicates in the Sense column, as well as finding and labeling the argument dependencies that go in the APred columns. During parsing the gold standard annotation is not available, and the predicted values of lemma, POS, Feats, Head, and Deprel are used. The FillPred column, however, was available also during parsing. This column was provided by the Shared Task organizers due to the variations in semantic annotation of different languages.

ID	Form	Lemma	**P**Lemma	POS	**P**POS	Feats	**P**Feats	Head	**P**Head	Deprel	**P**Deprel	FillPred	Sense	APred1	APred2
1	El	el	el	d	d	postype=article|gen=m|num=s	postype=article|gen=m|num=s	2	2	spec	spec	_	_	_	_
2	acuerdo	acuerdo	acuerdo	n	n	postype=common|gen=m|num=s	postype=common|gen=m|num=s	4	4	suj	suj	_	_	arg1-tem	_
3	también	también	también	r	r	_	_	4	4	mod	mod	_	_	_	_
4	supone	suponer	suponer	v	v	postype=main|gen=c|num=s|...	postype=main|gen=c|num=s|...	0	0	sentence	sentence	Y	suponer.c2	_	_
5	que	que	que	c	c	postype=subordinating	postype=subordinating	8	8	conj	conj	_	_	_	_
6	los	el	el	d	d	postype=article|gen=m|num=p	postype=article|gen=m|num=p	7	7	spec	spec	_	_	_	_
7	accionistas	accionista	accionista	n	n	postype=common|gen=c|num=p	postype=common|gen=c|num=p	8	8	suj	suj	_	_	_	arg0-agt
8	recibirán	recibir	recibir	v	v	postype=main|gen=c|num=p|...	postype=main|gen=c|num=p|...	4	4	cd	cd	Y	recibir.a2	arg2-atr	_
9	un	uno	un	d	d	postype=indefinite|gen=m|num=s	postype=numeral|gen=m|num=s	10	10	spec	spec	_	_	_	_
10	pago	pago	pago	n	n	postype=common|gen=m|num=s	postype=common|gen=m|num=s	8	8	cd	cd	_	_	_	arg1-pat
11	de	de	de	s	s	postype=preposition|gen=c|num=c	postype=preposition|gen=c|num=c	10	10	sp	sp	_	_	_	_
12	capital	capital	capital	n	n	postype=common|gen=f|num=s	postype=common|gen=f|num=s	11	11	sn	sn	_	_	_	_
13	de	de	de	s	s	postype=preposition|gen=c|num=c	postype=preposition|gen=c|num=c	10	10	sp	sp	_	_	_	_
14	90	90	90	z	n	_	postype=proper|gen=c|num=c	15	15	spec	spec	_	_	_	_
15	francos	franco	franco	z	a	postype=currency	postype=qualificative|gen=m|num=p	13	13	sn	sn	_	_	_	_
16	suizos	suizo	suizo	a	a	postype=qualificative|gen=m|num=p	postype=qualificative|gen=m|num=p	15	15	s.a	s.a	_	_	_	_
17	(	(	(	f	f	punct=bracket|punctenclose=open	punct=bracket|punctenclose=open	19	19	f	f	_	_	_	_
18	53	53	53	z	n	_	postype=proper|gen=c|num=c	19	19	spec	spec	_	_	_	_
19	dólares	dólar	dólar	z	n	postype=currency	postype=common|gen=m|num=p	15	15	sn	sn	_	_	_	_
20	)	)	)	f	f	punct=bracket|punctenclose=close	punct=bracket|punctenclose=close	19	19	f	f	_	_	_	_
21	y	y	y	c	c	postype=coordinating	postype=coordinating	10	10	coord	coord	_	_	_	_
22	un	uno	un	d	d	postype=indefinite|gen=m|num=s	postype=numeral|gen=m|num=s	23	23	spec	spec	_	_	_	_
23	dividendo	dividendo	dividendo	n	n	postype=common|gen=m|num=s	postype=common|gen=m|num=s	10	10	sn	sn	_	_	_	_
24	especial	especial	especial	a	a	postype=qualificative|gen=c|num=s	postype=qualificative|gen=c|num=s	23	23	s.a	s.a	_	_	_	_
25	de	de	de	s	s	postype=preposition|gen=c|num=c	postype=preposition|gen=c|num=c	23	23	sp	sp	_	_	_	_
26	135	135	135	z	n	_	postype=proper|gen=c|num=c	27	27	spec	spec	_	_	_	_
27	francos	franco	franco	z	a	postype=currency	postype=qualificative|gen=m|num=p	25	25	sn	sn	_	_	_	_
28	suizos	suizo	suizo	a	a	postype=qualificative|gen=m|num=p	postype=qualificative|gen=m|num=p	27	27	s.a	s.a	_	_	_	_
29	(	(	(	f	f	punct=bracket|punctenclose=open	punct=bracket|punctenclose=open	32	32	f	f	_	_	_	_
30	unos	uno	uno	d	d	postype=indefinite|gen=m|num=p	postype=indefinite|gen=m|num=p	32	32	spec	spec	_	_	_	_
31	80	80	80	z	n	_	postype=proper|gen=c|num=c	30	30	z	z	_	_	_	_
32	dólares	dólar	dólar	z	n	postype=currency	postype=common|gen=m|num=p	27	27	sn	sn	_	_	_	_
33	)	)	)	f	f	punct=bracket|punctenclose=close	punct=bracket|punctenclose=close	32	32	f	f	_	_	_	_
34	por	por	por	s	s	postype=preposition|gen=c|num=c	postype=preposition|gen=c|num=c	8	23	cc	sp	_	_	_	argM-adv
35	cada	cada	cada	d	d	postype=indefinite|gen=c|num=s	postype=indefinite|gen=c|num=s	36	36	spec	spec	_	_	_	_
36	acción	acción	acción	n	n	postype=common|gen=f|num=s	postype=common|gen=f|num=s	34	34	sn	sn	_	_	_	_
37	de	de	de	s	s	postype=preposition|gen=c|num=c	postype=preposition|gen=c|num=c	36	36	sp	sp	_	_	_	_
38	Algroup	Algroup	Algroup	n	n	postype=proper|gen=c|num=c	postype=proper|gen=c|num=c	37	37	sn	sn	_	_	_	_
39	.	.	.	f	f	punct=period	punct=period	4	4	f	f	_	_	_	_

Table 3.2: Example of the Spanish sentence *El acuerdo también supone que los accionistas recibirán un pago de capital de 90 francos suizos (53 dólares) y un dividendo especial de 135 francos suizos (unos 80 dólares) por cada acción de Algroup.*, 'The agreement also means that shareholders will receive a payment of capital of 90 Swiss francs (53 dollars) and a special dividend of 135 francs (about 80 dollars) for each share of Algroup.' (our translation)
Columns prefixed by **P** denote predicted values.

Chapter 4

Automatic Semantic Role Labeling

In this chapter, we start by reviewing previous work in Sect. 4.1. The rest of the chapter is devoted to describing the *baseline* system. This constitutes the initial implementation and we use it to compare the performance of subsequent extensions that we introduce in the next chapter.

4.1 Previous Work

The usage of probabilistic methods to perform semantic role labeling was pioneered by Gildea and Jurafsky (2002) using the FrameNet corpus. In their work, they assumed semantic frames to be identified and only addressed identification and labeling of arguments. They applied an automatic constituent-based syntactic parser to the training data. Using the semantic annotation available in the FrameNet corpus, and the constituent-based parse trees, they extracted a number syntactic, grammatical, and lexical features on which two probability models were conditioned, one for identification and one for labeling argument constituents. These models were applied sequentially in a *pipeline* manner by first identifying argument constituents, then labeling them.

Since the work of Gildea and Jurafsky, the decomposition into a pipeline and the usage of syntactic parse trees for input has become the predominant method to tackle SRL. The simple probability models have since been replaced with more efficient statistical classifiers, commonly SVM or maximum entropy (ME) classifiers. Designing optimal feature sets for these classifiers still remain an open problem and has been addressed by multiple authors (Xue and Palmer, 2004, *inter alia*).

4.1.1 Global Models

Despite possible gains in feature engineering and improvement of machine-learning algorithms, we believe there is a ceiling to how far this method can reach. Different extensions to the pipeline-based architecture have been proposed (Punyakanok et al., 2004; Toutanova et al., 2005). Although these approaches are not identical, they both employ a similar strategy: generating several possible semantic parses of a sentence using a pipeline of classifiers, that are subsequently reevaluated by a separate inference mechanism. The key to success is the ability to globally (as opposed

to the pipelined classifiers that work locally) score complete propositions in such post-processing step, ruling out unlikely and/or contradicting parses generated by the pipeline.

Punyakanok et al. (2004) applied integer linear programming (ILP) to enforce structural and linguistic constraints, such as the *no duplicate core argument constraint* which claims a predicate can not have more than one core argument of each type (In English these are the labels A0 through A5).

Toutanova et al. (2005) carried out *beam search* in order to generate a pool of candidate parses. These candidates were filtered with respect to linguistic and structural constraints and the remaining candidates were subsequently scored by an additional classifier, a *reranker*.

Indeed, the top performing system (Johansson and Nugues, 2008a) of the CoNLL 2008 Shared Task employed a global model that utilized both linguistic constraints as well as a reranker.

4.2 The Baseline System

We constructed a baseline system that consisted of a pipeline of classifiers that parses text given in the CoNLL 2009 data format. Since predicates were given, we did not have to identify predicates ourselves. We decomposed the problem into three modules: predicate disambiguation (PD), argument identification (AI), and argument classification (AC).

Figure 4.1 shows how an example sentence is passed through the modules. The pipeline takes a sentence read into memory and passes each predicate through the pipeline of classifiers. The PD module assigns a sense label to each of the predicates that are pointed out (denoted by the boxes, however the proper frame is unknown, which is indicated by the suffix *.xx*). Knowing the proper sense, the AI module considers each token as an argument of each predicate and constructs an unlabeled proposition, i.e. the unlabeled semantic dependencies. In the last step, the AC module assigns the proper label to each argument dependency, which concludes the analysis.

4.2.1 Predicate Disambiguation

Frame assignment was treated exclusively as a word sense disambiguation problem. During training we constructed a table mapping each lemma to a list of every observed sense (i.e. frame name). For each lemma, we trained one multiclass classifier using all occurrences of this predicate lemma. Lemmas with a single observed sense were considered unambiguous and no classifier was trained, and the observed sense was assigned to every occurrence of that lemma during parsing.

Default Sense Labels

During parsing we can not assume that every predicate lemma we parse was seen during training and is in our table. To handle this, we defined default predicate sense labels in each language. The default sense labels were constructed differently in each language as the representation of sense labels varies. However, they were all based on the most common suffix or appearance of sense labels.

For instance, in English, the default label was obtained by concatenating the lemma with the suffix .01 yielding sense labels of the form *lemma.01*. The same procedure was used in Chinese and German, as the sense labels follow the same pattern in these languages.

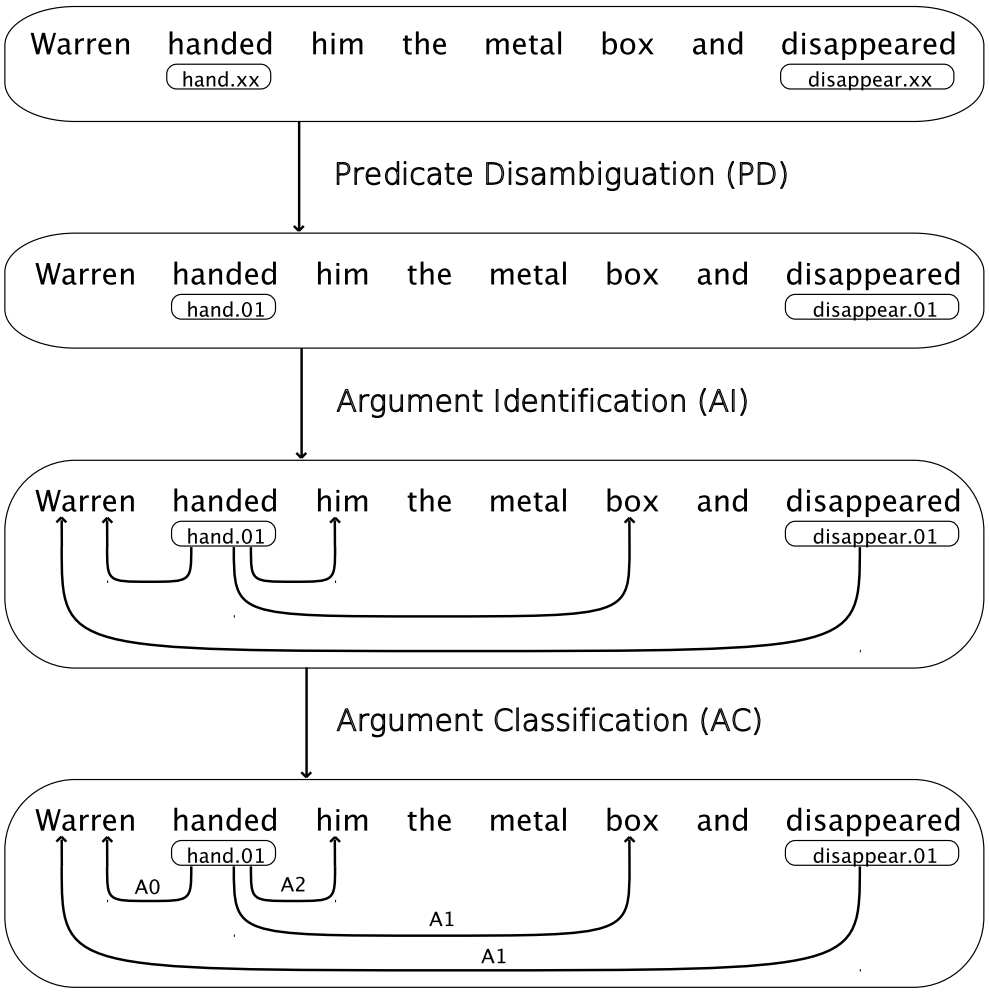

Figure 4.1: Sequential view over the steps performed by each module in the pipeline. Dependency graphs and other annotation is omitted from the images.

In Catalan and Spanish, sense labels follow a similar pattern, but the suffixes are on the form *a21, b21, a22, b23* etc. The most common suffix was .a21, and the default sense label was obtained by concatenating the lemma with this suffix, i.e. *lemma.a21*.

In the Czech corpus, the sense labels were sometimes identical to the lemma, or, more commonly, just an obscure string defined in the Czech semantic lexicon. These strings did not follow any pattern that could be exploited to generate default sense labels. Instead, we simply used the lemma of the predicate as the default label. Moreover, as the Czech training corpus was rather large, most predicates have been seen during training and default labels were rarely employed.

As mentioned in Sect. 3.2.2, Japanese predicates required no disambiguation and the sense label was identical to the lemma. Thus we did not have to train any classifiers for predicate disambiguation and used the predicate lemma as default sense label, and used this for every predicate.

4.2.2 Argument Identification

Once the predicate was disambiguated, the AI module identified the arguments of the predicate and created an unlabeled proposition. This was done by means of a binary classifier that marked a token as being either an argument of the predicate or not. The module simply works left-to-right, independently considering each token an argument of the predicate until it reaches the end of the sentence.

The training data for the AI classifier was generated in the same manner, i.e. for each predicate in a sentence of length *n*, *n* training examples were derived. For each token that had an argument dependency to the predicate, a positive training instance was obtained, and the rest of the tokens yielded negative instances. A consequence of this procedure is that there will be a significantly greater number of negative examples than positive examples, which could potentially bias the classifiers.

4.2.3 Argument Classification

The unlabeled proposition generated by the AI module was labeled using a multiclass classifier. It labeled each argument dependency independently. During training, each argument in the training corpus yielded one training instance. Each label corresponded to one class. Compared to the training data in the AI step, we had much fewer training instance for the AC module.

4.2.4 Separating Classifiers over POS tags

The English corpora is annotated with both nominal and verbal predicates that derive from the NomBank and PropBank lexicons, respectively. These annotations are not completely similar in their nature, and although the NomBank annotation was developed to be consistent with the PropBank frames, the actual arguments of the same abstract frames are not necessarily identical in terms of POS tags, lexical forms and other morphological attributes. We addressed this by training two specialized classifiers in each step, one for nominal predicates and one for verbal predicates, using only training examples belonging to the respective POS category. Moreover, using the feature selection procedure (described in Sect. 4.3.4), we were able to select specialized feature sets for nominal and verbal predicates.

This subdivision, however, comes at the cost of fewer training instances for each classifier as well as the need to run the feature selection procedure more than once in each step. As we rely on predicted POS tags to differentiate between nominal and verbal predicates, we risk not knowing which subclassifier to apply if a predicate has been wrongly tagged as neither noun nor verb (i.e. other part-of-speech tags than N and V). To address this problem, we also used a default classifier catching these wrongly tagged predicates. This classifier was trained using all the predicates available in the training corpus and used the union of the features of the nominal and verbal classifiers. In total, we had three classifiers in each step, and two executions of the feature selection procedure in each step.

Although other languages also featured predicates of multiple POS tags, we refrained from implementing this architecture in these languages. Primarily because the feature selection procedure turned out to be too time consuming, but also because separating classifiers would have resulted in too few training instances for some POS tags in some languages.

4.3 Features

We defined a large set of feature templates used by the statistical classifiers. Most of the features can be traced back to Gildea and Jurafsky (2002) and have been used by participants in previous CoNLL Shared Tasks devoted to SRL. Essentially, we used the features defined by Johansson and Nugues (2008b), except that we omitted language specific features and added features involving the morphological FEATS data.

Not all features were used by every classifier. The features used by each classifier were selected by a feature selection procedure described in Sect. 4.3.4. A complete list of features used by every classifier is given in Appendix A.

4.3.1 Features Used

The features described below are the linguistic features represented as strings or sets of strings. Some features extract information from the predicate, others from the potential (AI module) or identified (AC module) argument, and some extract information that depends on both predicate and argument and the syntactic path between these features. Consequently we refer to these features as *predicate features*, *argument features*, and *path features*.

Predicate Features

We describe each feature by means of an example sentence in German (Burchardt et al., 2006) that is given in Fig. 4.2. The value of a feature is either a string or a set of strings, and the examples are taken with respect to the predicate *mitteilen*.

PREDWORD, the surface word of the predicate, e.g. ‘mitteilte’;
PREDLEMMA, the lemma of the predicate, e.g. ‘mitteilen’;
PREDPOS, the POS tag of the predicate, e.g. ‘VVFIN’;
PREDDEPREL, the predicates deprel to its parent, e.g. ‘MO’;
PREDFEATS, the set of FEATS of the predicate, e.g. {‘3’,‘Sg’,‘Past’,‘Ind’};

PREDPARENTWORD, the surface form of the parent word of the predicate, e.g. ‘wurden’;
PREDPARENTPOS, the POS tag of the parent word of the predicate, e.g. ‘VAFIN’;
PREDPARENTFEATS, the set of FEATS of the parent word of the predicate, e.g. {‘3’,‘Pl’,‘Past’,‘Ind’};
DEPSUBCAT, the concatenation of the dependency labels of the predicate dependents with respect to the linear order of the sentence, e.g. ‘MO+SB’;
CHILDDEPSET, the set of dependency labels of the children of the predicate, e.g. {‘MO’,‘SB’};
CHILDWORDSET, the set of surface words of the predicate children, e.g. {‘Wie’,‘Polizei’};
CHILDPOSSET, the set of POS tags of the children of the predicate, e.g. {‘PWAV’,‘NN’};
SENSE, the predicate sense label, e.g. ‘mitteilen.1’.

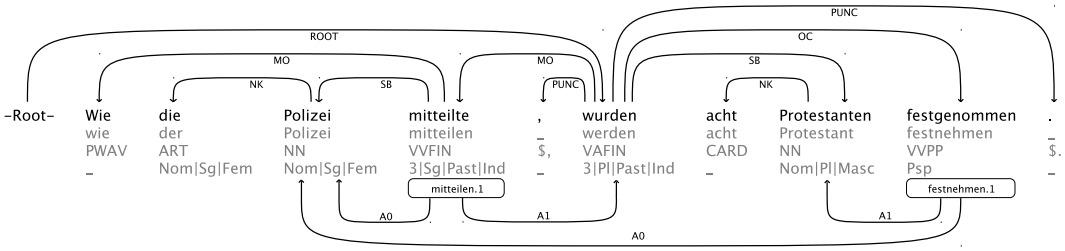

Figure 4.2: Example of the German sentence *Wie die Polizei mitteilte, wurden acht Protestanten festgenommen.*, ‘As the police announced, eight protesters were arrested’ (our translation). The sentence is annotated with lemma, POS, FEATS, and syntactic (above) and semantic (below) dependencies.

Argument Features

In addition to the predicate related features, the AI and AC classifiers also employ features centered around the argument. The examples given below use the same sentence from Fig. 4.2, and are taken with respect to the argument *Polizei*. Even though this token is an argument of both predicates in the sentence, the argument features evaluate to the same values regardless of whether it is considered an argument of one or the other.

ARGWORD, the surface form of the argument word, e.g. ‘Polizei’;
ARGPOS, the POS tag of the argument word, e.g. ‘NN’;
ARGFEATS, the set of FEATS of the argument {‘Nom’,‘Sg’,‘Fem’};
ARGDEPREL, the label of the syntactic dependency of the argument to its head, e.g. ‘SB’;
LEFTWORD, the surface form of the leftmost dependent of the argument, e.g. ‘die’;
LEFTPOS, the POS tag of the leftmost dependent of the argument, e.g. ‘PWAV’;
LEFTFEATS, the set of FEATS of the leftmost dependent of the argument, e.g. {};
RIGHTWORD, the surface form of the rightmost dependent of the argument, e.g. *null*;
RIGHTPOS, the POS tag of the rightmost dependent of the argument, e.g. *null*;
RIGHTFEATS, the set of FEATS of the rightmost dependent of the argument, e.g. {};
LEFTSIBLINGWORD, the surface form of the left sibling of the argument, e.g. ‘Wie’;

LEFTSIBLINGPOS, the POS tag of the left sibling of the argument, e.g. 'PWAV';
LEFTSIBLINGFEATS, the set of FEATS of the left sibling of the argument, e.g. {};
RIGHTSIBLINGWORD, the surface form of the right sibling of the argument, e.g. *null*;
RIGHTSIBLINGPOS, the POS tag of the right sibling of the argument, e.g. *null*;
RIGHTSIBLINGFEATS, the set of FEATS of the right sibling of the argument, e.g. {}.

Path Features

The features we have introduced so far, extract information based on the predicate *or* argument, exclusively. In order to capture the intra-sentential relation between the predicate and an argument, we consider the *path* in the syntactic dependency tree from the predicate to the argument. The examples given below are taken following the path from *festnehmen* to *Polizei*.

DEPRELPATH, the concatenation of dependency labels and link direction when moving from predicate to argument '$OC \uparrow MO \downarrow SB \downarrow$';
POSPATH, the concatenation of POS tags and link direction when moving from predicate to argument '$VVPP \uparrow VAFIN \downarrow VVFIN \downarrow NN$'.

Position

We defined one additional feature, POSITION, that denoted the position of the argument with respect to the predicate. This could be either 'Before', 'On', or 'After'. The value 'On' appears only in a few languages where the predicate could be an argument of itself. This occurs, for instance, with nominal predicates in English.

4.3.2 Constructing the Features

From an engineering perspective, there is a distinction between linguistic features employed by the SRL system and features used by the statistical classifiers: In the context of classifiers, a feature refers to one dimension of the classifier feature space. A *linguistic feature*, on the other hand, is a set of possible values that a certain grammatical, lexical, or syntactical feature can attain. We will refer to this number of values as the *size* of the feature.

We implement each linguistic feature of size *n* by creating *n* binary features for the statistical classifiers, one for each possible value. They are implemented either as *singular* or *set* features.

The singular features can only attain a single value, hence regardless of size, the corresponding classifier features will all be zero-valued except for one. During parsing it may very well happen that the value of a linguistic feature was not seen during training, in which case all the corresponding classifier features will be zero-valued.

The set features, on the other hand, may have several positive corresponding classifier features. For each string in the set, the corresponding classifier feature was set positive.

All CHILDSET- features were implemented as set features. Moreover, all -FEATS features were also implemented as sets. As described in Sect. 3.2, and seen in Fig. 4.2, FEATS is just

a string of morphological information separated by |. This string was transformed to a set by splitting the string into parts with respect to | and each value was considered a member of the FEATS-set.

4.3.3 Constructing Pairs

In order to improve the separability of the linear classifiers, pairs of features were constructed. A pair of linguistic features was created by taking the Cartesian product between the classifier features of each of the linguistic features. This works well for singular features and pairs between singular features and set features. Pairs between CHILDSET- features were constructed by concatenating the values of each set on each child. We refrained from constructing pairs of involving -FEATS. Partly because it is not always obvious how to pair them in a meaningful way, but also to lower the number of features in the search space during the feature selection.

4.3.4 Feature Selection

Following Johansson (2008), we selected the features used by the classifiers from a pool of possible features using the *greedy forward* procedure shown in Algorithm 1. The first part of the algorithm, greedily selects the feature that increases the μ the most until no feature remaining increases the score more than the threshold ε, while the second part of the algorithm adds feature pairs in the same manner.

We intended to carry out a cross-validation search. However, due to the lack of time, we resorted to using 80% of the training set for training and 20% for evaluating the features. The metric μ used was the PredF_1 in the PD step, the unlabeled ArgF_1 in AI step, and the labeled ArgF_1 in the AC step. The threshold ε was set to 0.01 in all steps.

The feature selection turned out to be a consuming task and in most cases we were forced to interrupt the selection process after a few feature pairs in order to have our system ready in time. In total, we spent three to four weeks running this algorithm to construct feature sets for all languages.

Algorithm 1 Greedy forward feature selection. from Johansson (2008)

function GREEDY-FORWARD-SELECTION(F, μ, ε)
input Feature set F, evaluation metric μ, tolerance ε

```
S ← ∅
P ← ∅
μ_max ← −∞
repeat
    μ_curr ← μ_max
    for each f ∈ F \ S
        μ' ← μ(S ∪ {f}, P)
        if μ' > μ_max + ε
            μ_max ← μ'
            f_max ← f
    if μ_max > μ_curr
        S ← S ∪ {f_max}
until μ_max = μ_curr
repeat
    μ_curr ← μ_max
    for each p ∈ (F × F) \ P
        μ' ← μ(S, P ∪ {p})
        if μ' > μ_max + ε
            μ_max ← μ'
            p_max ← p
    if μ_max > μ_curr
        P ← P ∪ {p_max}
until μ_max = μ_curr
return ⟨S, P⟩
```

Chapter 5

Introducing a Global Model

Intuitively, a predicate-argument proposition is a complete structure where arguments and their labels interdepend. Since our pipeline of classifiers executes each step sequentially and independently with respect to each argument, we fail to catch this structure. The remedy is to generate entire propositions and exploiting their complete structure when selecting the best one.

Since the number of possible propositions of a predicate is exponential in both the length of its sentence, as well as the number of labels, we can not consider every possible proposition. We therefore use a method that produces a limited number of *candidate* propositions from which the final proposition is selected.

The method by which the candidate propositions are generated is described in Sect. 5.1. We describe how these propositions are reranked by an additional classifier in Sections 5.2 and 5.3. In Sections 5.4 and 5.6 we show how the pipeline and the reranker was combined to select the final proposition. In Sect. 5.5 we discuss how to evaluate the upper bound to this approach.

5.1 Generating Candidate Propositions

Following Toutanova et al. (2008), we explore the possibilities to generate a limited number of possible, yet likely, propositions. In order to generate a pool of propositions for a given predicate, the pipeline system is extended. Instead of letting each classifier output the most likely partial proposition, we use the output probabilities associated with each different outcome. Assuming high probabilities correspond to likely (partial) propositions, it is then possible to select a few likely outputs in each step.

In the first step, we have one or many choices of sense to assign the predicate. However, we only consider the most likely sense as assigned by the PD module. By multiplying the probabilities output by the AI classifier on each token, we get a probability for the resulting unlabeled proposition. We then select the k best unlabeled propositions according to their score (i.e. probability). We denote the probability of a complete unlabeled proposition according to AI module P_{AI}. This procedure is commonly referred to as *beam search*, and the variable k is referred to as the beam *width*.

As each token in the sentence is considered an argument of the predicate, P_{AI} is the product of n factors, where n denotes the length of the sentence. The k best propositions will be quite

similar, though not identical, keeping the most likely argument dependencies, and alternating the less likely ones (as well as proposing new argument dependencies that were ruled out by a small margin by the greedy baseline choice).

Similarly, the AC classifier outputs probabilities for each label on an argument dependency. For each unlabeled proposition, we generated the l most likely labellings using these probabilities. We compute the probability of each labeling, P_{AC}, by taking the product of the probabilities of the labels of each argument dependency. Unless one of the unlabeled propositions is empty, this yields a total of $N = k \times l$ labeled proposition. If there is an empty proposition among the k unlabeled propositions, this can not be labeled by the AC module at all, and we get a total of $N = (k-1) \times l + 1$ labeled propositions.

As opposed to the probability of the unlabeled proposition, P_{AI}, which is always a product of the same number of factors, the number of factors that form the P_{AC} probability depends on the number of argument dependencies in the proposition. Since different unlabeled propositions may have a varying number of argument dependencies, the AC probabilities of different propositions may not be comparable in a fair way, i.e. propositions with more dependencies are likely to have a lower probability than propositions with fewer dependencies. To resolve this, we normalized the AC probabilities with respect to the number of argument dependencies in the proposition in the following manner

$$P'_{AC} = (P_{AC})^{1/a},$$

where a is the number of argument dependencies in the proposition. Using the probabilities from the AI and AC steps, we defined the complete probability according to the pipeline as

$$P_{Local} = P_{AI} \times P'_{AC}.$$

The top N propositions are reranked by a separate classifier (described in the next section), and the output probabilities of the pipeline and the reranker are combined (described in Sect. 5.4) in a last step to yield the final output. Figure 5.1 gives an overview of the architecture of the system.

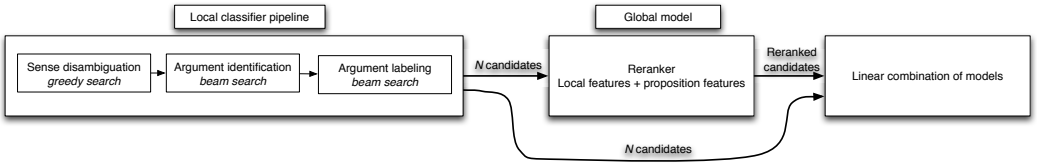

Figure 5.1: System architecture.

5.2 The Global Reranker

Given a pool of candidate propositions, we need a way to select the best one. The straightforward way to do this is to pick the top scoring proposition according to the pipeline, P_{Local}. However,

this still only considers the local choices that the classifiers make and often yield the same result as the baseline system. Moreover, it tends to be biased towards propositions with fewer arguments because this usually yields a higher P'_{AC} despite normalization.

A slightly more sophisticated approach is to train another statistical classifier that can make the choice for us. This binary classifier, called the *reranker*, is applied on all propositions in the candidate pool and gives a probability, $P_{Reranker}$, on each proposition. It is, however, not trivial to generate good training instances for such reranker. Its purpose is, after all, to consider several rather likely propositions and select the proper one.

In order to generate training instances for the reranker, n AI and AC classifiers were trained by partitioning the training set in n parts and using $n-1$ of these parts for each AI and AC classifier, respectively. These AI and AC classifiers were used in the pipeline to generate the top m propositions for each predicate on the part of the corpus they were *not* trained on. If the correct proposition was found in the candidate pool it was marked as a positive instance, and the rest negative.

If the correct proposition was not in the candidate pool, the top scoring (according to the ArgF_1 measure) propositions of the pool were considered positive instances, and the rest negative. Moreover, the correct proposition from the gold standard was also inserted as an *additional* positive instance.

5.3 The Reranker Feature Space

For the reranker, we used all the features from the local AI and AC classifiers combined with an additional global feature that takes the complete predicate-argument structure into consideration. The feature space was created by simply concatenating the feature spaces of each local classifier. Furthermore, for each argument dependency label the AC feature space was concatenated, each repetition corresponding to a certain label, and only invoked for argument dependencies with that particular label.

The values of the reranker classifier features are computed as the sum of the values of each local feature for all the arguments of the predicate-argument proposition. Contrary to the local classifiers, this implies the reranker classifier features may take any non-negative integer value, and not only binary values.

For instance, consider the *hand* proposition from the introductory example (Fig. 1.1). It has three arguments, *Warren*, *him*, and *box*. The feature ARGDEPREL is used by the pipeline in both the AI and AC steps. In the reranker, this feature is included once for AI, and once for each semantic dependency label. The arguments evaluate to the following feature values

- *Warren* - {'SBJ-AI', 'SBJ-A0'};
- *him* - {'OBJ-AI', 'OBJ-A2'};
- *box* - {'OBJ-AI', 'OBJ-A1'};

where the suffixes denote which copy of the ARGDEPREL feature to invoke. Thus, this feature obtains the values (SBJ-AI,1) and (OBJ-AI,2) for the copy imported from the AI classifier. From

the AC classifier, the ARGDEPREL feature is included once for each label. Most of them will be nullvalued since only a few semantic dependency labels are used in this proposition. Only the three copies that correspond to the labels A0, A1, and A2 will be nonnull, with values (SBJ-A0,1), (OBJ-A1,1), and (OBJ-A2,1), respectively.

5.3.1 Global Features

We experimented with various global features but eventually settled with using only the (core) *argument label sequence* (ALS). This feature is a string defined by concatenating the (core) argument labels and the predicate in a sequential order with respect to the sentence. For instance the ALS of the predicate *hand.01* in Fig. 1.1 would be 'A0+hand.01+A2+A1'. When this feature was introduced by Toutanova et al. (2005), it only included core argument labels. The underlying assumption is that core arguments fairly often come in one or a few certain orders with respect to the predicate, whereas the placement of adjuncts in a sentence is less strict.

Core argument labels are explicitly defined in the annotations of Catalan, Chinese, English and Spanish. In the other languages, however, no such distinction is made. For that reason, we considered different sets of labels for the ALS for different languages, in

- Catalan and Spanish all labels prefixed by arg0, arg1, arg2, or arg3 were considered;
- Chinese and English only the labels A0 through A4 were considered;
- Czech, German, and Japanese all argument labels were considered.

5.3.2 Using Voice

In Sect. 2.1.2, we mentioned the notion of grammatical voice and gave an example of how it reverses the order of arguments with respect to the predicate. Since the ALS feature is meant to catch this ordering of arguments, this could potentially confuse the reranker if there are training instances of the same predicate but with different voice. To cope with this, Toutanova et al. (2008) included the voice of each predicate in the ALS, e.g. the handing in Fig. 1.1 could be rephrased *He was handed the metal box by Warren*, yielding the ALS 'A2+hand.01/Passive+A1+A0'.

As voice is a language dependent construct, and not provided in the annotation of the corpora (Czech excluded), this has to be deduced using hand-crafted rules. To the best of our knowledge in each language, we constructed such rules in Catalan, English, German and Spanish and included this in the ALS. In Czech, we obtained the voice directly from the FEATS column.

During preliminary experiments, however, we only observed tiny improvements or even a decrease in performance. For this reason, as well as the general idea of keeping the system language independent, we decided not to use voice in the ALS. Moreover, the hand-crafted rules used to deduce voice are based on the lexical and syntactic properties of the predicate. We believe the reranker classifier is able to identify the same properties by means of the features from the local classifiers, probably even more precisely than our hand-crafted rules.

5.4 Weighting the Models

The reranker does not always make the best choice of candidate proposition. In fact, when we selected the candidate proposition based on the output probabilities of the reranker, the system performed worse than the baseline. We addressed this by trying to combine the local and the reranking models in the following manner

$$P_{Final} = P'_{Local} \times (P_{Reranker})^{\alpha},$$

where α is a parameter that can be used to assign more or less influence to the reranker model. P'_{Local} denotes the normalized probability of the candidate propositions, i.e. we divided each local probability by the sum of all local probabilities. Consequently, the normalized probabilities over all propositions sum up to 1. This procedure rescales the local probabilities and assigns a *very small* local probability to propositions that the local classifiers consider highly unlikely.

5.5 Reranker Potential

Although the correct proposition may not be in the pool of candidates, we would expect a significant increase in performance if the reranker would always make the best choice. By peeking at the gold standard annotation and ignoring the reranker, we can select the optimal candidate proposition from the pool. Obviously this is not an option during real parsing, however, it gives us an opportunity to check the potential of the reranker, i.e. finding an upper bound on the score with respect to a given beam width.

The upper bound is of interest when adjusting the beam widths used by the local classifiers. A wider beam yields a larger pool of candidates, increasing the upper bound and the chance that the correct proposition is in the pool. Increasing the beam width, however, comes at the cost of increased training and parsing time. We will get back to this in Sect. 7.2 when we review the upper bound of the reranker.

5.6 Tuning the Parameters

We conducted several experiments using various beam widths. Unfortunately, we were running short on time, and were not able to investigate this exhaustively. Eventually we settled with setting the beams to $m = l = 4$, yielding candidate pools of size $N = 16$ or $N = 13$ (cf. Sect. 5.1). During these experiments, we set the weight parameter $\alpha = 1$.

When the beam widths were set, we adjusted α to see if further improvement was possible. Surprisingly, in every language we achieved the best results with α equal, or very close, to 1. As we did not have the time to carry out a cross-validation to verify these slight improvements, we let $\alpha = 1$ in all languages.

Chapter 6

Results

In this chapter, we describe the results of our system. We begin by reviewing the baseline results in Sect. 6.1, then go on and describe the performance of the reranker in Sect. 6.2. In Sect. 6.3 we review our performance in the CoNLL 2009 Shared Task and compare it with the other participants.

Evaluation Measures

The measure we have used are more thoroughly defined in Sect. 2.4. To recapitulate, these measures are the

- Labeled attachment score (LAS) – measures the validity of the dependency graphs used as input;
- Labeled semantic F_1 ($SemF_1$) – measures the validity of the complete predicate-argument proposition;
- Labeled predicate F_1 ($PredF_1$) – same as $SemF_1$, but takes only predicates into consideration;
- Labeled argument F_1 ($ArgF_1$) – same as $SemF_1$, but takes only arguments into consideration.

The LAS and $SemF_1$ are the major metrics used to evaluate syntactic and semantic dependencies, respectively. LAS is interesting as it gives an idea of the quality of the dependency graphs used for input. The $SemF_1$ gives a joint score on complete propositions and denotes the overall performance of our system. These scores were obtained using the official CoNLL 2009 Shared Task evaluation script[1]. It gives LAS and $SemF_1$ directly, as well as all the figures required to deduce the $PredF_1$ and $ArgF_1$ measures.

We introduce the $PredF_1$ and $ArgF_1$ measures to get a better idea on how the performance of different components of the system. Moreover, as mentioned in Sect. 3.2, in our input corpora predicates were already identified. This results in a predicate recall of 100%, and essentially

[1] Available at the Shared Task website at http://ufal.mff.cuni.cz/conll2009-st/.

reduces the problem to labeling the predicates. Additionally, predicate disambiguation can be considered a relatively simple problem as the number of different predicate senses is usually quite low (usually only one or two per lemma, ranging up to about half a dozen). Consequently, we expected (and achieved) higher $\text{Pred}F_1$ scores than $\text{Arg}F_1$ scores.

Although we can expect high $\text{Pred}F_1$ figures, we still think the joint $\text{Sem}F_1$ measure is suitable because it scores predicates and arguments equally. Since the argument-predicate ratio (APR) is larger than one (Czech excluded), the joint measure will be biased towards identification and classification of arguments. Moreover, since the measure is computed as a harmonic mean between precision and recall, poor performance with respect either to predicates or arguments, is enough to penalize the joint score sufficiently.

However as the APR varies, the influence of predicate disambiguation vs. argument identification and classification in the $\text{Sem}F_1$ score varies as well. Since the reranker is only able to improve argument identification and labeling, the $\text{Arg}F_1$ measure gives a more fair estimate on the improvement of the reranker in each language.

6.1 Baseline Results

The performance of the baseline system on the official test set is given in Table 6.1. The first column gives the LAS of the testing corpora that were used for input. As mentioned in Sect. 3.2 these dependencies were generated by the Shared Task organizers using freely available dependency parsers. The LAS scores remain the same in each language throughout all experiments described below, and are omitted from all other tables. The remaining three columns denote the performance of our system using the three measures $\text{Pred}F_1$, $\text{Arg}F_1$, and $\text{Sem}F_1$.

	LAS	$\text{Pred}F_1$	$\text{Arg}F_1$	$\text{Sem}F_1$
Catalan	86.13	87.20	76.13	79.54
Chinese	78.46	94.92	69.83	77.84
Czech	78.46	94.20	74.24	84.99
English	85.50	95.59	79.29	84.44
German	85.93	81.45	77.44	79.01
Japanese	91.12	99.07	60.26	75.61
Spanish	86.20	85.43	76.56	79.28
Average	84.54	91.12	73.39	80.10

Table 6.1: Baseline results. LAS denotes the syntactic accuracy of input, $\text{Pred}F_1$ and $\text{Arg}F_1$, the labeled semantic F_1 using predicates and arguments only, respectively, and $\text{Sem}F_1$ the joint labeled Semantic F1 measure.

We achieved an average $\text{Sem}F_1$ above 80%, which we believe to be very good. The expected difference between $\text{Pred}F_1$ and $\text{Arg}F_1$ was also verified, yet the absolute difference between these scores and the $\text{Sem}F_1$ score are leaning towards the $\text{Arg}F_1$ measure (again, Czech excluded), also in accordance with our expectations.

The Japanese PredF_1 score is close to 100%, which is explained by the fact that Japanese lacks ambiguous predicates (explained in Sect 3.2). The reason why it is not exactly 100% is caused by errors in the predicted lemmas in the provided corpus. The other PredF_1 scores roughly fall into two groups: Catalan, German, and Spanish are significantly lower than Chinese, Czech, and English. We believe this is a consequence of the differences in size in the training corpora – The larger training corpora, the more training examples for each subclassifier.

With respect to the ArgF_1 measure, Japanese also turns out to be in the extreme, the 60.26% is indeed very low in comparison to the other languages. Again, we believe its reasonable to expect this result as Japanese has the smallest training corpus of all languages, yet a very large set of possible argument labels.

Not surprisingly, English receives the highest ArgF_1 score. The vast training corpus as well as splitting the classifiers between nominal and verbal predicates justifies this. Moreover, as the feature templates we used were originally defined and tested out on English, it is reasonable to assume they do well there, but not necessarily as well in other languages.

Considering the syntactic dependencies provided by the organizers, they are rather good, but slightly lower than state-of-the-art in each language (Nivre et al., 2007; Surdeanu et al., 2008). The high figure in Japanese is explained by the low number of dependency labels (Only five labels are used), as well as the typographic tokenization of the Japanese "bunsetsu" phrase style where most words are attached to the following one using the most common label. This also causes the DEPREL-related feature templates used to be less powerful during classification. In fact, using features similar to ours and a dependency-based syntax, Johansson and Nugues (2008b) has shown that a richer set of dependency labels improves semantic role labeling in English.

Excluding Japanese, Chinese and Czech have the worst ArgF_1 scores. This correlates well with the poor syntactic dependencies provided.

6.2 Applying the Reranker

In Chapter 5, we described how to extend the baseline system using a global reranker. The reranker results, as well as baseline results and the gain in absolute difference is given in Table 6.2. As previously stated, the SemF_1 score is biased differently with respect to predicates and arguments in different languages. For that reason, we present the results of the reranker using both the ArgF_1 and SemF_1 measures. Because the reranker does not improve predicate disambiguation, the PredF_1 figures remain unaffected and these figures are omitted.

Although the improvement of the reranker is quite modest, it gives an increase in every language. In fact, the English 1.19 point gain in SemF_1 is better than the improvement achieved by top performing systems in the CoNLL 2008 Shared Task that also employed a reranking approach (Johansson and Nugues, 2008a; Ciaramita et al., 2008). The ArgF_1 measure allows for interlingual comparison without the bias of the varying APR. We achieved the best gain in English, followed by German. Not surprisingly, we achieved the lowest gain in Czech and Japanese. This was somewhat expected as we included all the argument labels in the ALS in these languages. However, we also included all labels in German, and received betters gains than in Czech and Japanese. The difference is explained by the small number (only 10, shown in Table 3.1) of labels in this annotation. Moreover, Czech and Japanese also had the tricky double dependencies

	ArgF$_1$			**SemF$_1$**		
	Baseline	Reranker	Gain	Baseline	Reranker	Gain
Catalan	76.13	76.76	0.63	79.54	80.01	0.47
Chinese	69.83	70.87	1.04	77.84	78.60	0.76
Czech	74.24	74.77	0.53	84.99	85.41	0.42
English	79.29	81.00	1.71	84.44	85.63	1.19
German	77.44	78.79	1.35	79.01	79.71	0.70
Japanese	60.26	61.03	0.77	75.61	76.30	0.69
Spanish	76.56	77.46	0.90	79.28	79.91	0.63
Average	73.39	74.38	0.99	80.10	80.80	0.70

Table 6.2: Results and improvement by reranker. Scores are given using both the ArgF$_1$ and SemF$_1$ measures.

(mentioned in Sect. 3.2.1) that were not handled in any special way by our system.

6.3 The CoNLL 2009 Shared Task Evaluation

As previously mentioned, participants of the CoNLL 2009 Shared Task could choose between parsing syntactic and semantic dependencies (joint), or just the semantic dependencies (SRL-only). In Table 6.3 we list the SemF$_1$ scores of all participants in the closed task. All figures in the table are taken from the official Shared Task description paper (Hajič et al., 2009). All participating systems, except for two, submitted a paper describing their system. We have replaced the team names with the affiliation of the primary author of each system. Our results are listed on the second row under the name *Lund*.

Unfortunately, for the submission of our test results to the Shared Task, the Spanish reranker was trained improperly and yielded significantly lower results. This was exclusively caused by human error and resolved simply by retraining the reranker without any changes in the implementation. However, this mistake did not come to our attention until after the submission of test runs, and we received a lower score in the official evaluation. Aside from Spanish, our scores are identical to the ones shown in Table 6.2.

Our system received the best scores in Chinese and German, and ranked second among 20 participants using the average SemF$_1$ measure. Despite the bug in Spanish, our average still came out greater than baseline average (80.10%). On the other hand, this bug also appears to have cost us the top position.

	Team	Average	Catalan	Chinese	Czech	English	German	Japanese	Spanish
1	Hong Kong	80.47	**80.32**	77.72	85.19	85.44	75.99	78.15	**80.46**
2	Lund†	80.31	80.01	**78.60**	85.41	85.63	**79.71**	76.30	76.52
3	Hong Kong	79.96	80.10	76.77	82.04	**86.15**	76.19	78.17	80.29
4	Harbin	79.94	77.10	77.15	**86.51**	85.51	78.61	**78.26**	76.47
5	Geneva	78.42	77.44	76.05	86.02	83.24	71.78	77.23	77.19
6	Edinburgh	77.46	78.00	77.73	75.75	83.34	73.52	76.00	77.91
7	Berkeley	76.00	74.53	75.29	79.02	80.39	75.72	72.76	74.31
8	NIST	75.65	72.35	74.17	84.69	84.26	63.66	77.93	72.50
9	Brown◇	72.85	72.18	72.43	78.02	80.43	73.40	61.57	71.95
10	Hefei	70.78	66.34	71.57	75.50	78.93	67.43	71.02	64.64
11	DFKI	70.31	67.34	73.20	78.28	77.85	62.95	64.71	67.81
12	Harbin	69.72	66.95	67.06	79.08	77.17	61.98	69.58	66.23
13	Dublin	69.26	74.06	70.37	57.46	69.63	67.76	72.03	73.54
14	Antwerp	68.95	70.14	66.71	71.49	75.97	61.01	68.82	68.48
15	Paris	66.49	65.60	67.37	71.74	72.14	66.50	57.75	64.33
16	Barcelona	63.06	46.79	59.72	76.90	75.86	62.66	71.60	47.88
17	Stockholm	61.27	57.11	63.41	71.05	67.64	53.42	54.74	61.51
18	Lin◇	57.18	61.70	70.33	60.43	65.66	59.51	23.78	58.87
19	Wuhan	56.69	41.00	72.58	62.82	67.56	54.31	58.73	39.80
20	Prague	32.14	24.19	34.71	58.13	36.05	16.44	30.13	25.36

Table 6.3: Official SemF_1 scores for all participants in the closed task. Team names have been replaced with the affiliation of the primary authors of each system, except names marked with ◇, where this information was not available. † denotes our team. Bold figures denote the best score in each language.

Chapter 7

Discussion

In this final chapter we summarize our work and discuss some ideas that might be of particular interest for continued research.

7.1 Conclusion

In this thesis, we proposed a method to construct a language independent semantic role labeler that works on top of a dependency based syntactic representation of the input text. The system uses supervised learning methods and requires annotated corpora to generate training data. Provided corpora annotated with lemma, POS, and syntactic dependency trees (optionally also FEATS), the system is trained and run without requiring any further knowledge of the language on which it is working.

A core system consisting of a pipeline of classifiers that sequentially performs the tasks of predicate sense disambiguation, argument identification, and argument classification has been presented. Suitable feature sets used by the classifiers in executing these subtasks have been tailored by an automatic procedure for each training corpus.

The proposed system was implemented and has been trained and run on seven languages. The CoNLL 2009 Shared Task evaluation shows that our system sets the state of the art in Chinese and German, and is not far behind other systems in the other languages.

We have also shown that the baseline system can be further improved by means of a global model. The system gave good scores in all languages with an overall average score outperforming that of any other system that we are aware of.

7.2 Reranker Potential

The upper bound of the reranker on the test set is given in Table 7.1. They were acquired using the gold standard (described in Sect. 5.5) and the same beam widths as we did during training and parsing ($k = l = 4$).

Compared to the performance of our reranker, we see that there is room for substantial improvements to the reranking model. Except for one global feature, all the the features used by the reranker are the same as the local AI and AC features. All local features were selected to optimize

	ArgF$_1$			**SemF$_1$**		
	Baseline	Reranker	Upper Bound	Baseline	Reranker	Upper Bound
Catalan	76.13	76.76	91.45	79.54	80.01	90.11
Chinese	69.83	70.87	85.80	77.84	78.60	88.70
Czech	74.24	74.77	90.60	84.99	85.41	92.55
English	79.29	81.00	92.97	84.44	85.63	93.80
German	77.44	78.79	92.62	79.01	79.71	88.78
Japanese	60.26	61.03	84.35	75.61	76.30	90.04
Spanish	76.56	77.46	90.79	79.28	79.91	89.12
Average	73.39	74.38	89.80	80.10	80.80	90.44

Table 7.1: Reranker on test set. Beam widths set to $k = l = 4$. Scores are given using both the ArgF$_1$ and SemF$_1$ measures.

the performance of the local classifiers. Performing a separate feature selection for the reranker, perhaps over an extended pool of features, would therefore most likely improve its performance.

More generally, we believe that the overall design of the reranker feature space should be reconsidered. One could argue that the features of the local classifiers should not be included in the reranker, as they have already contributed through the local classifiers. Moreover, because of the vastness of the reranker feature space, it requires large amounts of memory and takes very long time to train and apply during parsing.

7.3 Further Work

The system has been designed and developed under hard time constraints in a way that allowed easy debugging and trying out new ideas. Basically all constituents of the system, as well as their interaction, could be improved from an engineering point of view. Furthermore, the possibilities to extend the system are also close to endless. Other than the already mentioned ideas for the reranker, we propose two more ideas that we believe are worth investigating.

Cross-validated Feature Selection

The features selected by the selection procedure was obtained by using 80% of the training set for training and 20% for evaluating the features. Although this gave good results, a cross-validation search would have been able to generate the best feature set based on the entire training corpus. This would also reduce the risk of overfitting to the partition of the training set that was used for evaluation during feature selection.

Argument Pruning

The AI module considers every token of a sentence as an argument of each predicate. Several tokens can easily be ruled out by their lexical form or path to the predicate in the dependency

tree. Reducing the set of potential arguments in a sentence, commonly referred to as *argument pruning*, would improve speed and possibly also reduce errors. Although there is a risk of ruling out actual arguments in the pruning process, this strategy deserves to be explored. This would also reduce the large number of negative training instances for the AI classifier, possibly resulting in improved accuracy.

Appendix A

Features Used

Table A.1: Feature sets for predicate disambiguation.

	cat	**chi**	**cze**	**eng**	**ger**	**spa**
PredWord	•		•			•
PredPOS	•			•		
PredDeprel		•	•	•		
PredFeats			•		•	•
PredParentWord	•	•	•	•	•	
PredParentPOS		•		•	•	
PredParentFeats			•		•	
DepSubCat	•	•	•	•	•	
ChildDepSet	•	•	•	•	•	•
ChildWordSet	•	•	•	•	•	•
ChildPOSSet		•	•	•	•	•
ChildDepSet+ChildPOSSet	•			•		
ChildDepSet+ChildWordSet				•		
ChildDepSet+PredPOS						•
ChildWordSet+DepSubCat		•				
ChildWordSet+PredDeprel		•				
ChildWordSet+PredParentPOS		•		•		
ChildWordSet+PredParentWord		•		•		
ChildWordSet+PredPOS				•		
DepSubCat+PredParentWord		•		•		
DepSubCat+PredFeats					•	
DepSubCat+PredParentFeats					•	
PredDeprel+PredParentWord		•				
PredParentPOS+PredParentWord		•				
PredPOS+PredParentWord		•		•		

Table A.2: Singular features used for Argument Identification. V = Verb Predicate, N = Noun Predicate

	cat	**chi**	**cze**	**eng**	**ger**	**jap**	**spa**
PredWord							
PredPOS				N	•		
PredLemma				N	•		
PredDeprel							
PredLemmaSense		•	•	V	•		
PredFeats			•				
PredParentWord				V			
PredParentPOS				V			
PredParentFeats			•				
DepSubCat	•	•					
ChildDepSet	•				•		
ChildWordSet				N			
ChildPOSSet		•					
ArgWord	•	•		N,V	•	•	•
ArgPOS	•	•		N,V	•	•	•
ArgFeats	•						
ArgDeprel	•	•	•	V	•		•
LeftWord	•				•		
LeftPOS						•	
LeftFeats						•	•
RightWord	•			N			
RightPOS				N			
RightFeats							
LeftSiblingWord	•						•
LeftSiblingPOS					•	•	
LeftSiblingFeats			•				
RightSiblingWord	•	•		V	•	•	•
RightSiblingPOS							
RightSiblingFeats							
DeprelPath	•	•	•	N,V	•	•	•
POSPath	•	•	•	N,V	•	•	•
Position			•	N,V		•	

Table A.3: Feature pairs used for Argument Identification. V = Verb Predicate, N = Noun Predicate

	cat	**chi**	**cze**	**eng**	**ger**	**jap**	**spa**
ArgDeprel+ArgPOS				V			
ArgDeprel+ChildDepSet				V			
ArgDeprel+POSPath			•				
ArgDeprel+DeprelPath				V			
ArgDeprel+Position		•					
ArgDeprel+PredDeprel				V			
ArgDeprel+PredParentWord				N			
ArgFeats+LeftPOS					•		
ArgFeats+POSPath						•	
ArgFeats+PredPOS						•	
ArgPOS+ChildPOSSet						•	
ArgPOS+POSPath			•				
ArgPOS+PredLemmaSense		•		N	•		
ArgPOS+RightSiblingPOS					•		
ArgWord+DepSubCat	•						
ArgWord+Position				N			
ArgWord+PredLemma					•		
ArgWord+PredLemmaSense					•		
ChildDepSet+POSPath		•			•		
ChildPOSSet+PredDeprel					•		
ChildWordSet+DeprelPath						•	
ChildWordSet+PredLemmaSense				N			
DeprelPath+Position	•	•		N			•
DeprelPath+PredFeats						•	
DepSubCat+DeprelPath							•
LeftFeats+RightSiblingFeats	•						
LeftFeats+PredParentPOS					•		
LeftPOS+RightPOS				V			
LeftPOS+PredWord				N			
LeftWord+POSPath							•
Position+PredLemmaSense				N			
Position+PredParentPOS						•	
POSPath+RightSiblingFeats							•
POSPath+RightSiblingPOS				V			

Table A.4: Singular features used for Argument Classification. V = Verb Predicate, N = Noun Predicate

	cat	**chi**	**cze**	**eng**	**ger**	**jap**	**spa**
PredWord		•		N		•	
PredPOS		•		V			
PredLemma	•	•	•	N,V	•		•
PredDeprel							
Sense	•	•	•	N,V	•	•	•
PredFeats			•		•		
PredParentWord		•		V		•	
PredParentPOS				V	•		
PredParentFeats							
DepSubCat							
ChildDepSet	•	•		V	•	•	•
ChildWordSet			•			•	
ChildPOSSet		•		N		•	
ArgWord	•	•	•	N,V	•	•	•
ArgPOS	•	•	•	N,V		•	
ArgFeats	•		•		•		•
ArgDeprel	•	•	•	V	•		•
DeprelPath	•	•		V	•		
POSPath			•	V	•	•	
Position	•	•	•	N,V		•	•
LeftWord	•	•		N		•	•
LeftPOS		•		V			
LeftFeats					•		
RightWord	•	•		N,V			•
RightPOS	•	•		N,V			•
RightFeats	•						•
LeftSiblingWord	•	•		N		•	
LeftSiblingPOS	•	•		N,V		•	
LeftSiblingFeats	•				•		
RightSiblingWord		•			•		•
RightSiblingPOS					•	•	
RightSiblingFeats						•	

Table A.5: Feature pairs used for Argument Classification. V = Verb Predicate, N = Noun Predicate

	cat	**chi**	**cze**	**eng**	**ger**	**jap**	**spa**
ArgDeprel+ArgPOS				V			
ArgDeprel+ChildDepSet	•						
ArgDeprel+ChildPOSSet					•		
ArgDeprel+POSPath					•		
ArgDeprel+RightPOS				V			
ArgDeprel+Sense				V	•		
ArgFeats+ArgWord						•	
ArgFeats+Sense					•		
ArgPOS+ArgWord				V			
ArgPOS+Position				N			
ArgPOS+PredLemma		•		V			•
ArgPOS+Sense				N	•		
ArgWord+ChildDepSet							•
ArgWord+Position		•		N			
ArgWord+PredLemma					•		
ArgWord+RightWord							•
ArgWord+Sense				N,V	•		
ChildDepSet+Position				V			
ChildWordSet+Position	•						
ChildWordSet+DeprelPath						•	
ChildWordSet+POSPath						•	
DeprelPath+PredParentWord					•		
LeftPOS+RightSiblingPOS				N			
LeftSiblingPOS+Sense				N			
LeftSiblingPOS+RightSiblingPOS					•		
LeftWord+Sense					•		
PredFeats+Sense	•						•
PredPOS+PredParentWord					•		
Position+Sense		•		N,V	•		
POSPath+PredLemma						•	
POSPath+Sense				N			
RightPOS+RightSiblingPOS		•			•		
Sense+RightSiblingPOS				N	•		

Bibliography

Baker, C. F., Fillmore, C. J., and Lowe, J. B. (1998). The Berkeley FrameNet Project. In *Proceedings of COLING/ACL-1998*.

Bornkessel, I., Schlesewsky, M., Comire, B., and Friederici, A. D. (2006). *Semantic Role Universals And Argument Linking: Theoretical, Typological, And Psycholinguistic Perspectives*. Walter de Gruyter.

Burchardt, A., Erk, K., Frank, A., Kowalski, A., Padó, S., and Pinkal, M. (2006). The SALSA corpus: a German corpus resource for lexical semantics. In *Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC-2006)*, Genoa, Italy.

Burges, C. J. (1998). A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2:121–167.

Carreras, X. and Màrquez, L. (2004). Introduction to the CoNLL-2004 shared task: Semantic role labeling. In *Proceedings of CoNLL-2004*.

Carreras, X. and Màrquez, L. (2005). Introduction to the CoNLL-2005 shared task: Semantic role labeling. In *Proceedings of CoNLL-2005*.

Ciaramita, M., Attardi, G., Dell'Orletta, F., and Surdeanu, M. (2008). DeSRL: A linear-time semantic role labeling system. In *Proceedings of the Shared Task Session of CoNLL-2008*.

Cristianini, N. and Shawe-Taylor, J. (2000). *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press.

Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R., and Lin, C.-J. (2008). LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874.

Gildea, D. and Jurafsky, D. (2002). Automatic labeling of semantic roles. *Computational Linguistics*, 28(3):245–288.

Hajič, J., Ciaramita, M., Johansson, R., Kawahara, D., Martí, M. A., Màrquez, L., Meyers, A., Nivre, J., Padó, S., Štěpánek, J., Straňák, P., Surdeanu, M., Xue, N., and Zhang, Y. (2009). The CoNLL-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proceedings of the 13th Conference on Computational Natural Language Learning (CoNLL-2009), June 4-5*, Boulder, Colorado, USA.

Hajič, J., Panevová, J., Hajičová, E., Sgall, P., Pajas, P., Štěpánek, J., Havelka, J., Mikulová, M., and Žabokrtský, Z. (2006). Prague Dependency Treebank 2.0.

Hsu, C.-W. and Lin, C.-J. (2001). A comparison of methods for multi-class support vector machines.

Hudson, R. (1984). *Word Grammar*. Blackwell.

Johansson, R. (2008). *Dependency-based Semantic Analysis of Natural-language Text*. PhD thesis, Lund University.

Johansson, R. and Nugues, P. (2008a). Dependency-based syntactic–semantic analysis with PropBank and NomBank. In *Proceedings of the Shared Task Session of CoNLL-2008*.

Johansson, R. and Nugues, P. (2008b). The effect of syntactic representation on semantic role labeling. In *Proceedings of COLING-2008*.

Kawahara, D., Kurohashi, S., and Hasida, K. (2002). Construction of a Japanese relevance-tagged corpus. In *Proceedings of the 3rd International Conference on Language Resources and Evaluation (LREC-2002)*, pages 2008–2013, Las Palmas, Canary Islands.

Levin, B. (1993). *English Verb Classes and Alternations: A preliminary Ivestigation*. University of Chicago Press.

Lin, C.-J., Weng, R., and Keerthi, S. (2008). Trust region Newton method for large-scale logistic regression. *JMLR*, 2008(9):627–650.

Mel'čuk, I. A. (1988). *Dependency Syntax: Theory and Practice*. State University Press of New York.

Meyers, A., Reeves, R., Macleod, C., Szekely, R., Zielinska, V., Young, B., and Grishman, R. (2004). The NomBank project: An interim report. In Meyers, A., editor, *HLT-NAACL 2004 Workshop: Frontiers in Corpus Annotation*, pages 24–31, Boston, Massachusetts, USA. Association for Computational Linguistics.

Nivre, J., Hall, J., Kübler, S., McDonald, R., Nilsson, J., Riedel, S., and Yuret, D. (2007). The CoNLL 2007 shared task on dependency parsing. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL–2007*.

Palmer, M., Kingsbury, P., and Gildea, D. (2005). The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–106.

Palmer, M. and Xue, N. (2009). Adding semantic roles to the Chinese Treebank. *Natural Language Engineering*, 15(1):143–172.

Punyakanok, V., Roth, D., Yih, W.-t., and Zimak, D. (2004). Semantic role labeling via integer linear programming inference. In *COLING '04: Proceedings of the 20th international conference on Computational Linguistics*, page 1346, Morristown, NJ, USA. Association for Computational Linguistics.

Surdeanu, M., Johansson, R., Meyers, A., Màrquez, L., and Nivre, J. (2008). The CoNLL-2008 shared task on joint parsing of syntactic and semantic dependencies. In *Proceedings of the 12th Conference on Computational Natural Language Learning (CoNLL-2008)*.

Taulé, M., Martí, M. A., and Recasens, M. (2008). AnCora: Multilevel Annotated Corpora for Catalan and Spanish. In *Proceedings of the 6th International Conference on Language Resources and Evaluation (LREC-2008)*, Marrakesh, Morroco.

Tesnière, L. (1959). *Éléments de syntaxe structurale*. Klincksieck.

Toutanova, K., Haghighi, A., and Manning, C. (2005). Joint learning improves semantic role labeling. In *Proceedings of ACL-2005*.

Toutanova, K., Haghighi, A., and Manning, C. D. (2008). A global joint model for semantic role labeling. *Compututational Linguistics*, 34(2):161–191.

Xue, N. and Palmer, M. (2004). Calibrating features for semantic role labeling. In *Proceedings of EMNLP-2004*.