Charles University in Prague
Faculty of Mathematics and Physics
&
University of Malta
Department of Intelligent Computer Systems

Masters Thesis

Pranava Swaroop Madhyastha

Exploring Higher Order Dependency Parsers

Institute of Formal and Applied Linguistics
&
Department of Human Language Science and Technology

Supervisors: Professor Michael Rosner
& RNDr. Daniel Zeman

Study program: Erasmus Mundus Masters in
Language & Communication Technology

Charles University in Prague and University of Malta, Malta

Dedicated sincerely to my Father.

At the outset, I would like to thank my supervisors, Dr. Michael Rosner and RNDr. Daniel Zeman for their guidance, support and extended patience, during this time of the thesis. I also thank the co-ordinators of this program RNDr. Vladislav Kubon, RNDr. Marketa Lopatkova and Dr. Michael Rosner for their continued support and help during the whole two years of this program, without which, this wouldn't have been possible. I offer a special thanks to Dr. Xavier Carreras for motivating me during my stay at UPC in Barcelona.

My gratitude goes to both my parents who have provided me with unconditional love and support for years and have until now received nothing tangible in return, except this lousy dedication.

I would like to acknowledge all my friends, colleagues, professors and every one who have made this a valuable and worthwhile journey.

Prague, August 2011. Pranava Swaroop Madhyastha

Title: Exploring Higher Order Dependency Parsers
Author: Pranava Swaroop Madhyastha
Department: Institute of Formal and Applied Linguistics
Department of Human Language Science and Technology
Supervisor: Professor Michael Rosner& RNDr. Daniel Zeman
Supervisor's e-mail address:mike.rosner@um.edu.mt & zeman@ufal.mff.cuni.cz

Abstract: Most of the recent efficient algorithms for dependency parsing work by factoring the dependency trees. In most of these approaches, the parser loses much of the contextual information during the process of factorization. There have been approaches to build higher order dependency parsers - second order, [Carreras2007] and third order [Koo and Collins2010]. In the thesis, the approach by Koo and Collins should be further exploited in one or more ways. Possible directions of further exploitation include but are not limited to: investigating possibilities of extension of the approach to non-projective parsing; integrating labeled parsing; joining word-senses during the parsing phase [Eisner2000]

Keywords: Dependency Parsing, Discriminative Learning, Higher order, Semantic features.

Název práce: Český název
Autor: Pranava Swaroop Madhyastha
Katedra (ústav): Ústav formální a aplikované lingvistiky
Vedoucí diplomové práce: Professor Michael Rosner& RNDr. Daniel Zeman
E-mail vedoucího: mike.rosner@um.edu.mt & zeman@ufal.mff.cuni.cz

Abstrakt: Český abstrakt (abstract in Czech).

Klíčová slova: česká klíčová slova (keywords in Czech)

Table of contents

Abstract v

List of tables **viii**

List of figures **ix**

1 Introduction **1**

1.1 Dependency Grammar: Definition and Current Status 2
1.2 Dependency Parsing: Status Quo 2
1.3 Research Objective 3

2 The Premise **5**

2.1 Why Parsing? 5
2.2 The Dependency Grammar 6
2.3 Projectivity and Non-Projectivity Constraints 7
2.4 Discriminative Graph Based Dependency Parsing 9
2.5 Notational Conventions 9
2.6 Factoring Structures 10
2.6.1 Parameter Estimation using Structured Perceptron 10
2.7 Parsing Algorithms 12
2.7.1 Second-Order Factored Parsing Algorithm 14
2.7.2 Third-Order Factored Parsing Algorithm 16
2.8 Feature Space 17
2.9 Effect of Features in Dependency Parsing 18
2.9.1 Effect of Semantic Features 19
2.9.2 Morphosyntactic and Morphosemantics with Dependency Parsing 19

3 State-of-the-Art and Current Research 20

4 System Configuration 22
4.1 In Focus 22
4.2 System Information 23
4.2.1 Input Format 23
4.2.2 Corpus Used 24
4.2.3 Tools 25
4.3 Word Sense Extraction 26
4.4 Morphosyntactic Feature Extraction 26

5 Experiments 28
5.1 Experiment 1 28
5.1.1 Research Question 28
5.1.2 Theoretical Plausibility 28
5.1.3 Experimentation 29
5.1.4 Results 30
5.1.5 Discussion 30
5.2 Experiment 2 32
5.2.1 Research Question 32
5.2.2 Theoretical Plausibility 32
5.2.3 Experimentation 32
5.2.4 Results 32
5.2.5 Discussion 33
5.3 Experiment-3 33
5.3.1 Research Question 33
5.3.2 Theoretical Plausibility 33
5.3.3 Experimentation 34
5.3.4 Results 34
5.3.5 Discussion 35

6 Discussion 36

7 Future Work 38

8 Conclusion 40

A An appendix 41

Bibliography 46

List of tables

3.1 CoNLL 2007 Shared Task Results. Unlabeled Attachment Score. The parsers are - [Carreras2007], [Shimizu2007], [Sagae and Tsujii2007], [Nivre et al.2007a], [Titov and Henderson2007]. 20

4.1 Morphological Tagset for Czech 27

5.1 A sample set from the Penn Treebank tagged with fine-grained word-sense tags for English 30
5.2 Semantic POS-Tags for Czech from the Prague Dependency Treebank 31
5.3 Results for Penn Treebank 31
5.4 Results for Prague Dependency Treebank 31
5.5 Results for Penn Treebank 33
5.6 Results for Prague Dependency Treebank 35
5.7 Results for Penn Treebank 35

List of figures

2.1 Basic dependency structure - A head and a modifier 6
2.2 A simple example;please note that we are adding an extra root here . . 7
2.3 The directed graph for previous example 7
2.4 Projective Structure: none of the edges are crossing each other. 8
2.5 Non-Projective Structure: the red edge is basically crossing. 8
2.6 The basic dependency structure. 12
2.7 Sibling structure. 13
2.8 Grandchild Structure 13
2.9 Grandsibling Structure 14
2.10 First-Order dynamic programming structures 14
2.11 Second order dynamic programming structure 15
2.12 Second order dynamic programming structures 16
2.13 Third order dynamic programming structures 17
2.14 third order dynamic programming structures 18

4.1 Example Structure where Sense information is included 23

5.1 Effect of Wordsense 29

A.1 Result for experimentation with fine-grained wordsenses with English 41
A.2 Result for experimentation with fine-grained wordsenses with Czech . 42
A.3 Result for experimentation with coarse-grained wordsense with English 43
A.4 Result for experimentation with morphological tags for Czech 44
A.5 Result for experimentation with morphological tags for English 45

Chapter 1

Introduction

One of the major challenges in the field of computational linguistics is to transform text from the native natural language representation to representations which can be fed as an input to the computer. The computer would then be using this to perform various tasks. The transformation of representations from natural language to well-defined formal languages involves several layers of processing. Among these, parsing is one of the most important and the most difficult of them. Parsing of a natural language can be defined as the process of mapping sentences in the natural language to their syntactic representations. Parsing also lays an important foundation for understanding the natural language syntax and semantics. Recently, statistical parsing has taken precedence over other forms of parsing due to its highly efficient parsing capabilities [Marcus et al.1993]. While parsing accuracy of these parsers are mostly rising, this is still not enough for integration with the practically implementable natural language processing applications and hence there is a pressing need for better accuracy [Merlo et al.2011]. This is also due to the highly ambiguous nature of the natural language. High accuracy natural language parsing would be very useful for modern NLP applications which include machine translation, question answering systems, information extraction, text summarization, semantic role labeling, etc..

The syntactic structure of the natural language is formalized into a certain syntactic representation, this is also known as the grammatical formalisms. There are several syntactic representations which are used in computational linguistics. In this thesis, we would be focusing on one of the most important representations which is based on the notion of dependency [Tesnière1959]. This formalism is now formally known as the "Dependency Grammar" (DG) Framework. Also, we would be concerned with a particular type of parsing for the DG called the data-driven discriminative graph based dependency parsing, also known more frequently as graph based dependency parsing formalism. This is a type of statistical parsing.The rest of the thesis will set the very basic premise of the thesis.

1.1 Dependency Grammar: Definition and Current Status

A dependency tree can be defined in the most basic way as a directed acyclic graph in which all the words in the given sentence are connected together by grammatical relations. For example, the subject and object depend on the main verb; adjectives depend on the nouns that they modify; etc.. In each pair of connected words, one is called the ‘dependent’ which is basically a modifier and the other is called the ‘head’. That is, the modifier modifies the head. Simply, an analysis based on DG can be explained as a tree where, each token in the sentence is a node in the tree, and each arc connects a head to its modifier. A more detailed discussion on the dependency grammar will be made in the following chapter. DG is an increasingly important grammar representation in modern computational linguistics. It is particularly well-suited for languages with approximately free word order [Covington2001]. Also, dependency representations are emerging as the standard for comparing the result of syntactic analysis across different grammar formalisms and parsing approaches. In a way, the DG formalizes the syntactic structure as a directed tree of dependencies. The classical phrase-structure [Chomsky1956] models have been of less help in exploring the joint ‘syntactic and semantic’ phenomena, especially with a cross-linguistic perspective. [Mel’čuk et al.1987] and [Covington2001] claimed that one of the advantages of DG over approaches based on phrase based or constituent structures is that it allows for a more adequate treatment of languages with variable word order, where discontinuous syntactic constructions are more common than in languages like English. Also, note that, dependency links are close to the semantic relationships needed for the next stage of interpretation.

There are two dominant and mostly studied approaches to dependency parsing: *graph-based and transition-based*, where graph-based parsing is understood to be slower but exhaustive, and often more accurate.

DGs have been at the forefront of computational linguistics since last two decades. This can be seen by its application to functional description of grammar [Sgall1984], possibilities of extracting rich lexical information from corpora [Bangalore et al.2009], applications related to semantic graphs [Marneffe et al.2007] and adaptability to various languages with the same formalism [Bourdon et al.1998].

1.2 Dependency Parsing: Status Quo

Highly efficient parsers have made DG to be one of the most explored grammar formalisms in the last decade [Merlo et al.2011]. One of the major hurdles in understanding natural languages is mostly concerned with producing an optimized natural language system. Implementations of efficient grammar formalisms form one of the basic components of these systems. Current data driven dependency parsing formalisms can be divided into three different types:

- Local-and-greedy transition based parsers (e.g., MALTPARSER and similar parsers [Nilsson et al.2006], [Yamada and Matsumoto2003]),
- Globally optimized graph-based parsers which are also known as discriminative graph based dependency parsers. (e.g., MSTPARSER [McDonald et al.2005] ; [Koo and Collins2010] and [Carreras et al.2006]), and
- Hybrid systems (e.g., ([Sagae and Lavie2006] and [Nivre and McDonald2008])), which combine the output of various parsers into a new and improved parse.

Transition based parsers basically scan the input from left to right. They usually have linear complexity and mostly make use of a big list of features. Most of their their decisions are local. Some of the transition based parsers have the restriction of sticking to the 'left to right' direction [Nilsson et al.2006]. [Nilsson et al.2006] also states that the transition based parsers have $O(n)$ complexity, that is, it has a 'linear complexity'. Also note, even if they can use the big list of features, which can basically include the rich structural information, it is basically restricted, as only the next two or three lexemes are available to the parser. This implies that it has a very small look-ahead window, and hence, it is right to predict the relatively less rich contextual information. This usually results in error propagation and relatively bad performance on root and long distance dependencies when compared to graph based or discriminative dependency parsers [McDonald and Nivre2007].

The graph based dependency parsers on the other hand, have a better contextual information. They usually perform an 'exhaustive' search over all the probable parse trees for a given sentence, and hence are globally optimized. Once they do an 'exhaustive' search they sum all the possible tree structures (this will be discussed in length in the following) to find the best scoring tree for a given sentence. But then, an increase in the feature sets actually makes it a hard problem, hence the feature sets are mostly restricted to the single edges - (which is the first order parsing model) or edges pairs (second order parsers - ([McDonald et al.2006]; [Carreras2007]) or edge triplets (third order [Koo and Collins2010]). There have been efforts on incorporating arbitrary tree-based features, but then these adversely affect the overall complexity of the parser. These models have at least $O(n^3)$ complexity when being highly greedy. In this thesis, we will concentrate on this family of parsers and explore mostly the higher order parsers - second and higher.

1.3 Research Objective

The major focus of the research presented in this thesis is to the investigate the effects of semantic and morphological features on the discriminative data driven dependency parsing. Especially, we will be concentrating on the graph based dependency parsing.

One powerful aspect of discriminative models is their ability to incorporate rich sets of highly dependent features (this will be discussed in the next chapter). The question that we seek an answer for in this thesis is: *"How do these features effect parsing?"*. [Bikel2004a] has done a detailed analysis on each class of distribution of features for generative models. But a similar analysis seems to be missing for discriminative models, especially the graph based parsing models. This thesis is a step ahead into a similar analysis.

Chapter 2

The Premise

In this section, we review the background concepts of Dependency Grammar and dependency parsing. We will then provide a succinct description of the research orientation adopted. We will also present a holistic view of the graph based dependency parsing models.

2.1 Why Parsing?

Natural languages like English is hard to define in exact terms and is ambiguous in many situations, while a formal language is mostly well defined and less ambiguous. A natural language has often evolved during thousands of years and yet it continues to evolve. This makes it impossible to state an exact definition at a given time. It is also hard to draw boundaries between natural languages, and whether a particular language is counted as an independent language is usually dependent on historical events and culture, seldom only on the linguistic criteria. These properties not only make natural language processing a challenging task but also a very interesting research topic. Especially with the increasing use of information technology in combination with natural languages. Many computer applications that involve natural languages like machine translation, question answering and information extraction are dependent on modeling natural language in an easier representation. Moreover, these applications usually have to deal with unrestricted text, including grammatically correct text, ungrammatical text and foreign expressions.

Thus, parsing of natural languages can be seen as the process of mapping an input string or a sentence to its syntactic representation. We assume that every sentence in the given set of sentences (or in other words the corpus) has a single correct analysis which the speakers of the language generally agree that this analysis is preferable. We do not necessarily assume that a formal grammar defines the relationship between sentences and their preferred interpretations. [Nilsson et al.2006] in his paper uses the concept of text analysis to characterize this problem that can only be evaluated with respect to the empirical evidence of a language text.

Several attempts have been made to formalize the grammar of the language over a long period. The first records of such an attempt dates back to 400 BC, when Panini described and formalized the Sanskrit grammar. The first computational study of grammar could be dated back to the early 1950s with the seminal work of CFG [Chomsky1956]. Since then, there have been a very large set of grammatical formalisms, which have existed and which have been used and implemented in several domain of computational linguistics. These plethora of grammar formalisms help the objective of parsing in several ways for different applications. In the next section we would solely concentrate on the Dependency Grammar formalism that forms the backbone of the formal structure in our work.

2.2 The Dependency Grammar

Syntactic representations based on word-to-word dependencies have a long tradition in general linguistics. The basic assumptions behind the notion of dependencies are summarized in the following sentences from the seminal work of [Tesnière1959]: (translated from French verbatim)

"The sentence is an organized whole; its constituent parts are the words. Every word that functions as part of a sentence is no longer isolated as in the dictionary: the mind perceives connections between the word and its neighbors; the totality of these connections forms the scaffolding of the sentence. The structural connections establish relations of dependency among the words. Each such connection in principle links a superior term and an inferior term. The superior term receives the name governor; the inferior term receives the name dependent"

The very basic dependency structure can be viewed as shown in 2.1.

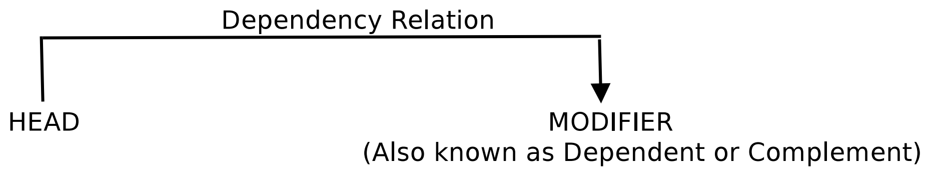

Figure 2.1: Basic dependency structure - A head and a modifier

As we have explained before, we usually represent the dependency relations among the words of a sentence as a graph. A dependency representation is a labeled directed graph, where the nodes are the lexical items and the labeled arcs represent dependency relations from heads to dependents.

It is these binary dependents - *the head and the modifier* that play a major role in the structure. Let us have a quick glance at a dependency structure in 2.2.

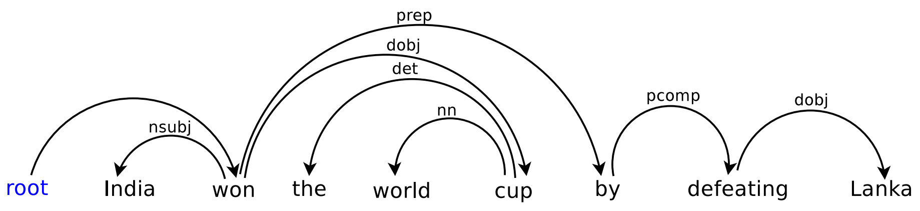

Figure 2.2: A simple example;please note that we are adding an extra root here

The dependency structure for a sentence **x** with words $\vec{x} \in [x_1...x_n]$, is the directed graph on the set of positions of $\vec{x}$ that contains an edge from i to j if and only if x_j depends on x_i . In this way, the dependency structures can capture information about certain aspects of the linguistic structure of a sentence. This notion allows us to express linguistic concepts as structural constraints on graphs. In practice, the dependencies are usually required to form a well defined and well formed tree. There are well formed rules for the retention of the tree order for a dependency tree.

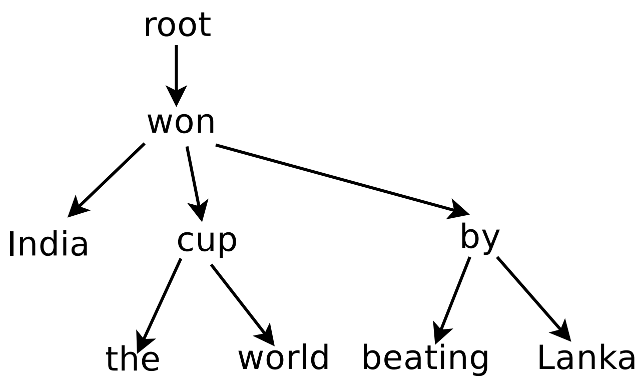

Figure 2.3: The directed graph for previous example

2.3 Projectivity and Non-Projectivity Constraints

Projectivity is concerned with the restriction of the span of the dependency relation. It requires each dependency subtree to cover a contiguous region of the sentence, hence, making sure that there is no crossing in dependency relations. Or in other words, the dependency spans don't cross each other. This is a very important constraint and it makes sure that there is no shuffle in the word order, which acts as a boon for the dependency algorithms, since, it adds a constraint. These have been strongly exploited with dependency parsers like [Eisner and Satta1999].

However, there are many languages where the dependency subtree may be spread out over a discontinuous region of a sentence, which usually results in the crossing of the spans. [Kuboň et al.1998] have mentioned that such representations mostly occur due to the linguistic phenomena such as extraction of entailment, topicalization and extrapolation. These are particularly common in languages with flexible word-order. Unfortunately, dependency parsing of non-projective structures using graph-based dependency parsing is found to be NP-Complete [McDonald and Nivre2007]. Let us investigate this quickly by venturing into the two diagrams here. 2.4 represents a projective dependency structure. 2.5 represents a non-projective dependency structure.

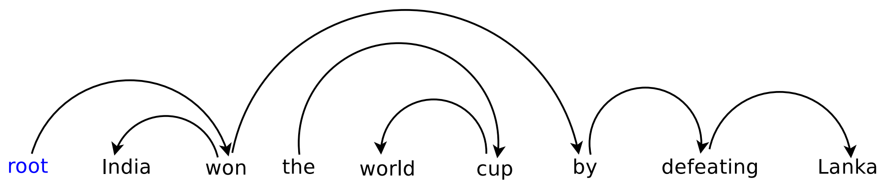

Figure 2.4: Projective Structure: none of the edges are crossing each other.

For English, projective trees are sufficient to analyze most sentence types, this is also stated in [Sleator and Temperley1993]. However, there are certain examples in which a non-projective tree is preferable. We consider one of those examples here in the next sentence. But, in general, most of the free-word-order based languages have non-projective structures more often than for English.

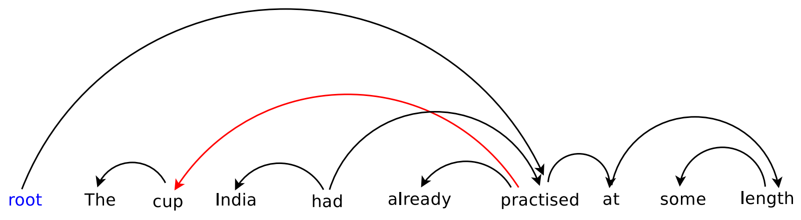

Figure 2.5: Non-Projective Structure: the red edge is basically crossing.

In this thesis all our algorithms are following the above definition of projectivity. Also note that, the parsing algorithms that we would discuss ahead in the thesis correspond to the the projective structures.

2.4 Discriminative Graph Based Dependency Parsing

Let us now shift our focus to the type of dependency parsing upon which we will concentrate in this thesis. Graph based parsers are parsers which implement a discriminative learning technique. In this sub-section we will try to explain the theory of discriminative parsing.

2.5 Notational Conventions

Let us consider that a dependency parser gets as an input a sentence $\mathbf{x}$ of n tokens and outputs a labeled dependency tree $\mathbf{y}$. A labeled dependency would be the triplet $< h, m, l >$, where the index of the head token is represented as $h \in [0...n]$, the index of the modifier token is represented as $m \in [1...n]$ and the label for each dependency pair (h, m) is represented as $l \in [1...L]$ (here, $\mathbf{L}$ is the set of all possible dependency labels in the given dataset). Also note that, the total number of sentences in the corpus is assumed to be χ and the number of possible trees for the sentences in the whole corpus is assumed to be γ.

The head of the sentence is assigned a value $h = 0$, in this thesis, we represent it by a special *root* symbol, which is as represented in the previous figures. $D(x)$ represents all possible dependencies and the set $(\vec{x})$ represents all possible dependency structures for a sentence $\mathbf{x}$. $(\vec{x})$ for projective parsing algorithms and non-projective parsing algorithms differ.

Before we proceed ahead to define the parsing algorithms, let us first explore the discriminative modeling.

$$\mathbf{y}^*(\vec{x}; \vec{w}) = \arg\max_{y \in \gamma(\mathbf{x})} \vec{w} \cdot \phi(\vec{x}, y) \tag{2.1}$$

In the above equation: please note that, χ is the set of all the sentences in the corpus and γ. Here, $\phi(\vec{x}, y)$ produces a d-dimensional (where the d is dimensionality of $(\chi, \gamma)) \rightarrow \mathbf{R}^d$) vector representation of the event that dependency tree y is assigned to sentence x. Each dimension in $\phi(\vec{x}, y)$ is a feature that measures some quantifiable aspect of x and y. Hence we call $\phi(\vec{x}, y)$ as the feature vectors. The parameter vector $\vec{w}$ contains d weights corresponding to the d separate features; these parameters are learned on a training corpus of examples.

The maximization is performed over the set $(\mathbf{x})$, the size of this set increases exponentially with the length of the sentence. This makes the enumeration intractable. To take care of this situation we factor the dependencies in the following way.

2.6 Factoring Structures

Factorization constrains the feature representation so that each feature is only sensitive to a limited region of y. Essentially, the factorization breaks each structure into sets of parts, which are local substructures of y with well-defined interactions.

Consider, for a given sentence $\mathbf{x}$, with parameter vector w, and parts p, we have a modified and reduced version of the equation mentioned in the last subsection:

$$\mathbf{y}^*(\vec{x};\vec{w}) = \operatorname*{arg\,max}_{y\in\gamma(\mathbf{x})} \sum_{p\in y} \vec{w}\cdot\phi(\vec{x},p) \tag{2.2}$$

Let us consider an example to illustrate this, the simplest type of factorization is the generic factorization, that is, the first order factorization, which is also implemented by [McDonald et al.2006]. In this case a tree y is broken into n component dependencies. If this is the case, the equation would then transform to:

$$\mathbf{y}^*(\vec{x};\vec{w}) = \operatorname*{arg\,max}_{y\in\gamma(\mathbf{x})} \sum_{(h,m)\in y} \vec{w}\cdot\phi(\vec{x},h,m) \tag{2.3}$$

since - (h, m) would represent the head and modifier indices of a dependency in y. We can then apply dynamic programming algorithms efficiently to solve the parsing problem. This is well described in [Eisner2000]. We will briefly explore this shortly.

The leftmost component i.e., the $\arg\max$ function is referred to as the factorization. This method is used to decompose the tree into parts. The $\phi()$ corresponds to the features. In theory, every component can result in an opportunity for improvement, we would restrict ourselves to the feature section.

Now, if the above problem also considers non-projective structures, then a simple dynamic-programming would not suffice. But it could, still be efficiently solved by using directed maximum spanning tree algorithms as shown by [McDonald et al.2005].

Consider a case where the dependency trees are factored into larger parts, i.e., scoring groups of two or more neighboring dependencies with a shared head. This is known as higher-order factorizations, and parsers which implement this are known as higher order dependency parsers. This forms one of the seminal parts of this thesis. We shall explore later the different approaches to the higher order dependency parsing and the advantages and disadvantages of the higher order dependency parsing.

Let us now switch our focus back to the previous mentioned equations, especially the concept of estimating the parameters w.

2.6.1 Parameter Estimation using Structured Perceptron

A parameter estimation problem is usually formulated as an optimization problem. This is mostly because of different optimization criteria and also several possible parameterizations, a given problem can be solved in many ways. In this thesis we use one of the simplest parameter estimation methods - the structured perceptron algorithm, which is the generalization of the perceptron algorithm.

This algorithm was introduced by [Collins2002] in his seminal work of discriminative training models for Hidden-Markov-Models (HMM). The perceptron is one of the easiest and the simplest parameter estimation algorithms.

The averaged perceptron begins with a parameter vector, which is initialized to 0. It is then proceeds in a series of T iterations, which are basically divided into a series of estimations. Each estimation step involves selecting a **random example** from the training set, parsing that example, and checking the parsers prediction against the standard structure. If the structures differ, then the parameters $\vec{w}$ are updated with the difference between the feature vectors of the gold standard and model prediction. The output of the algorithm is not the final parameter vector, but the average of all parameter vectors across every trial in the training run. Henceforth, it is also called the averaged perceptron for parameter estimation. The parameters are basically updated in the case of a mistake. We can see the pseudocode of the algorithm in the here:

```
Input: Training Data = $(\vec{x}, \vec{y})$ where $i \in [1...n]$
$\vec{w} = 0$
$\vec{v} = 0$
**for** $t = 1 \rightarrow T$ **do**
  **for** $j = 1 \rightarrow n$ **do**
    i = Random[1, n]
    $y' = y''(x_i; \vec{w})$
    **if** $y' \neq y_i$ **then**
      $\vec{w} = \vec{w} + \phi(x_i, y_i) - \phi(x_i, y')$
    **end if**
    $\vec{v} = \vec{v} + \vec{w}$
  **end for**
**end for**
$\vec{v} = \vec{v}/T_n$
```

Algorithm 1: Pseudocode for average perceptron algorithm: In this algorithm $\vec{w}$ is the normal parameters and $\vec{v}$ is the summed parameters. Also note that the resultant output as described here is $\vec{v}$.

The difference computed in the algorithm, i.e., $\vec{w} = \vec{w} + \phi(x_i, y_i) - \phi(x_i, y')$, has an additional property: if the model prediction y' is mostly correct, then only a few of its parts will differ from the parts in the gold standard y_i. Thus, the update performed on the parameter vector will only modify features pertaining to incorrect or missing parts.

The averaging of parameter vectors is crucial for obtaining best results with the perceptron algorithm. As pointed out by [Carreras2007] the actual perceptron parameters yield only mediocre parsing performance, while the averaged parameters v resulting from the same run are of much higher quality. This seems to generate desirable performance for our parsers. We are only sticking to this form of parameter estimation.

We now seem to have the necessary background to understand higher order discriminative parsing models.

What we have described above is just a formalization of the tree factored into parts, that is, we have describe the score of the tree to be the sum of the edge scores. A detailed account on the factorization is explained in [McDonald et al.2005]. In the process of factorization, the whole problem of finding the dependency tree of a particular sentence has been reduced to the problem of finding maximum spanning trees. In this section we restrict the definition to the more refined, projective dependency trees.

[Koo and Collins2010] defines order of a part as *the number of dependencies a part contains*. In the following sub-sections we shall see explore the different projective algorithms and then we would concentrate on the existing feature structure and the proposed changes in the feature structure.

2.7 Parsing Algorithms

In this section we briefly describe the current research on graph based dependency algorithms. We have briefly described the algorithms which are one of the most essential parts for the justification of our hypothesis.

Before we go further with the description of the dynamic programming structures, let us understand the terminology which is generally accepted:

- Simple Dependency: The simple dependency as we have been defining since the beginning, is made of a head (h) and a modifier (m) relationship.

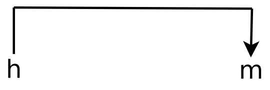

Figure 2.6: The basic dependency structure.

- Sibling Structure: A sibling is defined as the relation where the modifiers share the same head. We consider here one as the main modifier (m), while the other as the sibling (s).
- Grandchild Structure: In case of a grandchild structure, the head has a parent, that is the grandparent (g). The structure is depicted in the figure here.
- Grandsibling Structure: Here, the sibling structure defined above is headed by a grandparent. Hence, forth a grand-sibling structure.

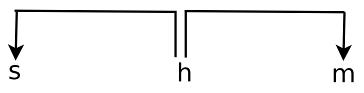

Figure 2.7: Sibling structure.

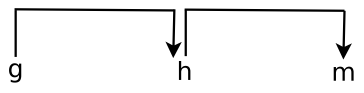

Figure 2.8: Grandchild Structure

Let us now understand the basic dynamic programming structures. The algorithms in detail would not be mentioned, but the relevant reference would be made for each of the algorithmic formulation.

First-Order Factored Parsing Algorithm

One of the earliest implementation of discriminative dependency parsing was introduced in the seminal work by [Eisner2000] who used dynamic programming for first order parsing. This laid the foundation to other parsing algorithms in the area of graph based dependency parsing.

The most important part of the algorithm is that it has two main components - *the complete span* and *the incomplete span*. The complete span on one hand consists of a headword and its modifiers, while the incomplete span consists of the region between the head and the modifier.

A slightly modified version of CKY [Eisner and Smith2010] chart parsing algorithm would generate and represent the dependency trees in less than $O(n^5)$ (which is the standard time complexity of CKY algorithm) time to create. However, [Eisner2000] parses left and right dependents of a word independently and then combines them at a later stage. This reduces the time complexity from the standard - $O(n^5)$ to $O(n^3)$ as each derivation is defined by fixed boundaries of a 'span (which is two)' and 'a split point'.

We can think about a complete span as a **'half-constituent'** of a dependency tree part. This half-constituent is headed by a head 'h' and is modified by a modifier 'm'. Similarly, an incomplete span can be thought of as a **'partial half-constituent'**, because, this is extended by adding modifiers to m.

Let us consider a complete span as $C_{h,e}$ where h and e are the indices of the span's headword and endpoint. An incomplete span may then be written as $I_{h,m}$, where m is

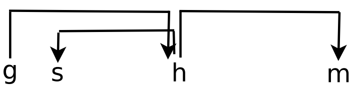

Figure 2.9: Grandsibling Structure

the modifier of h. As we have seen above, with the Eisner's independent combination strategy, again, each span is created by recursive combination of smaller spans. An incomplete span is constructed from a pair of complete spans, while a complete span is created by combining the incomplete span with the other half of the constituent.

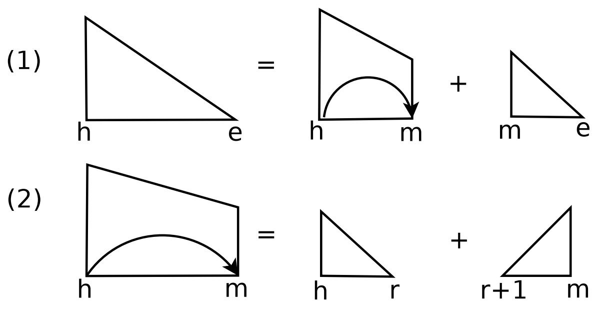

Figure 2.10: First-Order dynamic programming structures
Kindly note here that complete spans are triangles and incomplete spans are trapezoids.

The above process is a recursive process. Also in the figure, the span combines at a point m (in (a)) and at a point r (in (b)) is the free index that must be enumerated to find the optimal construction. This is also split point. The point of concatenation is found to infer the optimal construction. This is exactly accomplished by modifying the CKY parsing techniques as mentioned in [Cocke and Schwartz1970] and [Kasami1965].

2.7.1 Second-Order Factored Parsing Algorithm

In a second order factored algorithm a part contains 2 dependencies. The implementation of second order parsing algorithms have been majorly done in the below mentioned ways:

- ***Sibling factorization*** - This was introduced by [McDonald et al.2006] where the dynamic programming structures were modified to explore the possibility of extending the parts to include the sibling information. That is two words with a shared head word. In this case, a sibling information is a triplet $< h, m, s >$. Extending from the previous algorithm, (h, m) and (h, s) are dependencies and s and m are successive modifiers to the same side of h. For this case, the dynamic programming structure has been augmented to include an extra structure: sibling spans. Sibling span represents the region between successive modifiers and of a head. Let us consider a sibling span as $S_{s,m}$, here s and m are the successive modifiers involved in the relationship.

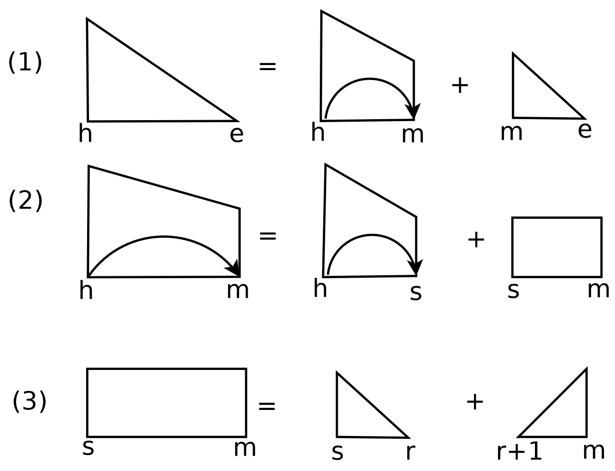

Figure 2.11: Second order dynamic programming structure
This shows the sibling spans

In this case, the incomplete spans are constructed in a completely different way as opposed the earlier case. Here, the parser combines incomplete span, that represents the innermost dependency with a sibling span. Even in this case, each derivation is still defined by a span and split point only. Hence, even here the parser requires $O(n^3)$ time.

- ***Grandchild factorization*** - The grandchild factorization was introduced by [Carreras2007] in which parser tries to exploit the grandchild parts, that is, the part includes the children of head and modifier. So, in this case, a grandchild information is a triplet $< h, m, c >$. Extending from the same first order factorization, (h, m) and (m, c) are now the dependencies. Again, for this case, the dynamic programming structure is modified to include the identity of the *outermost* modifier of the head of the complete span. These also make use of the aforementioned sibling parts - (h, m, s) but then s and c are effectively independent of each other and hence the algorithm is optimized to deal with each index separately. Here, please note that the grand child relation changes the parsing algorithm now, in terms of the computational complexity. That is the complexity increases from $O(n^3)$ to

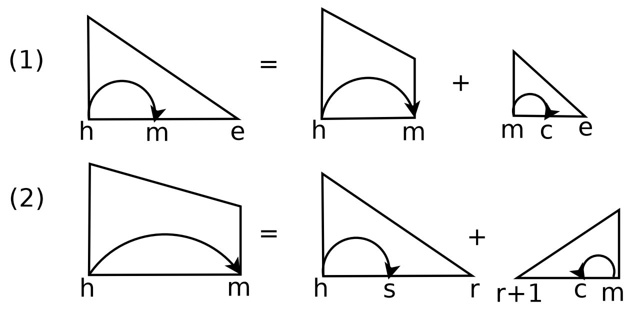

Figure 2.12: Second order dynamic programming structures
Grandchild spans

$O(n^4)$.

2.7.2 Third-Order Factored Parsing Algorithm

Third order parsing algorithms was introduced by [Koo and Collins2010] which basically extends the above approaches. This is mostly by augmenting the grand-parent index. [Koo and Collins2010] remarks that the efficiency of the third order algorithms is due to sa fundamental asymmetry in the structure of a directed tree, i.e., a head can have any number of modifiers but a modifier always has a single head. So in a way, this exploits the structural asymmetry. [Koo and Collins2010] specifically divides the parsing algorithms into three different models, the important two are mentioned below, which we have tried to experiment on:

- *Include all grandchildren* In [Koo and Collins2010]'s structures a grandchild is a part contains the information of the triplet $< g, h, m >$, where (g, h) and (h, m) are dependencies. For this both complete and incomplete spans are augmented with g-spans. Hence, in other words, it basically represents the same first-order algorithm, but now it includes the indices of the grandparent. Please do note that here each derivation copies the grandparent index g into smaller g-spans. This actually causes each g-span to have non-contiguous structure. This is basically an extension of the second order grand-child factorization

  Even with this algorithm the time complexity is $O(n^3)$ as each derivation is defined by three fixed indices and one split point.

- *Include all grand-siblings* In this case, we decomposed each tree into a set of grand-sibling parts which consist of the sibling parts and the grandchild parts. i.e., a grand-sibling is a quadruple $< g, h, m, s >$ where (h, m, s) is basically the sibling part from above and (g, h, m) is the grandchild parts. Its almost like a hybrid of the aforementioned approaches.

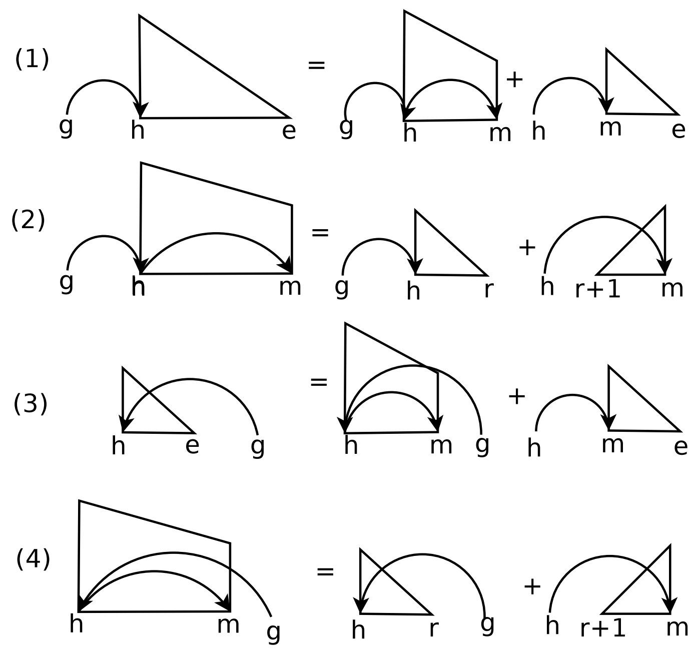

Figure 2.13: Third order dynamic programming structures
Grandchild

2.8 Feature Space

All through the previous sections we defined the score of an edge, but we intrinsically made an assumption about the feature space. We supposed that we have a high dimensional feature representation for each edge. The feature set description in the above implementations maintain the successful previous work in first order dependency parsing [McDonald et al.2005], [McDonald et al.2006], [Carreras2007].

Let us understand feature space with an example from [McDonald et al.2005]. Let us consider a feature selection from the form of the words, the lemma of the words, the pos tags of the words. The features are basically the indicatory functions (most often binary), each of the functions evaluate the presence of a certain pattern in a dependency. Consider a feature pattern which takes into account part-of-speech tag and the form of the word into consideration. There is an implicit direction on each dependency part - the left or the right. Let us assume that if $|h| < |m|$ then the direction is right and it is left otherwise. If these are the given conditions, then the part $\phi(x, h, m, c)$ can be defined as:

- dir.pos (h).pos (m)
- dir.form (h).pos (m)

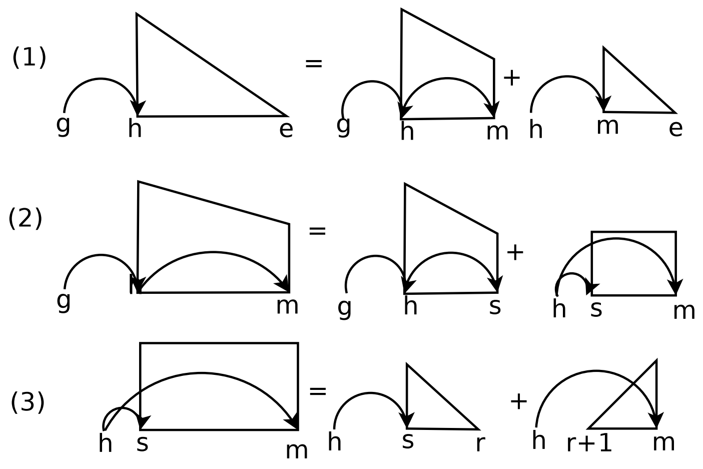

Figure 2.14: third order dynamic programming structures grand-sibling.

- dir.form (m).pos (h)
- dir.form (h).pos (m)
- dir.form (h).form (m)

The most basic feature patterns consider the surface form, part-of-speech, lemma and other morphosyntactic attributes of the head or the modifier of a dependency. The representation also considers complex features that use a variety of part-of-speech tags of the following items: the head and modifier; the head, modifier, and any token in between them; the head, modifier, and the two tokens following or preceding them. Most of the above implementations of the parsing algorithms involve using the syntactic features.

In our experiments we are experimenting with a deprived tree, by depriving it of the dependency-labels. This is because the addition of labels actually increases feature space. For simplicity and computational reasons, we have restricted the experiments to unlabeled parsing. Just like the concept of direction defined above, the label is actually dependent on both head and modifier.

2.9 Effect of Features in Dependency Parsing

[Bikel2004b] provided a detailed analysis of the contribution of each class of distribution to the generative power of the model for generative parsing models. But, unfortunately the non-probabilistic nature of our models prevents that detailed analysis. In this thesis we investigate the effective improvement of certain features and also explain their importance.

2.9.1 Effect of Semantic Features

Semantic information basically focuses on the relation between signifiers, like words, phrases, signs and symbols, and what they stand for. Use of semantic information to improve parsing accuracy has been an interesting but difficult goal since the early days of NLP [Ratnaparkhi et al.1994], [Hektoen1997], [Xiong2005]. There have been some good results as shown by [Ratnaparkhi et al.1994] but the overall integration has been a tough challenge. Recently [Agirre et al.2011] made an attempt to integrated semantic word classes with transition based data driven dependency parser - the Maltparser [Nivre and Hall2005] using basic semantic representations using WordNet [Fellbaum1998a]. Recently, [Agirre et al.2011] concludes that semantic information gives an improvement on a transition-based deterministic dependency parsing. Also, he mentions that feature combinations give an improvement over using a single feature. Semantic information can be further classified according to its exactness to its intended meaning.

2.9.2 Morphosyntactic and Morphosemantics with Dependency Parsing

Morphosyntactic feature is a feature which is involved in either syntactic agreement or government. A typical example is the gender, number and person are involved in agreement. In languages like English and many other languages, syntax is not sensitive to the tense value of the verb. But in languages, especially, highly inflected languages, the tense plays a major. This is a very important attribute for languages with rich morphology and inflections. The relationship of the concept of ‘gender’ i.e., the concepts ‘masculine’, ‘feminine’, ‘neuter’; or between the concept ‘case’ i.e., the concepts ‘nominative’, ‘accusative’, ‘genitive’, etc., with many languages also play a major role in deciding the sentence structure. There are some interesting results about integrating Tense, Aspect, Modality and Minimal Semantics with a dependency parser to obtain better results for parsing morphologically rich free word order language [Ambati et al.2010] [Ambati et al.2009]. They claim that with the introduction of semantic features there is a significant improvement in the performance of both the parsers. They further state that adding semantic features for nouns helps with label identification more than head identification.

Most of the current research on dependency parsing is focussed on the algorithms employed by the parsers, in this thesis we would concentrate on the issue of the relevant information to the parsing algorithm. That is experimenting on extending the feature structure, the problem of relevant and important feature extraction is as important as the research on parsing algorithms. Feature structures provide information for the

Chapter 3

State-of-the-Art and Current Research

The research in the field of dependency parsing was boosted by the successful results in the open shared tasks of the Conference on Computational Natural Language Learning (CoNLL) which concentrated on the task of dependency parsing. The relevant shared tasks which concerns this thesis and the field of dependency parsing directly are shared tasks in CoNLL-2006 [Buchholz and Marsi2006], CoNLL-2007 [Nivre et al.2007b], CoNLL-2008 [Surdeanu et al.2008] and CoNLL-2009 [Hajič et al.2009]. Most of the participating teams had novel and highly competitive methods. Though, CoNLL-2008 and CoNLL-2009 shared tasks were joint assignment of syntactic and semantic dependencies.

This thesis focuses on the aspect of pure dependency parsing rather than the task of joint assignment of syntactic and semantic dependencies, hence we will be more concerned with the systems with exclusive dependency parsing results. The results of the CoNLL 2007 shared task is presented in 3.1.

Parsing Algorithm	English	Czech	Type of Parser
Carreras	90.63	85.16	Discriminative Graph Based
Nakagawa	90.13	84.19	Probabilistic based on Gibbs Sampling
Sagae	89.87	81.27	Transition Based
Nilsson	88.93	83.59	Inductive Transition
Titov	89.73	81.20	Probabilistic

Table 3.1: CoNLL 2007 Shared Task Results. Unlabeled Attachment Score. The parsers are - [Carreras2007], [Shimizu2007], [Sagae and Tsujii2007], [Nivre et al.2007a], [Titov and Henderson2007].

We can see that shift-reduce (or transition based parsers) and [Eisner2000] based techniques are the most used as parsing approaches and have a very good performance. Eisner-based parsers use edge-factorizations, which are usually simple to approach. Very recently, the new parsing algorithms have been trying to extract maximum number of context by considering more number of words in a part - that is the concept of higher order [Koo and Collins2010]. Let us now see some of the most important developments in the field of dependency parsing in the recent past which has inspired this thesis directly and indirectly.

[Carreras et al.2008] introduced a dependency parsing algorithm inspired by TAG formalism [Joshi1969]. The main advantage of this algorithm is the ability to use features from dependency trigrams. This approach uses the splittable grammar formalism - TAG into maximum usage and also produces accurate results.

[Koo et al.2008] used an interesting approach by using semi-supervised learning. They use Brown clustering algorithm as features and achieve some very interesting results. This is specifically useful method when small amount of training data is available. This is one such research where the use of feature has shown a relatively big improvement in the parsing accuracy. This thesis is totally exploiting the work done by [Koo and Collins2010] where he introduced the parsers with third order. These algorithms have been previously explained.

Noticeable research in the field of dependency parsing, especially in the area of exploring the feature set has been rather less and not as exhaustive as with other constituent parsers. Some of the recent work that concentrates with exploiting the features are mentioned here.

[Agirre et al.2011] introduces semantic classes using WordNet, but for transition based parser. The work shows an improvement in the retrieval of labeled accuracies. But, the work does not provide an exhaustive analysis of the semantics on parsing.

[Kitagawa and Tanaka-Ishii2010] have augmented the selection of parsing actions by using a tree based approach. They build a model that considers all the considers all words necessary for selection of parsing actions by including words in the form of trees. It chooses the most probable head candidate from among the trees and uses this candidate to select a parsing action. This is a very new and interesting approach, but, it is restricted to transition based parsers.

[Song et al.2011] demonstrates a method in which a classifier is used to determine whether a pair of words forms a dependency edge. The classifier trained on the projected classification instances significantly outperforms previous projected dependency parsers when augmented with graph based dependency parsers.

[Novák and Žabokrtský2007] have showed that optimizing feature templates, there is a chance of getting a better parse structure in state-of-the art graph based dependency parsing algorithms. This work also makes a comparison to the resources spent v/s the improvement in the obtained result.

Chapter 4

System Configuration

4.1 In Focus

The parsing algorithms used in the parsers are fine tuned on the framework of averaged perceptron [Koo and Collins2010]. The parser basically scores the part as shown here:

$$Part(\vec{x}, p) = \vec{w} \cdot \phi(\vec{x}, p) \tag{4.1}$$

In the above equation, ϕ is a feature vector mapping and $\vec{w}$ is a vector of related parameters. As described before, following the standard approaches here from [Koo and Collins2010], [Carreras2007] and [McDonald et al.2006], the model scores for all of the parsers follow a generic pattern. Please note that, it is not only the higher order parts that the parser are mapped, but even the lower order parts are evaluated. For example, let us consider the third order grandsibling parser. This parser evaluates the mappings for dependencies, siblings, grandchildren and all the grandsiblings as well.

The score is calculated as:

$$\begin{aligned} Score(\vec{x}, y) = \sum_{(h,m)\in y} \vec{w} \cdot \phi(\vec{x}, h, m) + \sum_{(h,m,s)\in y} \vec{w} \cdot \phi(\vec{x}, h, m, s) + \\ \sum_{(g,h,m)\in y} \vec{w} \cdot \phi(\vec{x}, g, h, m) + \sum_{(g,h,m,s)\in y} \vec{w} \cdot \phi(\vec{x}, g, h, m, s) \end{aligned} \tag{4.2}$$

In this equation, we have defined the dependency- $(\vec{x}, h, m)$, the siblings - $(\vec{x}, h, m, s)$, the grandchildren - $(\vec{x}, g, h, m)$ and the grandsiblings - $(\vec{x}, g, h, m, s)$ as the different parts which are actually the decomposed part structures.

The speciality in this parsing approach is not just the consideration of all the lower order parse structures, but the way the feature combinations take place. In this parsing substructure, we can have 4-gram context features - consisting of 4-gram POS augmented with adjacent POS tags (when using POS as the feature). Consider an example of $\phi(\vec{x}, g, h, m, s)$. This basically includes the POS features at these positions - $(g, h, m, s, g+1, h+1, m+1)$, this means a POS 7-gram feature structure.

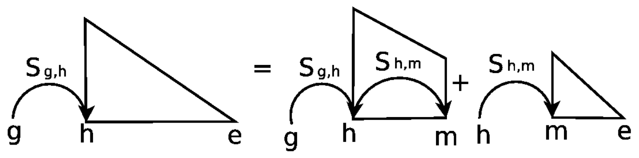

Figure 4.1: Example Structure where Sense information is included

Consider a simple example as shown in figure 4.1. In this example we can see that given the sense the information with the parser increases the amount of information required for the parser to parse the sentence structure correctly. This is, in general, more pronounced when we talk about higher order structures. [Koo et al.2008] has mostly tried to augment structures which are either syntactic or approaches like the brown clustering. In this thesis, we try to experiment with a linguistic analysis of feature addition.

4.2 System Information

A large number experiments were performed through the development process. As we stated at the start of this work, our main point of interest is to compare the scores with different features and see if there any significant effect on the score. We re-ran some experiments with the latest system configuration to facilitate a comparison across experiments. Some experiments are really expensive and hence were done using a reduced corpus, which will be explained below.

4.2.1 Input Format

The input format for our experiments is using the CoNLL-X format [1] [Nivre and Hall2005].

The data format is as follows:

- The data files contain sentences separated by a blank line.
- A sentence consists of one or more tokens and the information for each token is represented on a separate line.

[1]This can be accessed here http://ilk.uvt.nl/conll/

- A token consists of at least 11 fields, described in the table below. The fields are separated by one or more whitespace characters. Whitespace characters are not allowed within fields.

The CoNLL format which we used for our experimentation looks something like this:

1. ID - Token counter, starting at 1 for each new sentence.
2. FORM - Word form or punctuation symbol.
3. LEMMA - Lemma or stem (depending on particular data set) of word form, or an underscore if not available.
4. CPOSTAG - Coarse-grained part-of-speech tag, where tagset depends on the language.
5. POSTAG - Fine-grained part-of-speech tag, where the tagset depends on the language, or identical to the coarse-grained part-of-speech tag if not available.
6. FEATS - Unordered set of syntactic and/or morphological features (depending on the particular language), separated by a vertical bar (|), or an underscore if not available.
7. HEAD - Head of the current token, which is either a value of ID or zero ('0').
8. DEPREL - Dependency relation to the HEAD. The set of dependency relations depends on the particular language.

4.2.2 Corpus Used

We are experimenting with two languages - **English and Czech** which are widely used in shared tasks and are also widely experimented. This is also largely because of the ready availability of finely tagged standard corpora. In this section we will describe about the corpora which were used in the process of experimentation.

English

The corpus consisted of Penn Treebank (PTB) [Marcus et al.1993] corpus and it was converted to the required format by using Penn2Malt constituency-to-dependency converter. We used a subset of this corpus that consisted of:

1. 15, 000 Sentences - Training
2. 1000 Sentences - Validation
3. 2000 Sentences - testing

We used a reduced version (i.e., reduced state) as these experiments are heavily computationally intensive and need a lot of resources for different experiments.

For English, the interest in dependency parsing has been a bit weaker than for other languages. This is probably because of the strong tradition of constituent analysis in

Anglo-American linguistics, this is reinforced by the creation of a big treebank for American English, the Penn Treebank [Marcus et al.1993], that is annotated with constituent analyses. At the same time, there has been increasing interest in using dependency parses for a range of NLP tasks, from machine translation to question answering. This is one of the reasons, why most of the other languages have shifted to building treebanks natively with a dependency grammar formalism. Even in this work, we use Penn2Malt [Johansson and Nugues2007] for the process of conversion from the constituency tagged corpus format to a dependency tagged corpus.

Czech

We used the Prague Dependency Treebank (PDT) [Böhmová et al.2001] for Czech, which was converted by using some scripts provided in the TectoMT. Though the data consisted of a big set of sentences, unfortunately generating models for all of the them is a very expensive task. Hence we reduced this by extracting a set of thousand sentences from each of these corpora. Since the result was computed with several models with similar dataset, we believe that the results still holds merit as these were comparative results.

Again the division of the sets were:

1. 15, 000 Sentences - Training
2. 1000 Sentences - Validation
3. 2000 Sentences - testing

The Prague Dependency Treebank [Böhmová et al.2001] consist of Czech texts annotated with syntactical information consisting mainly of dependency relationships. Unlike English, Prague Dependency Treebank is natively a corpus following the dependency grammar formalism. One of the most popular benchmarks for evaluating parser quality is by evaluating against the surface-syntactic trees provided by the Prague Dependency Treebank.

4.2.3 Tools

Dependency Parser

We use [Koo and Collins2010]'s higher order dependency parser dpo3[2] which is freely available with a GPL license. This parser already builds algorithms for all the higher order parsers and provides the ideal basis to experiment. This lets us use the ideal base to experiment with different dynamic algorithms and test our hypotheses.

[2]This can be accessed here http://groups.csail.mit.edu/nlp/dpo3/

4.3 Word Sense Extraction

- **Fine-grained Word Sense Extraction**We use the standard word sense disambiguation [Pedersen and Kolhatkar2009] algorithm to do the basic word sense disambiguation. What it does is, finds the sense of each word that is most related to the senses of the surrounding words based on a similarity measure. It proceeds word by word from left to right, centering each content word in a balanced window of context, whose size is determined by the user, of surrounding words. At each stage, the token being disambiguated is called “the target”, and the surrounding tokens “the context window”. The size of the context is determined by the user and will be referred to as window size. . A balanced context is chosen according to the size of the window. The goal of the algorithm is to select one of the senses from the set of possible senses. This is done by measuring the semantic relatedness between the possible senses of the target and the possible senses of each of the tokens in the context window. Consider a sense pair (s_k, s'_{il}), where s_k is the k^{th} sense of the word being considered and s'_{il} represents the l^{th} sense of the i^{th} word in the context window. A ‘relatedness’ function takes as input two senses, and outputs a real number. It is assumed that this real number is indicative of the degree of semantic similarity between the two input senses. A larger number denotes high relatedness between the two senses and a smaller number denotes low relatedness between the senses. In simple terms the equation that is used to calculate the sense is given below:

  $$S_k = \sum_{1ton}(\max(s_k, s'_{il})) \tag{4.3}$$

  Here, S_k is the final-chosen sense for the given word in the context. WordNet is used to fetch the number of possible senses.
- **Coarse-grained Word Sense Extraction** [Fellbaum1998b] describes that the organization of words in the WordNet are done as sets of synonyms, called synsets. Each synset in turn belongs to a unique semantic file (SF). There are total of 45 SFs (1 for adverbs, 3 for adjectives, 15 for verbs, and 26 for nouns), based on syntactic and semantic categories. A script is written in order to fetch coarse-sense disambiguation. This script also uses [Pedersen and Kolhatkar2009]’s word sense disambiguation tool, but then makes it more coarse grained for our experiments. The tags are $SF00...SF44$.

In both cases for Czech we obtain these by from the Prague Dependency treebank, since these are already annotated.

4.4 Morphosyntactic Feature Extraction

These were exclusively done for the Czech language since we had the corpus already tagged, also these were of GOLD standard. The morphosyntactic features are the basic

Category	Number of Values
POS	10
SUBPOS	75
GENDER	8
NUMBER	4
CASE	9
POSSGENDER	4
POSSNUMBER	3
PERSON	5
TENSE	4
GRADE	5
NEGATION	3
VOICE	3
VAR	3

Table 4.1: Morphological Tagset for Czech

morphological features in the Prague Dependency Treebank which are extracted from the m-layer.

There are 13 categories in the Czech morphological tagset with 4452 plausible combinations. These are briefly mentioned in the table 4.1. We extract the morphosyntactic features out of this table - both full and take specific morphosyntactic features into consideration and try to experiment with different possible combinations. Some of these tags are very important and are mostly specific for an inflectional language as these contain a lot of information which is essential to build a good sentence. This inspires us to embed these features into the various dependency parsing algorithms and gain an insight into these feature structures.

Chapter 5

Experiments

In this chapter we systematically explore the experiments that were conducted conducted and finally we conclude with a result.

5.1 Experiment 1

5.1.1 Research Question

What is the effect of using the fine-grained semantic word-sense as a feature with different graph based dependency parsing algorithms?

5.1.2 Theoretical Plausibility

English

As explained in the previous chapters, we can see that dependency parsing, in general, is very useful for semantic analyses. This draws us closer to the question of whether there is any effect on the parsing accuracy if semantic information is available. We investigate the possibility by including specific word-senses. This becomes especially interesting with the higher order parsing algorithms, since, these have better context information. We use the technique as explained in the previous chapter to extract the wordsense extraction by disambiguation.

To explain the possibility of wordsense let us consider an example as shown in 5.1. Now, given the word form and the part of speech tag, “cricket” has exactly the same pos tag in both ‘cricket as an insect’ and ‘cricket as a sport’. We can see that the possibilities, given several parses with the same form and POS information, would make it ambiguous. This would change if we have more information. In this example, the sense can be useful to distinguish between the two different forms of the word.

A sense tagged corpus would clearly give better set of information to the parsing algorithm. Also, if we consider, second and higher order parsing algorithms, then the amount of information would be significant and hence theoretically the accuracy in predicting the structure would be considerably higher. In the past, there have been approaches, like [Agirre et al.2011], where they have experimented with a coarse grained tagset. This we would explore in the forthcoming experiment.

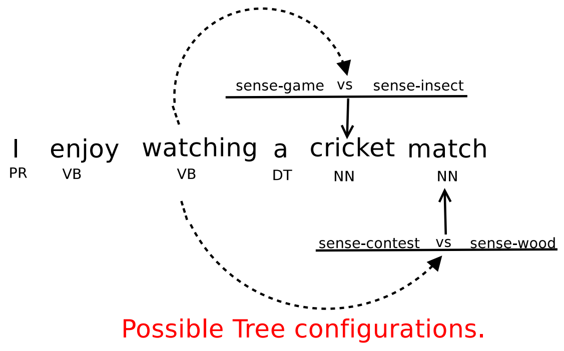
sense-game vs sense-insect
I enjoy watching a cricket match
PR VB VB DT NN NN
sense-contest vs sense-wood
Possible Tree configurations.

Figure 5.1: Effect of Wordsense

Czech

Following the same logic as for English, for Czech, there is a well defined Semantic POS tag in the t-layer of the Prague dependency treebank. We have symbolically treated this as the coarse grained semantic tagset. But one interesting issue is that, these are GOLD standard tagsets and hence we can have a clearer look at the results.The tagset for the Czech has been explained in the following section.

5.1.3 Experimentation

Research Question

English

As explained before, we use [Pedersen and Kolhatkar2009]'s algorithm to disambiguate and associate senses for each word in a given sentence. Now, given a target word and its part of speech, the algorithm chooses the best possible sense. Consider an example sentence “I enjoy watching a cricket match”, in this case cricket as a game would in the WordNet would belong to the sense number 1. Hence, the output would be n1 when tagged. A sample sentence from the training dataset is mentioned below:

1	Ms.	-	NN	n2	-	2	NMOD
2	Haag	-	NN	-	-	3	SUB
3	plays	-	VB	v3	-	0	ROOT
4	Elianti	-	NN	-	-	3	OBJ
5	.	-	.	-	-	3	P

Table 5.1: A sample set from the Penn Treebank tagged with fine-grained word-sense tags for English

Czech

The t-layer of the PDT [Böhmová et al.2001] is tagged with the grammateme semantic postags or sempos. We have extracted the sempos from the PDT and we are have experimented with the sempos. The tagset of the sempos is given in 5.2.

5.1.4 Results

English

As we have stated above, for this experiment, we evaluated the performance of our system using 15, 000 sentences from the penn treebank corpus which are converted by the “pennconverter” into the CoNLL format. This is a tool to automatically convert the constituent format used in the Penn Treebank into dependency trees. This was also used in the previous versions of the CoNLL shared task. We tagged the corpus with the fine-grained semantic tags, i.e., these tags actually mention the possible sense number from the list of senses retrieved from the WordNet. Kindly note that, the corpus was tagged with coarse grained TagSet which was the same as was present in the Penn Treebank. Seeing table 5.3 we notice a gradual improvement in the parsers accuracy for unlabeled attachment score.

Czech

The table 5.4 shows the result for the Prague Dependency treebank.

5.1.5 Discussion

One basic difference between our approach and the other previous approaches on augmenting wordsense is that here, we use relatively a ‘more specific’ wordsense of the word. This provides more specific information to the parsing algorithm. The grand-sibling based parsing algorithm shows better performance than the grandchild based parsing algorithm. A close analysis reveals that the sibling based interactions that are local, are easily retrieved. While, the farther sibling interactions don’t necessarily give better results.

Type	Explanation
n.denot	denominating semantic noun
n.denot.neg	denominating semantic noun with separately represented negation
n.pron.def.demon	definite pronominal semantic noun: demonstrative
n.pron.def.pers	definite pronominal semantic noun: personal
n.pron.indef	indefinite pronominal semantic noun
n.quant.def	definite quantificational semantic noun
adj.denot	denominating semantic adjective
adj.pron.def.demon	definite pronominal semantic adjective: demonstrative
adj.pron.indef	indefinite pronominal semantic adjective
adj.quant.def	definite quantificational semantic adjective
adj.quant.indef	indefinite quantificational semantic adjective
adj.quant.grad	gradable quantificational semantic adjective
adv.denot.ngrad.nneg	nongradable denominating semantic adverb, impossible to negate
adv.denot.ngrad.neg	nongradable denominating semantic adverb, possible to negate
adv.denot.grad.nneg	gradable denominating semantic adverb, impossible to negate
adv.denot.grad.neg	gradable denominating semantic adverb, possible to negate
adv.pron.def	definite pronominal semantic adverb
adv.pron.indef	indefinite pronominal semantic adverb
v	semantic verb

Table 5.2: Semantic POS-Tags for Czech from the Prague Dependency Treebank

Dependency Parsing Algorithm	Add Semantic Tags	Original Score	Difference
Third order grand-sibling	91.10	90.29	+0.81
Third order GrandChild	90.72	90.57	+0.15
Second order grand-sibling	88.22	87.45	+0.77
Second order GrandChild	88.48	88.34	+0.14

Table 5.3: Results for Penn Treebank

Dependency Parsing Algorithm	Add Semantic Tags	Original Score	Difference
Third order grand-sibling	86.87	85.67	+1.20
Third order GrandChild	86.82	86.03	+0.79
Second order grand-sibling	85.19	84.32	+0.87
Second order GrandChild	85.23	84.79	+0.44

Table 5.4: Results for Prague Dependency Treebank

If we closely look at the results we find that for both the languages, there is a generic uplift in performance for all the different kind of algorithms. But, the statistically significant results can be seen especially in the sibling-based parsing algorithms. Both in English and in Czech the grand-sibling based algorithm show a good improvement.

In this particular case you see the difference, there is a big difference for the Czech language, where the third order sibling parser differs from the original score by a difference of $+1.20$. The other differences are of the similar magnitude, but more pronounced with sibling parsers.

5.2 Experiment 2

5.2.1 Research Question

What is the effect of using the coarse-grained semantic word-sense as a feature with different graph based dependency parsing algorithms?

5.2.2 Theoretical Plausibility

Before we justify the cause of the hypothesis, let us try to understand what coarse-grained semantic word-sense tags are. For English, these tags are basically, a very high level representation of the possible semantic orientation of the word. For example, an animate noun which denotes action or acts would be classified in a particular set containing all of the animated nouns.

For English, a similar approach has been tried for transition based dependency parsing by [Agirre et al.2011]. We approximately approach in a similar manner and work on the level of semantic files to extract the details of the coarse-grained wordsenses. Though, in our case, due to the lack of gold tagged data, we use just WordNet to tag our corpus.There has been some improvement with the transition based parsing, we experiment if at all there is an effect with graph based dependency parsing algorithms.

Unfortunately, we couldn't extend the same results to Czech. This was primarily due a problem in fetching a similar structural source like the WordNet due to several practical constraints.

5.2.3 Experimentation

As we have explained before, these were basically the semantic files and out tag representations were the closed set $[SF00...SF45]$.

5.2.4 Results

The results for the experiment is mentioned in table 5.5.

Dependency Parsing Algorithm	Add Semantic Tags	Original Score	Difference
Third order grand-sibling	91.10	90.29	+0.81
Third order GrandChild	90.72	90.57	+0.15
Second order grand-sibling	88.54	87.45	+1.09
Second order GrandChild	88.62	88.34	+0.28

Table 5.5: Results for Penn Treebank

5.2.5 Discussion

Again, in this case, we see that the sibling based parsers perform better than the other forms of parsing algorithms. Also, please note that the second order parser shows a better performance than the third order sibling parser with an improvement of +1.09 units.

The improvement in the performance of the parser is a very encouraging result. Given the nature of semantic classes and word sense disambiguation algorithms, there seems to be room for a lot of improvement. This gives us the possibility of exploring information like WordNet concepts, wikipedia concepts and other related concepts, which could be essentially important for various Natural Language Processing tasks like Semantic Role Labeling, etc., also, these results are very interesting for fields of Machine Translation and other related fields.

5.3 Experiment-3

5.3.1 Research Question

What is the effect of using the morphosyntactic tags as a feature with different graph based dependency parsing algorithms?

5.3.2 Theoretical Plausibility

Czech

Czech is a morphologically rich language [Horák et al.2007]. The morphological tags are known to contain a lot of significant syntactic information. These information seemingly play an important role in the parsing of morphologically rich free order languages. With a big window of contexts, the amount of useful information could be extended.

English

For English, though morphological information is important, but the amount of contribution of the morphology might not be as great as the contribution from the Czech. We also could tag the corpus from Fine-Grained POS tags directly from the Penn Treebanks tagged corpus. We found out that the Fine-Grained POS contains a lot of morphological information.

5.3.3 Experimentation

Czech

We initially tried with the 15-letter tags as individual features to and exploit the whole tagset with the parsing algorithm. But unfortunately, it couldn't give better results mostly due to the problem of overfitting of the feature space. This made us make experimentation on linguistically coherent choices with respect to the parsing decision. Later, we chose a subset of these morphological features, the subset was chosen on the basis of relative importance of the particular morphological tag in providing the relevant information which might be helpful. The tags that we considered were specific tags like:

- **POSSGender** - Gender is an inherent feature of nouns and is also a contextual feature. Gender in basically determined through agreement. Gender is lexically prodsuced and its value is fixed for the noun.
- **Number** - It is a morphosyntactic feature if it participates in agreement or as a government in the language
- **Case** - Case is a feature that expresses a syntactic or semantic function of the element that carries the particular case value.
- **Person** - Person as a morphosyntactic feature is typically a feature of agreement.
- **Tense** - It denotes the semantic feature of location in time.
- **Voice** - The temporal feature between the subject and the verb.

English

As explained before, for english we use the standard Penn Treebank's fine grained tagset. The fine grained tagset also contains a lot of morphological information.

5.3.4 Results

Czech

As explained in the previous section, preliminary tests on a portion of train data showed that the complete morphological tagset feature templates decrease the accuracy. Hence we concentrated on experimenting with the smaller morphological tagset. 5.6 shows the result.

Dependency Parsing Algorithm	Add Morph. Tags	Original Score	Difference
Third order grand-sibling	86.12	85.67	+0.45
Third order GrandChild	87.75	86.03	+1.72
Second order grand-sibling	84.88	84.32	+0.56
Second order GrandChild	85.51	84.79	+0.72

Table 5.6: Results for Prague Dependency Treebank

Dependency Parsing Algorithm	Add Morph. Tags	Original Score	Difference
Third order grand-sibling	90.50	90.29	+0.21
Third order GrandChild	91.78	90.57	+1.21
Second order grand-sibling	87.67	87.45	+0.22
Second order GrandChild	89.32	88.34	+0.98

Table 5.7: Results for Penn Treebank

English

As explained before, we are using the fine-grained tagset here.

5.3.5 Discussion

Third order grandchild shows a significant improvement of about 1.72% for Czech. This is an important result, since grandchild based parsing algorithms seem to be better than their counterparts when included with the morphological tags. Also, please note that, this corroborates the linguistic assumption about morphological information, an important factor for the morphologically rich languages.

In case of English, the fine-grained POS tags, basically these tags are enriched with morphological information, also show a very interesting improvement. The third-order grandchild based parser shows +1.21 improvement.

Chapter 6

Discussion

Ideally it is desirable to use many features collectively and perform the process of parsing, eventually arriving at an optimal solution. But, each feature increases the search space quite remarkably and hence, there are two important problems here -

1. **Problem of Overfitting** Overfitting generally occurs when a model is excessively complex, such as having too many parameters. This works in a contra-productive way most of the time.
2. **Parsing Time** The amount of parsing time increases with increase features too. This is again because of the increase in the search space which might act in a negative way to reduce the parsing accuracy.

Also, please note that the current tagging of the wordsense was done using a simple algorithm [Pedersen and Kolhatkar2009]. There are better algorithms which could give better tagging accuracies. This might have a direct effect on the parsers' performance. Another important thing to note is that, the morphological taggers have better accuracies in practical applications. Hence the proposed approach would be useful, if, we are able to carefully combine the accurate features.

POS tags provide very basic linguistic information in the form of broad grained categories. Among all the parsing structures, we saw that an augmentation of the specific wordsenses has improved the unlabeled accuracy scores. Its evident that one of the parsing algorithms - is performing much better than other parsing algorithms. It is more important to note that, it is the type of the parsing algorithm along with the features that makes makes a noticeable change in the score.

Also, after a close investigation when we introduced morphological tags, we found that we have a very strong problem with the agreement. The agreement problem is bad whenever there is a coordination conjunct or when there is a complex verb.

There is still a lot of work to be done with the parsing structures and on the observe the correct set of features to take full advantage of the dependency grammar in practical application of NLP.

Chapter 7

Future Work

One of the most important things that we would be working next would be to experiment with the labelled accuracy score and augment parser with wordsenses. The current parser doesn't score the labels. We would extending the parser to score both the labels and also integrate wordsenses in the parsing structure.

The overall approach is to augment each part and each dynamic-programming structure with senses and labels. Let us assume that the word senses can be represented as indices in the set $1, ..., S^*$ while dependency labels can be represented as indices in $1, ..., L$; here, S^* and L denote the total number of senses and labels. For every part now, we will have (h, m, l, s_h, s_m) (that is $s_h and s_m \in S^* and l \in L$ that is we need to embed both labels and senses with the parse structures.

A natural idea for future work is to evaluate the effects of combining several semantic features with the different parsing algorithms and building the specific set of features which increase the parsing accuracy for each language.

We can also investigate if increasing the order of the parsers - that is 4th order and higher would be interesting. Primarily because these would give us a range of order for optimal parses. Though, this might be a difficult task due to the inherent complexity of both space and time problems.

Another interesting dimension of approach would be about application of these parsers in several fields. The field of Machine Translation and automated summarization would especially need high end results. Also, since semantics plays an important role in these parsing frameworks, it might help to experiment with higher order parsers. Based on the interesting work by [Popel et al.2011], it would be interesting to use these higher order parsers for Czech in TectoMT framework to extract better parse information.

As the higher order parsers are better in extracting the structural information, we could use these for the preliminary stages of various kinds of treebank tagging. Also, even if the performance of the dependency parsers have been seeing a positive change, the best parsing techniques still fall short of almost accurate performance obtained by part of speech tagging etc., and hence an area of potential research.

Chapter 8

Conclusion

In this work we provide some of the insights into extension of feature set by augmenting some lesser known features. With the introduction of semantic and morphological features there is a significant improvement in the performance of parsing algorithms. We have seen the importance of some of these features individually.

We have shown in this thesis that, the feature set in the dependency parsing algorithm need not be restricted to syntactic features only, the semantic features also add a lot of information to the parsing structure. This provides evidence that there is a relative improvement in unlabeled accuracy score whenever there is an inclusion of semantic features. The effects of semantic features are visible with higher order parsing structures.

Also, we see from the results that order plays an important part with languages which have rich morphology or inflectional languages, especially the grandchild parsing algorithms. Although we worked and presented our results only on two languages, our approach can be generalized to any language and to any framework.

Finally, we hope that the work done in this thesis inspires the use of dependency parsers, especially the higher order dependency parsers in several tasks in the field of NLP. We also hope, this further increases the interest in research for better features.

Appendix A

An appendix

The graphs below show the performance of various parsers with the 10 runs on the validation dataset.

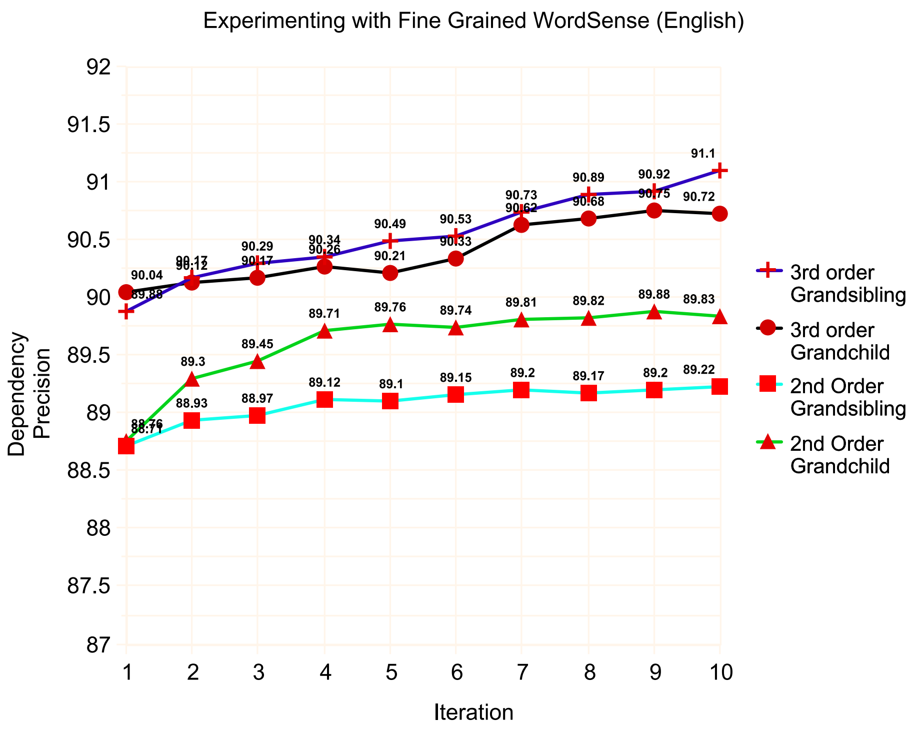

Figure A.1: Result for experimentation with fine-grained wordsenses with English

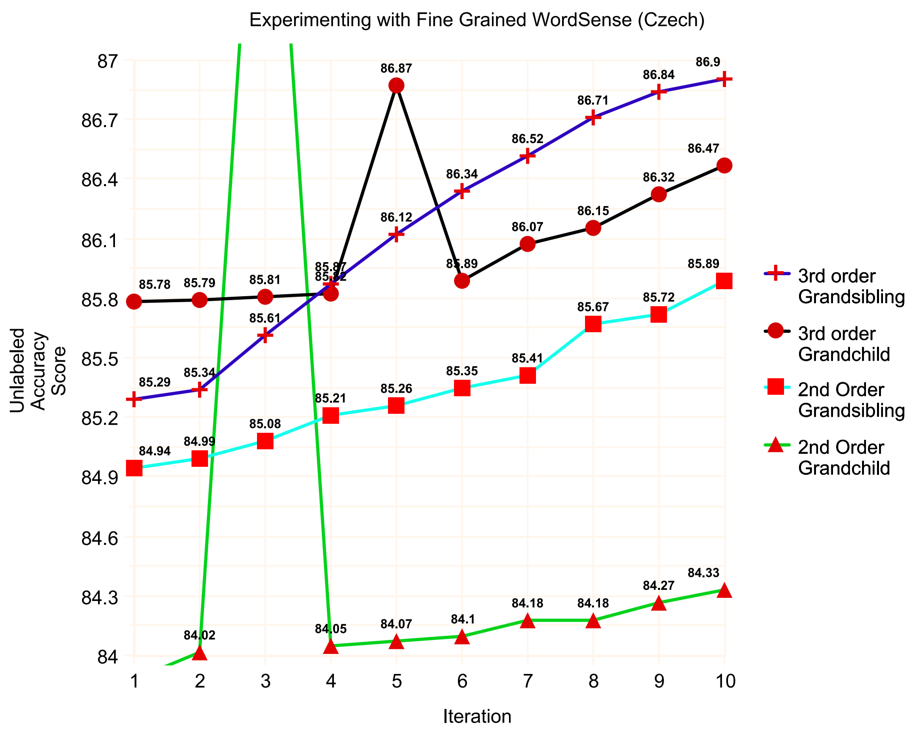

Figure A.2: Result for experimentation with fine-grained wordsenses with Czech

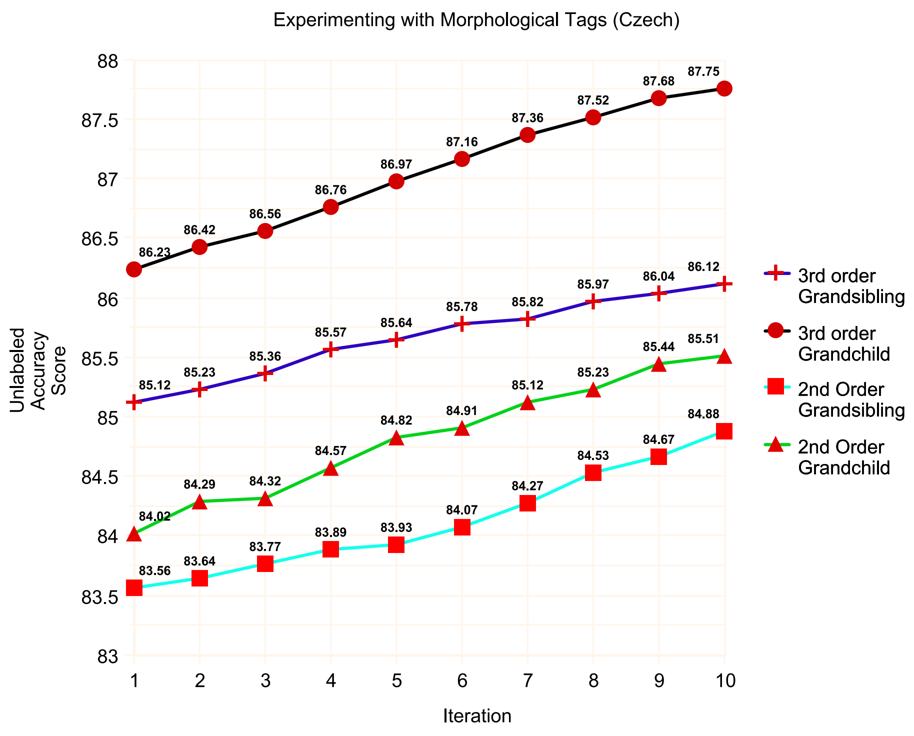

Figure A.3: Result for experimentation with coarse-grained wordsense with English

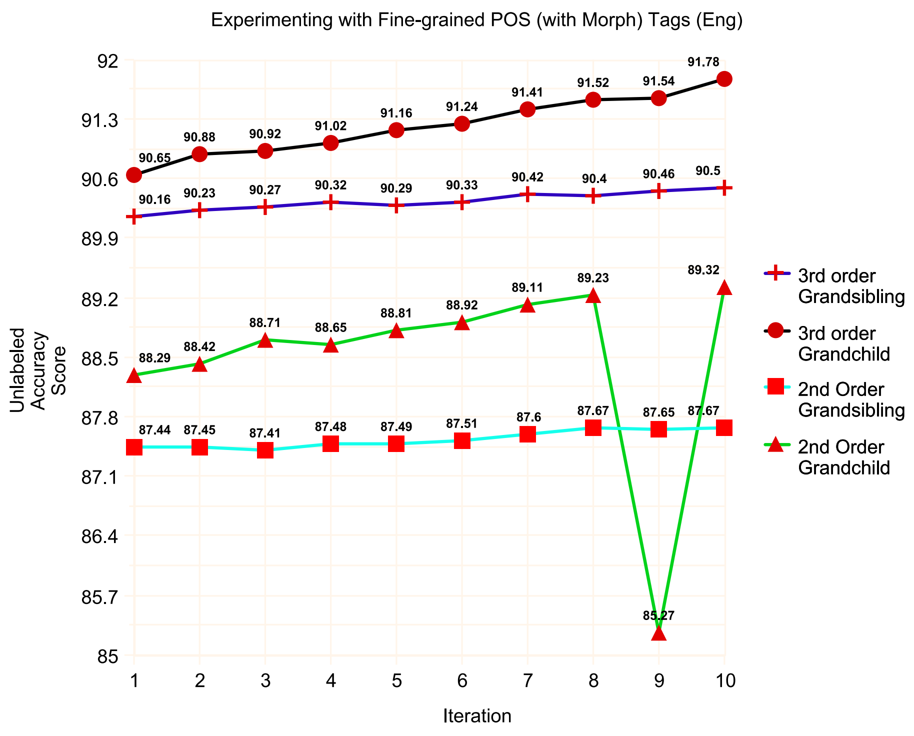

Figure A.4: Result for experimentation with morphological tags for Czech

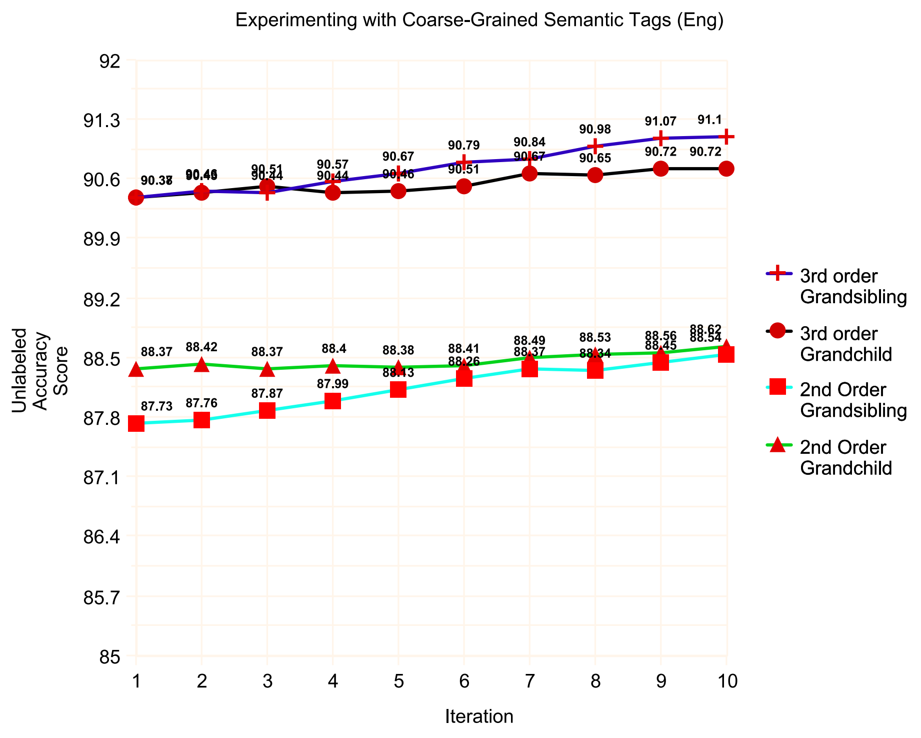

Figure A.5: Result for experimentation with morphological tags for English

Bibliography

[Agirre et al.2011] Eneko Agirre, Kepa Bengoetxea, Koldo Gojenola, and Joakim Nivre. 2011. Improving dependency parsing with semantic classes. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 699–703, Portland, Oregon, USA, June. Association for Computational Linguistics.

[Ambati et al.2009] Bharat Ram Ambati, Pujitha Gade, Chaitanya Gsk, and Samar Husain. 2009. Effect of minimal semantics on dependency parsing. In *Proceedings of the Student Research Workshop*, pages 1–5, Borovets, Bulgaria, September. Association for Computational Linguistics.

[Ambati et al.2010] Bharat Ram Ambati, Samar Husain, Joakim Nivre, and Rajeev Sangal. 2010. On the role of morphosyntactic features in hindi dependency parsing. In *Proceedings of the NAACL HLT 2010 First Workshop on Statistical Parsing of Morphologically-Rich Languages*, pages 94–102, Los Angeles, CA, USA, June. Association for Computational Linguistics.

[Bangalore et al.2009] Srinivas Bangalore, Pierre Boulllier, Alexis Nasr, Owen Rambow, and Benoît Sagot. 2009. Mica: a probabilistic dependency parser based on tree insertion grammars application note. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers*, NAACL-Short '09, pages 185–188, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Bikel2004a] Daniel M. Bikel. 2004a. *On the parameter space of generative lexicalized statistical parsing models*. Ph.D. thesis, University of Pennsylvania, Philadelphia, PA, USA. AAI3152016.

[Bikel2004b] Daniel M. Bikel. 2004b. *On the parameter space of generative lexicalized statistical parsing models*. Ph.D. thesis, University of Pennsylvania, Philadelphia, PA, USA. AAI3152016.

[Böhmová et al.2001] Alena Böhmová, Jan Hajič, Eva Hajičová, and Barbora Hladká. 2001. The prague dependency treebank: Three-level annotation scenario. In Anne Abeillé, editor, *Treebanks: Building and Using Syntactically Annotated Corpora*. Kluwer Academic Publishers.

[Bourdon et al.1998] Marie Bourdon, Lyne Da Sylva, Michel Gagnon, Alma KHARRAT, Sonja Knoll, and Anna MACLACHLAN. 1998. A case study in implementing dependency-based grammars.

[Buchholz and Marsi2006] Sabine Buchholz and Erwin Marsi. 2006. Conll-x shared task on multilingual dependency parsing. In *Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL-X)*, pages 149–164, New York City, June. Association for Computational Linguistics.

[Carreras et al.2006] Xavier Carreras, Mihai Surdeanu, and Lluís Màrquez. 2006. Projective dependency parsing with perceptron. In *Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL-X)*, pages 181–185, New York City, June. Association for Computational Linguistics.

[Carreras et al.2008] Xavier Carreras, Michael Collins, and Terry Koo. 2008. Tag, dynamic programming, and the perceptron for efficient, feature-rich parsing. In *CoNLL 2008: Proceedings of the Twelfth Conference on Computational Natural Language Learning*, pages 9–16, Manchester, England, August. Coling 2008 Organizing Committee.

[Carreras2007] Xavier Carreras. 2007. Experiments with a higher-order projective dependency parser. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, pages 957–961, Prague, Czech Republic, June. Association for Computational Linguistics.

[Chomsky1956] N. Chomsky. 1956. Three models for the description of language. *IEEE Trans. Information Theory*, 2(3):113– 124.

[Cocke and Schwartz1970] John Cocke and Jacob T. Schwartz. 1970. Programming languages and their compilers. Technical report, Courant Institute, NYU, April. Preliminary notes.

[Collins2002] Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*, pages 1–8. Association for Computational Linguistics, July.

[Covington2001] Michael A. Covington. 2001. A fundamental algorithm for dependency parsing. In *In Proceedings of the 39th Annual ACM Southeast Conference*, pages 95–102.

[Eisner and Satta1999] Jason Eisner and Giorgio Satta. 1999. Efficient parsing for bilexical context-free grammars and head-automaton grammars. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 457–464, University of Maryland, June.

[Eisner and Smith2010] Jason Eisner and Noah A. Smith. 2010. Favor short dependencies: Parsing with soft and hard constraints on dependency length. In Harry Bunt, Paola Merlo, and Joakim Nivre, editors, *Trends in Parsing Technology: Dependency Parsing, Domain Adaptation, and Deep Parsing*, chapter 8, pages 121–150. Springer.

[Eisner2000] Jason Eisner. 2000. Bilexical grammars and their cubic-time parsing algorithms. In Harry Bunt and Anton Nijholt, editors, *Advances in Probabilistic and Other Parsing Technologies*, pages 29–62. Kluwer Academic Publishers, October.

[Fellbaum1998a] Christiane Fellbaum. 1998a. A semantic network of english: The mother of all wordnets. *Computers and the Humanities*, 32(2-3):209–220.

[Fellbaum1998b] Christiane Fellbaum. 1998b. Towards a representation of idioms in wordnet. In *In Proceedings of the workshop on the Use of WordNet in Natural Language Processing Systems (Coling-ACL*, pages 52–57.

[Hajič et al.2009] Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, Pavel Straňák, Mihai Surdeanu, Nianwen Xue, and Yi Zhang. 2009. The conll-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL 2009): Shared Task*, pages 1–18, Boulder, Colorado, June. Association for Computational Linguistics.

[Hektoen1997] E. Hektoen. 1997. Probabilistic parse selection based on semantic cooccurrences. In *5th International workshop on parsing technologies (IWPT-97)*, pages 113–122.

[Horák et al.2007] Aleš Horák, Karel Pala, Marie Duží, and Pavel Materna. 2007. Verb valency semantic representation for deep linguistic processing. In *ACL 2007 Workshop on Deep Linguistic Processing*, pages 97–104, Prague, Czech Republic, June. Association for Computational Linguistics.

[Johansson and Nugues2007] Richard Johansson and Pierre Nugues. 2007. Extended constituent-to-dependency conversion for English. In *Proceedings of NODALIDA 2007*, pages 105–112, Tartu, Estonia, May 25-26.

[Joshi1969] Aravind K. Joshi. 1969. Properties of formal grammars with mixed types of rules and their linguistic relevance. In *Proceedings of the 1969 conference on Computational linguistics*, COLING '69, pages 1–18, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Kasami1965] T. Kasami. 1965. An efficient recognition and syntax algorithm for context-free languages. Technical Report AFCLR-65-758, Air Force Cambridge Research Laboratory, Bedford, MA.

[Kitagawa and Tanaka-Ishii2010] Kotaro Kitagawa and Kumiko Tanaka-Ishii. 2010. Tree-based deterministic dependency parsing — an application to nivre's method —. In *Proceedings of the ACL 2010 Conference Short Papers*, pages 189–193, Uppsala, Sweden, July. Association for Computational Linguistics.

[Koo and Collins2010] Terry Koo and Michael Collins. 2010. Efficient third-order dependency parsers. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1–11, Uppsala, Sweden, July. Association for Computational Linguistics.

[Koo et al.2008] Terry Koo, Xavier Carreras, and Michael Collins. 2008. Simple semi-supervised dependency parsing. In *Proceedings of ACL-08: HLT*, pages 595–603, Columbus, Ohio, June. Association for Computational Linguistics.

[Kuboň et al.1998] Vladislav Kuboň, Tomáš Holan, Karel Oliva, and Martin Plátek. 1998. Two useful measures of word order complexity. In *Proceedings of the Dependency-Based Grammars Workshop, the COLING - ACL Conference.*

[Marcus et al.1993] Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: the penn treebank. *Comput. Linguist.*, 19:313–330, June.

[Marneffe et al.2007] Marie Catherine De Marneffe, Trond Grenager, Bill Maccartney, Daniel Cer, Daniel Ramage, Chloé Kiddon, and Christopher D. Manning. 2007. Aligning semantic graphs for textual inference and machine reading. In *In Proc. of the AAAI Spring Symposium at.*

[McDonald and Nivre2007] Ryan McDonald and Joakim Nivre. 2007. Characterizing the errors of data-driven dependency parsing models. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 122–131, Prague, Czech Republic, June. Association for Computational Linguistics.

[McDonald et al.2005] Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajic. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 523–530, Vancouver, British Columbia, Canada, October. Association for Computational Linguistics.

[McDonald et al.2006] Ryan McDonald, Kevin Lerman, and Fernando Pereira. 2006. Multilingual dependency analysis with a two-stage discriminative parser. In *Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL-X)*, pages 216–220, New York City, June. Association for Computational Linguistics.

[Mel'čuk et al.1987] I.A. Mel'čuk, N.V. Pertsov, and R. Kittredge. 1987. *Surface syntax of English: a formal model within the meaning-text framework*. Linguistic & literary studies in Eastern Europe. John Benjamins Pub. Co.

[Merlo et al.2011] Paola Merlo, Harry Bunt, and Joakim Nivre. 2011. Current trends in parsing technology. In Nancy Ide, Jean Véronis, Harry Bunt, Paola Merlo, and Joakim Nivre, editors, *Trends in Parsing Technology*, volume 43 of *Text, Speech and Language Technology*, pages 1–17. Springer Netherlands. 10.1007/978-90-481-9352-3_1.

[Nilsson et al.2006] Jens Nilsson, Joakim Nivre, and Johan Hall. 2006. Graph transformations in data-driven dependency parsing. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 257–264, Sydney, Australia, July. Association for Computational Linguistics.

[Nivre and Hall2005] Joakim Nivre and Johan Hall. 2005. Maltparser: A language-independent system for data-driven dependency parsing. In *In Proc. of the Fourth Workshop on Treebanks and Linguistic Theories*, pages 13–95.

[Nivre and McDonald2008] Joakim Nivre and Ryan McDonald. 2008. Integrating graph-based and transition-based dependency parsers. In *Proceedings of ACL-08: HLT*, pages 950–958, Columbus, Ohio, June. Association for Computational Linguistics.

[Nivre et al.2007a] Joakim Nivre, Johan Hall, Sandra Kübler, Ryan McDonald, Jens Nilsson, Sebastian Riedel, and Deniz Yuret. 2007a. The CoNLL 2007 shared task on dependency parsing. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, pages 915–932, Prague, Czech Republic, June. Association for Computational Linguistics.

[Nivre et al.2007b] Joakim Nivre, Johan Hall, Sandra Kübler, Ryan McDonald, Jens Nilsson, Sebastian Riedel, and Deniz Yuret. 2007b. The CoNLL 2007 shared task on dependency parsing. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, pages 915–932, Prague, Czech Republic, June. Association for Computational Linguistics.

[Novák and Žabokrtský2007] Václav Novák and Zdeněk Žabokrtský. 2007. Feature engineering in maximum spanning tree dependency parser. In *Proceedings of the 10th International Conference on Text, Speech and Dialogue*, pages 92–98.

[Pedersen and Kolhatkar2009] Ted Pedersen and Varada Kolhatkar. 2009. WordNet::SenseRelate::AllWords - a broad coverage word sense tagger that maximizes semantic relatedness. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Demonstration Session*, pages 17–20, Boulder, Colorado, June. Association for Computational Linguistics.

[Popel et al.2011] Martin Popel, David Mareček, Nathan Green, and Zdenek Zabokrtsky. 2011. Influence of parser choice on dependency-based mt. In *Proceedings of the Sixth*

Workshop on Statistical Machine Translation, pages 433–439, Edinburgh, Scotland, July. Association for Computational Linguistics.

[Ratnaparkhi et al.1994] Adwait Ratnaparkhi, Jeff Reynar, and Salim Roukos. 1994. A maximum entropy model for prepositional phrase attachment. In *Proceedings of the workshop on Human Language Technology*, HLT '94, pages 250–255, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Sagae and Lavie2006] Kenji Sagae and Alon Lavie. 2006. Parser combination by reparsing. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, pages 129–132, New York City, USA, June. Association for Computational Linguistics.

[Sagae and Tsujii2007] Kenji Sagae and Jun'ichi Tsujii. 2007. Dependency parsing and domain adaptation with LR models and parser ensembles. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, pages 1044–1050, Prague, Czech Republic, June. Association for Computational Linguistics.

[Sgall1984] Petr Sgall. 1984. *Contributions to Functional Syntax, Semantics and Language Comprehension*. Benjamins/Academia, Amsterdam, Netherlands/Academia, Czech Republic.

[Shimizu2007] Nobuyuki Shimizu. 2007. Structural correspondence learning for dependency parsing. In *In Proc.*

[Sleator and Temperley1993] D. D. Sleator and D. Temperley. 1993. Parsing english with a link grammar. In *Third International Workshop on Parsing Technologies*.

[Song et al.2011] Yang Song, Houfeng Wang, and Jing Jiang. 2011. Link type based pre-cluster pair model for coreference resolution. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task*, pages 131–135, Portland, Oregon, USA, June. Association for Computational Linguistics.

[Surdeanu et al.2008] Mihai Surdeanu, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. 2008. The conll 2008 shared task on joint parsing of syntactic and semantic dependencies. In *CoNLL 2008: Proceedings of the Twelfth Conference on Computational Natural Language Learning*, pages 159–177, Manchester, England, August. Coling 2008 Organizing Committee.

[Tesnière1959] Lucien Tesnière. 1959. *Éleménts de syntaxe structurale*. Klincksieck, Paris.

[Titov and Henderson2007] Ivan Titov and James Henderson. 2007. A latent variable model for generative dependency parsing. In *Proceedings of the Tenth International Conference on Parsing Technologies*, pages 144–155, Prague, Czech Republic, June. Association for Computational Linguistics.

[Xiong2005] Ziyu Xiong. 2005. Downstep effect on disyllabic words of citation forms in standard chinese. In *INTERSPEECH*, pages 1393–1396.

[Yamada and Matsumoto2003] Hiroyasu Yamada and Yuji Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *In Proceedings of IWPT*, pages 195–206.