Charles University in Prague
Faculty of Mathematics and Physics
Institute of Formal and Applied Linguistics

Unsupervised Dependency Parsing

David Mareček

DOCTORAL THESIS

Prague, 2012

Supervisor: Doc. Ing. Zdeněk Žabokrtskyý, Ph.D.
Charles University in Prague
Faculty of Mathematics and Physics
Institute of Formal and Applied Linguistics
Malostranské náměstí 25
118 00 Prague 1
Czech Republic

Opponents: Ing. Filip Jurčíček, Ph.D.
Charles University in Prague
Faculty of Mathematics and Physics
Institute of Formal and Applied Linguistics
Malostranské náměstí 25
118 00 Prague 1
Czech Republic

Anders Søgaard, Ph.D.
University of Copenhagen
Faculty of Humanities
Department of Scandinavian Studies and Linguistics
Njalsgade 120
2300 København S
Denmark

Acknowledgements

I would like to thank my supervisor Zdeněk Žabokrtský, for his guidance, enthusiasm and encouragement. Without his help, I would probably have been doing some rather unnecessary experiments and I would never have been able to finish my dissertation.

I would also like to thank my colleagues Pavel Pecina, Martin Popel and Ondřej Dušek, who helped me when I was finishing this thesis and gave me many valuable suggestions how to improve it so that it would be more understandable.

I would also like to thank my parents and my sister. They were always supporting me and encouraging me with their best wishes.

Finally, I would like to thank my girlfriend Pavlína for her patience. She was always there cheering me up and stood by me through the good times and bad times.

Abstract

Unsupervised dependency parsing is an alternative approach to identifying relations between words in a sentence. It does not require any annotated treebank, it is independent of language theory and universal across languages. However, so far quite low parsing quality is its main disadvantage.

This thesis discusses some previous works and introduces a novel approach to unsupervised parsing. Our dependency model consists of four submodels: (i) edge model, which controls the distribution of governor-dependent pairs, (ii) fertility model, which controls the number of node's dependents, (iii) distance model, which controls the length of the dependency edges, and (iv) reducibility model. The reducibility model is based on a hypothesis that words that can be removed from a sentence without violating its grammaticality are leaves in the dependency tree.

Induction of the dependency structures is done using Gibbs sampling method. We introduce a sampling algorithm that keeps the dependency trees projective, which is a very valuable constraint.

In our experiments across 30 languages, we discuss the results of various settings of our models. Our method outperforms the previously reported results on a majority of the test languages.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Inducing linguistic structure of a natural language text is one of the essential tasks of natural language processing and has received a great deal of attention since the beginnings of computational linguistics. A solution to this problem is often expected to bring a significant improvement to variety of linguistic applications, such as machine translation (Mareček et al., 2010) or question answering (Cui et al., 2005). However, this problem remains still open and the question is to what extent it is possible to replace the human world experience by a large number of raw texts for induction of relations between individual objects in a given sentence.

The current state-of-the-art methods for natural language parsing are based on supervised machine learning. A supervised learner needs a treebank (a set of training examples that consists of sentences with manually annotated structures), based on which it learns a grammar which is then used to parse new (previously unseen) sentences. The main disadvantage of this approach is that the development of such treebanks is very expensive and time-consuming. Moreover, one treebank is connected with a particular language and domain and being able to parse a different language requires to develop a new treebank.

Unsupervised parsing approaches have received a considerably growing attention in the last decade. The quality of their results is still far below the supervised approaches, but their indisputable advantage is the fact that no annotated treebanks are needed and induced structures are not burdened by any linguistic conventions. If they were to equal supervised parsers in accuracy one day, they would inherit all the applications supervised parsers have. Even if their accuracy was lower, they could substitute the supervised ones because of their language and linguistic theory independence.

1.1 Relation to Language Acquisition

Unsupervised parsing (or grammar induction) has much in common with several other areas, including psycholinguistics and cognitive science (Kwiatkowski et al., 2012). We would like to mimic children in learning their mother tongue. The utterances of their parents provide a set of positive examples, based on which the children can learn grammar and generate new utterances. Parents can correct their erroneously generated sentences and thus provide a negative feedback (Marcus, 1993; Penner, 1987). A weaker form of negative feedback is unsuccessful communication. If a child does not get what it wants, it tries to reformulate the utterance so that the listener understands.

The negative feedback is not available for unsupervised parsers. This may be partially compensated by a huge amount of positive examples. For some languages, there are text corpora available whose size is measured in "gigawords", i.e a much greater amount than what one can read in a lifetime.[1] In case a particular phenomenon is very sparse in the data, we can assume that it is not grammatically correct. One approach to such negative-feedback simulation using the large corpus is also described in this thesis in Section 5.4.

Another, probably more important thing that people have and machines do not is the experience with entities in the world and imagination. People know that the phrase "red apples and bananas" does not mean that the bananas are red and the phrase "a girl with a telephone bought a week ago" does not mean that the girl was bought a week ago. Here again, we must rely on a large corpus and hope that there are no "red bananas" in it and the majority of bananas are "yellow" and that there are much more "bought telephones" than "bought girls".

1.2 Motivations for Unsupervised Parsing

The first motivation for the development of unsupervised parsing techniques is obvious: They do not rely on the availability of manually annotated data. Although there are many treebanks,[2] many more languages remain uncovered. Moreover, since the treebanks were often developed independently at various places, they differ very often in underlying linguistic formalisms and data formats and/or use completely different labels for part-of-speech tags, constituents, and dependencies. Consequently, linguistic tools working on one treebank cannot be easily extended to other languages simply by adding new treebanks. An experiment on harmonizing all the available treebanks (Zeman et al., 2012) showed that automatic transformation between different annotation styles cannot be lossless, as various kinds of linguistic information expressed in one annotation style often cannot be captured in another style.[3] Examples of such discrepancies are given in Section 4.3.

Another problem of treebanks is their specific domain. For example, the English Penn Treebank (Marcus et al., 1994) consists of newspaper articles. Parsers trained on Penn Treebank achieve very good results on held-out data from the very same

[1]Koehn (2009) mentions that people do not read more than 10,000 words per day, which is 300 million words in their lifetime.

[2]We have collected more than 30 available treebanks of different languages (Zeman et al., 2012)

[3]For example, annotation of coordination structures belongs to the most problematic issues. They are expressed at least by ten different ways. See Figure 4.1.

domain, but when they are used to parse books, their accuracy goes down.[4]

The last motivation is the most challenging one. The question how children learn their mother tongue and how people parse a sentence in their mind has fascinated many researchers in different fields. What we know for sure is that children do not study treebanks and annotation manuals when learning their language. This provokes the following question: What if the structures in the treebanks proposed by linguists are not suitable for statistical language tools? For example, positions of function words in a dependency tree, such as prepositions, conjunctions, articles, or auxiliary verbs, differ across various treebanks. If we want to learn how these structures should look like from the purely statistical point of view, the only possibility is to employ a completely unsupervised parser with no language-dependent prior knowledge.

1.3 Dependency and Constituency

In the world of natural language parsing, there are two main types of linguistic structures: phrase-structure (constituency) trees and dependency trees.

The phrase structure (Chomsky, 2002) consists of nonterminal symbols representing particular constituent phrases, e.g. noun phrase (NP) for *"a warm climate"* or prepositional phrase (PP) for *"for their winter excursions"*, and terminal symbols in leaves representing the individual words. See Figure 1.1. A set of phrase structure trees can be represented by a context-free grammar.

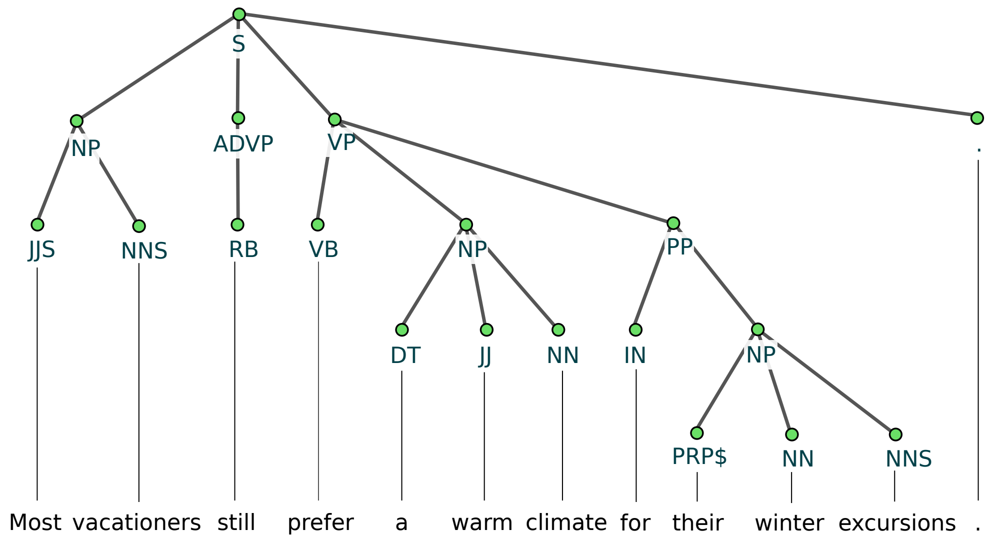

S
NP
ADVP
VP
.
JJS
NNS
RB
VB
NP
PP
DT
JJ
NN
IN
NP
PRP$
NN
NNS
Most vacationers still prefer a warm climate for their winter excursions .

Figure 1.1: A constituency tree.

[4]We could say that linguistic tools trained on Penn Treebank work well mainly on the Wall Street Journal texts. Stephan Oepen described this phenomenon as "Wall Street Journal science" at his invited talk at the Unified Linguistic Annotation Workshop 2007 in Bergen, Norway.

In a dependency tree (Sgall et al., 1986) (see Figure 1.2), every node represents one word in the sentence and edges represent dependency relations between the words. Unlike in constituency trees, we can directly see that the word "warm" is a modifier of the word "climate". Another advantage of dependency trees is the fact that they can easily capture so-called *non-projective* dependencies (see the definition in Section 1.4.3). Such dependencies would correspond to discontinuous phrases in phrase-structure trees, which are not allowed and must be solved in a rather complicated way using traces (Marcus et al., 1994).

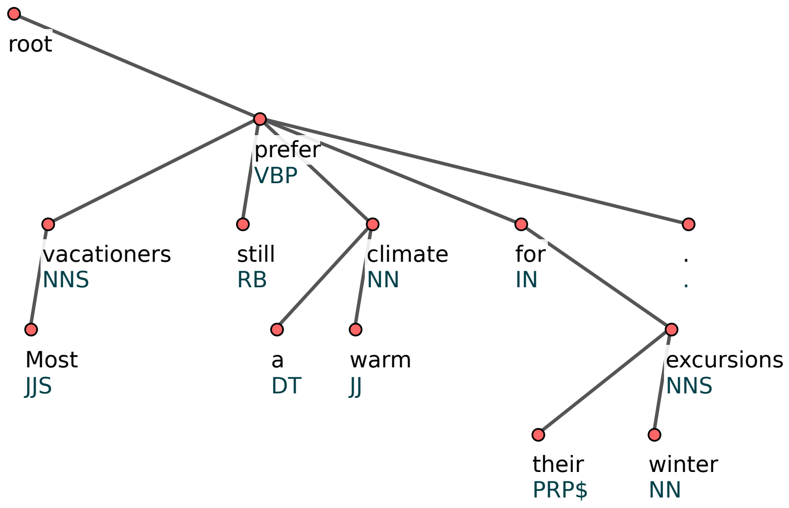


Figure 1.2: A dependency tree.

The unsupervised parsing approaches have been developed both for phrase-structure and dependency grammars. However, it seems that the dependency approaches predominate in the last years even for English, which has a long tradition of phrase-structure grammars. The main motivation for using sentence structures in natural language processing is to enable extraction of lexical dependencies. For example, we need to detect the arguments and modifiers of a given word. Dependency trees are more suitable for this purpose. Many unsupervised constituency parsers induce only phrases (i.e. bracketing). However, the absence of information about the types of particular phrases causes that head-modifier pairs cannot be extracted, as we do not know the governing words (heads) of the phrases.

Dependency context-free trees (Klein and Manning, 2004), depicted in Figure 1.3 are a mix between constituency and dependency trees. Their nonterminals have the same labels as the head terminals in the respective phrases. Thus, instead of inducing a label for each phrase, we try to find its head, which is the same problem we solve when we want to induce dependency trees. Each projective dependency tree can be simply converted to a context-free dependency tree.

In this thesis, we are concerned with the induction of dependency trees only.

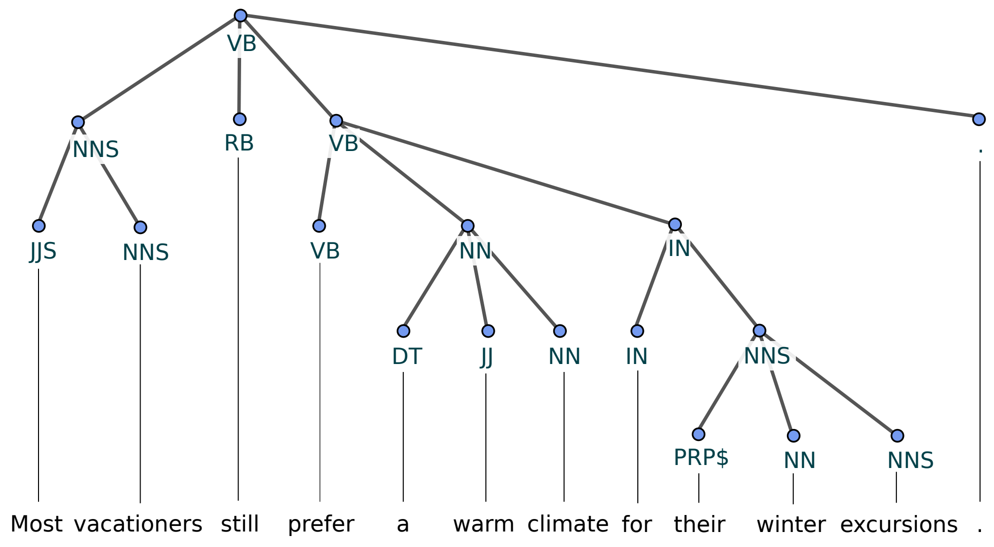

VB
NNS
RB
VB
.
JJS
NNS
VB
NN
IN
DT
JJ
NN
IN
NNS
PRP$
NN
NNS
Most vacationers still prefer a warm climate for their winter excursions .

Figure 1.3: A dependency context-free tree.

1.4 Basic Definitions

In this section, we informally define several basic terms that are used throughout this thesis. Some of them are commonly known, some of them are specific for this work.

1.4.1 Corpora

Raw corpus: By a *raw corpus*, we mean an unlabeled collection of texts written in one language. We suppose that the texts are already automatically tokenized and segmented into sentences. The tokenization and segmentation are done using very simple rules in the form of regular expressions, in order to resemble the tokenization used in testing treebanks as closely as possible. Section 7.2.1 describes the tokenization in more detail.

PoS tagged corpus: A *PoS tagged corpus* is a corpus in which a part-of-speech (PoS) tag is assigned to every word. Tags can be assigned manually or automatically by a supervised tagger trained on another manually PoS tagged data. Automatically assigned word classes induced in an unsupervised way (see Section 7.2.2) can also be used in place of PoS tags.

Word n-gram: A *word n-gram* is a continuous sequence of n words in a corpus. We call it *word n-gram* instead of a simple *n-gram* in order to be able to distinguish it from a *PoS n-gram*.

PoS n-gram: Analogously, a *PoS n-gram* is a sequence of n part-of-speech tags.

1.4.2 Tree Structure

We use definitions similar as in (Havelka, 2007):

Dependency tree: A *dependency tree* is a triple $(V, \rightarrow, \preceq)$, where V is a finite set of nodes, $\rightarrow$ is a dependency relation on V and $\preceq$ is a total order on V. Relation $\rightarrow$ models linguistic dependency, and represents a directed, rooted tree on V. In surface syntax, $V = W \cup \{root\}$, where each node in W corresponds to one word in the sentence and the order $\preceq$ corresponds to the word order in the sentence. The *root* node is an artificial root of the dependency tree. The *root* is formally the first node in dependency tree $(\forall w \in W : root \preceq w)$. Relation $\rightarrow^*$ is the transitive closure of $\rightarrow$ and is usually called *subordination.*

Rooted subtree: A *rooted subtree* S_i of a dependency tree $T = (V, \rightarrow, \preceq)$ is a set of nodes subordinated by $i \in V$ including i, i.e. $S_i = \{v \in V;\ i \rightarrow^* v\} \cup \{i\}$. In other words, a *rooted subtree* is a set of all descendants of a node including the node itself.

1.4.3 Projectivity

We will use the definitions of tree projectivity introduced by Harper and Hays (1959):

Projective dependency edge: Let us define a set $\{i, \ldots, j\}$ as a set of nodes between i and j, including i and j. A dependency edge $i \rightarrow j$ is *projective* if and only if $\forall v \in V : v \in \{i, \ldots, j\} \implies v \in S_i$. All words between the words i and j must be descendants of i.

Projective dependency tree: A dependency tree $T = (V, \rightarrow, \preceq)$ is *projective* if and only if all its edges are projective. See the example in Figure 7.1.

1.4.4 Dependency Treebanks

Dependency Treebank: A syntactically annotated corpus, i.e. a corpus in which a dependency structure is provided for each sentence, is called a *dependency treebank.*

Manually annotated dependency treebank: *Manually annotated dependency treebank* is a dependency treebank, in which the dependency trees were build manually by linguists using a common annotation manual.

1.5 Unsupervised and Semi-supervised Learning

The term "unsupervised learning" refers to the problem of finding hidden structures or patterns in unlabeled data. That is, it does not need any labeled examples – parse trees in our case. But of course, we cannot just insert any text (sequence of characters) into a "magic box" and expect that it would return some meaningful structures. It is necessary to define some basic properties of the structures we want to derive. Ideally, we would like to induce a grammar – a minimal description of a given language that would be able to generate all possible texts in this language. But this is a hard problem since we have relatively few examples to derive it. Therefore, we must go further and introduce our basic assumptions we have about the structures we want to obtain.

Assumption (a) The structure of a sentence is a dependency tree with nodes corresponding to individual words in the sentence.

This is a very strong assumption. Why not constituency trees? Why do we not split words into morphemes? What if cycles are needed? We are aware of all the problems with dependency tree structures, such as capturing coordinations, complements, or anaphora. However, we do not have any other general structure that would be simple enough and still reflect the main property of sentences, which is recursivity.

Assumption (b) Types of dependencies (e.g. pairs of governing and dependent words) tend to be repeated in the treebank.

This refers to the minimum description principle. A particular word can depend only on a small subset of all possible words.

Assumption (c) The structure of a sentence is a projective dependency tree (see Section 5.6).

Assumption (d) Dependency edges between words are rather short than long. Most dependency relations tend to occur between adjacent words.

Assumption (e) Words that can be removed from a sentence without violating its grammaticality are often leaves in the dependency tree.

All these statements may serve as the basic assumptions for induction of linguistic structure. In unsupervised parsing, we do not allow ourselves to use any kind of linguistic rules, such as

Assumption (f) Roots of dependency trees are often verbs.

Assumption (g) Adjectives very often depends on nouns.

Assumption (h) The English word “the” is always a leaf.

Why can we not use rules like this? Such rules are in fact also examples. Imagine that we have thousands of such rules. Then they could have similar power as treebanks. In supervised parsing, people use linguistic rules for the development of a treebank, and the rules together with other language properties are then learned from the treebank. Rule-based parsing applies the same linguistic rules but in a direct way.

If we have a small number of linguistic rules available, we speak about *minimally supervised* or *semi-supervised* learning. Semi-supervised parsing methods are based on several basic linguistic rules and big unlabeled data from which more subtle language features are learned.

The boundary between "not allowed" linguistic rules and "allowed" basic assumptions about dependency trees is very fuzzy. For example, we could say that the preference of short dependencies (Assumption (d)) is also a kind of rule. We define the linguistic assumptions allowed for unsupervised parsing as those that are independent of language and tag set. We cannot apply e.g. a rule which says that left attachments are more probable than right attachments as it holds only for some languages and the opposite is true for others.

Similarly, we cannot state that the most frequent part-of-speech (PoS) tag in the data is the PoS tag for nouns (Mareček and Žabokrtský, 2011), because it does not hold generally for all possible tag sets. In some tag sets, nouns could be for example subcategorized in more detail an such rule would not hold then. Moreover, the parser should work also on unsupervised PoS tags (word classes), where we cannot assume any similar characteristics. If we were able to induce some universal, language-independent PoS tags in an unsupervised way, at least to recognize nouns, verbs, adjectives, and adverbs, we could directly apply the linguistic rules saying that adjectives depend on nouns, nouns depend on verbs, etc. However, induction of universal PoS tags seems to be even harder problem than induction of structures. It is even questionable whether all languages have these four types of words (Evans and Levinson, 2009).

Another problem in unsupervised learning is the parser tuning. Assume that we develop a model, apply it on some data, and look at the resulting trees. Then we try another model, look at the trees, and see that the trees look better, so we proclaim the second model as the better one. But what are "better trees"? Probably the trees we know from a manually annotated treebank. If so, we can directly evaluate our parser on a testing part of our treebank. This is common practice in unsupervised parser evaluation if we have no better method[5] to compare the quality of different parsers. However, this yields a contradiction since we use manually annotated data and therefore the parsers become a bit supervised. This would be a problem in case we report results only on languages on which we have tuned our parser. If we show that even if the parameters of our parser were tuned only on one language and the

[5]Some kind of extrinsic evaluation using a final application would be much better. This is however beyond the scope of this thesis.

parser had good results on a variety of other languages as well, we can proclaim our approach as "enough" unsupervised.

1.6 Goals of the Thesis

Our task is to improve the unsupervised induction of dependency structures. Current state-of-the-art systems work (Spitkovsky et al., 2011c) quite well for English and for several other languages. However, there are languages on which they fail completely. For example, they have problems even with very basic dependencies such as attachments of adjectives to nouns. Since one of the motivations of unsupervised parsing is its applicability to any language, we will evaluate our parser on 30 different languages.

Our goal is to develop a new approach to the induction of dependency trees. We will exploit several new features, such as reducibility (dependent word can be removed from a sentence without violating its grammatical correctness) or fertility (number of children is determined by the head word). The most probable dependency trees will be induced using the Gibbs sampling technique (Gilks et al., 1996).

1.6.1 Unsupervised Dependency Parsing using Supervised PoS Tags

A purely unsupervised approach to dependency parsing algorithm should use only a raw corpus without any annotations or external tools that employ any type of linguistics annotation. To avoid sparsity issue associated with word forms, we make use of part-of-speech tags. However, the part-of-speech tags come from manually annotated corpora and bring in an element of human effort and decision making. The tagset choice can greatly affect the behavior of the dependency induction tool. Morphological disambiguation of an ambiguous word (e.g. the decision whether a word "hits" is a noun or a verb) directly predicts its syntactic position.

Nevertheless, many works show that tag sets developed by linguists are very useful (Blunsom and Cohn, 2010). The first task of this thesis is to develop an unsupervised dependency parser for a given language if we have a manually part-of-speech tagged corpus available for this given language.

1.6.2 Unsupervised Dependency Parsing without Supervised PoS Tags

Some recent works experiment with unsupervised dependency grammar using no supervised PoS tags. Instead, they make use of unsupervised PoS tags – automatically induced word classes (Spitkovsky et al., 2011a). Such solution is purer, more flexible, and tagset independent. The unsupervised methods for inducing word classes are beyond the scope of this thesis, but all the parsing methods developed in the first task will be tested also on automatically induced PoS tags using a publicly available word-clustering tool (Section 7.2.2).

1.7 Structure of the Thesis

The remainder of the thesis is structured as follows. Chapter 2 briefly outlines the history and state of the art in unsupervised dependency parsing and introduces related work. Chapter 3 describes the theoretical background of methods used in this work. In Chapter 4, we describe the data used for our experiments and discuss the possible evaluation methods.

Our own contribution begins from Chapter 5, in which we introduce our method of modeling dependency trees and its variants. The probability estimates based on these models are then used in dependency tree sampling and decoding, which is described in Chapter 6. All experiments and parsing results across various languages are summarized in Chapter 7. Chapter 8 concludes the thesis.

Chapter 2

Related Work

2.1 Beginnings of Unsupervised Parsing

The ability of inducing a grammar (or any relations between words) from a raw text has been a major goal for many researchers since the very beginning of computational linguistics. The first simple approaches were based on computing mutual information between words in the text (van der Mude and Walker, 1978; Magerman and Marcus, 1990).

There were early efforts in developing tools for the induction of phrase structure grammar or dependency grammar (Klein, 2005). In phrase structure grammar (Figure 1.1), we would expect both a tree structure and labels of nonterminal symbols for individual constituents. However, many reported approaches do not label nonterminal symbols and reduce the whole problem to the task of bracketing (Bod, 2006). The first approach to unsupervised constituents labeling was made by Borensztajn and Zuidema (2007). Another possibility is to derive names of nonterminal symbols from the heads of the respective constituents (Figure 1.3). Such a constituency grammar is then equivalent to a projective dependency grammar (Figure 1.2). In the rest of this chapter, we will be concerned only with these "context-free" dependency grammars and with the "real" dependency grammars.

The first related work we mention here is an experiment made by Carroll and Charniak (1992). They induce a probabilistic context-free grammar (PCFG). Their algorithm worked with word classes instead of words and they use a special PCFG grammar in a form of a dependency context-free grammar. Each nonterminal symbol (marked with a bar) corresponds to one word class. Rewrite rules have a nonterminal symbol $\overline{X}$ on the left side and the respective terminal symbol X on the right side together with other nonterminal symbols. An example of such a grammar follows.

$$\begin{array}{rcl@{\qquad\qquad}rcl}
S & \rightarrow & \overline{verb} & \overline{noun} & \rightarrow & \overline{adj}\ \text{noun} \\
\overline{verb} & \rightarrow & \overline{noun}\ \text{verb}\ \overline{prep} & \overline{noun} & \rightarrow & \text{noun} \\
\overline{prep} & \rightarrow & \text{prep}\ \overline{noun} & \overline{adj} & \rightarrow & \text{adj}
\end{array}$$

From the perspective of dependencies, the second rule says that a verb can have two children: a noun on the left and a preposition on the right. The last two rules say that adjectives and nouns can be leaves. Such a grammar could be also called projective dependency grammar.

Carroll and Charniak split their training corpus into two parts. From the first part, all possible rewrite rules were extracted and for each of them, the initial probability was computed. The probabilities were then tuned on the second part of the corpus using the inside-outside algorithm (Baker, 1979; Lari and Young, 1990). It is an iterative expectation-maximization (EM) algorithm, which can be described in the four following steps:

1. *initialization*: Assign initial probabilities to the rules.
2. *expectation*: Count how many times each rule could be used in the generation of the training corpus.
3. *maximization*: Update the probability estimates based on these counts.
4. Repeat the steps 2 and 3 until convergence.

The convergence of this EM process is guaranteed because after each iteration, the new estimated cross-entropy is lower than (or equal to) the previous one.

Experiments showed that the quality of the inferred grammar is very poor and it is very different from what the authors had expected. In addition, they found out that EM tends to converge to local maxima and that the outcome depends very much on the initial probabilities. When trying different random initializations, the algorithm converged to different results for each of them.

In another experiment, Carroll and Charniak introduced various restrictions on the rules, for example, the rules for rewriting adjectives or determiners to a noun were disabled. With these constraints, the inferred grammar improved. However, it is important to note that using such constraints is not a genuinely unsupervised approach and belongs rather to the category of semi-supervised or minimally-supervised approaches.

Besides the bad convergence to a global optimum, a further disadvantage of the inside-outside algorithm is its inherent computational complexity, which is $O(n^3t^3)$, where n is the total number of nonterminals and t is the length of the processed sentence. Although we have a big amount of data available for unsupervised methods, the inside-outside algorithm cannot exploit them. Paskin (2002) suggests a stronger assumption of independence for modeling dependencies. He assumes that all children (dependents) of a particular word are mutually independent and also their relative ordering is independent on their parent word. This approximation then allows a much simpler algorithm able to process a larger corpus. The time complexity of the respective simplified EM algorithm decreases to $O(n^3)$. Paskin uses only word forms in his experiments. Unfortunately, the results were also unsatisfactory. They were only slightly better than random dependency trees.

Similarly, Yuret (1998) assumes a mutual independence of edges. He computes the probability of a dependency tree as a product over all nodes' conditional probability given their parents. Maximizing such product is then equal to maximizing the product of the point-wise mutual information between parent and child in individual dependency edges. Unfortunately, this approach was also not very successful.

2.2 Dependency Model with Valence

Klein and Manning (2004) argue that conditioning the generation of a dependent only on its parent, as Paskin did, is not enough. There should be a notion of distance and valence included in the dependency model. The valence in their work is modelled very simply: the generation of a new dependent in a given direction is conditioned by its parent and by the fact whether it is the first dependent in this direction or not. They introduce a special `STOP` symbol, which is a virtual last dependent on each side of the head, denoting that no other dependent in the particular direction can be generated. This dependency model is called Dependency Model with Valence (DMV). The generative story of DMV is as follows:

- We start with the root, which is marked by the symbol "$\diamond$", and begin to generate its dependents.
- For each node, we first generate all its left dependents (one by one) and then the virtual left `STOP` symbol. We always first decide whether the `STOP` symbol will be generated or not and if not, we generate the new dependent.
- Similarly, we generate all the right dependents and then the virtual right `STOP` symbol.
- After a new node is generated, we recurse into its subtree.

During the generation, we decide at each point whether to generate a new dependent or the `STOP` symbol. This is modeled by $P_{STOP}(STOP|h, dir, adj)$. The decision is conditioned by the head h, the direction dir in which are currently generating the dependents, and the adjacency adj, which is a binary value saying whether any dependent has been already generated in this particular direction. If the `STOP` symbol is not generated, we generate a new dependent a in the direction dir according to the probability $P_{ATTACH}(a|h, dir)$. Dependents are generated conditionally on the head h and the direction dir. In the basic model by Klein and Manning, the attachment is not conditioned on adjacency. The recursive formula of computing the overall probability of a dependency tree D with a head h is shown in Equation 2.1.

$$\begin{aligned} P(D(h)) = \prod_{dir \in l,r} \prod_{a \in deps_D(h,dir)} & P_{STOP}(\neg STOP|h, dir, adj) \\ & P_{ATTACH}(a|h, dir) P(D(a)) \qquad (2.1) \\ & P_{STOP}(STOP|h, dir, adj), \end{aligned}$$

where $deps_D(h, dir)$ are all the dependents of the head h in the direction dir. It is apparent that each next dependent in a particular direction must pass a new `STOP`/$\neg$`STOP` decision. Higher numbers of dependents are therefore less probable, which is desired.

This generative schema can be described by a probabilistic context-free grammar (PCFG). Each node appears in four stages during the generation:

- $\bar{h}$ – the head h has been just generated,
- $\overset{\leftrightarrows}{h}$ – left arguments of h are being generated,
- $\overrightarrow{h}$ – right arguments of h are being generated,
- h – terminal symbol for h; all its argument have been generated.

Such PCFG includes three nonterminal symbols ($\bar{h}$, $\overset{\leftrightarrows}{h}$, $\overrightarrow{h}$) for each terminal symbol h, which is a PoS tag. An example of such PCFG is depicted in Figure 2.1.

Similarly to Carroll and Charniak (1992), the inside-outside algorithm was used for the estimation of PCFG probabilities. Klein also admits that EM easily converges to undesired local maxima and the assignment of initial probabilities (before entering the inside-outside algorithm) is very important. He introduces an ad-hoc "harmonic" completion where all non-root words take the same number of dependents and each takes other words as dependents in an inverse proportion to the distance between them. In this setting, the directed attachment score achieved 43.2% on the WSJ10[1] corpus, which was the first result breaking the left/right chain baseline.

Dependency Model with Valence became very popular and is used (with some modifications) in many current state-of-the art systems. Smith and Eisner (2005) use a contrastive estimation together with DMV. Their learner takes into account not only the observed positive examples, but also a set of similar examples that are deprecated because they could have been observed but were not. Cohen et al. (2008) use Dirichlet priors on the rewriting operations, which can encourage sparse solutions, a property which is important for grammar induction. They derive a

[1]The WSJ10 treebank is a subset of Penn Treebank (Marcus et al., 1994) consisting of sentences not longer than 10 words.

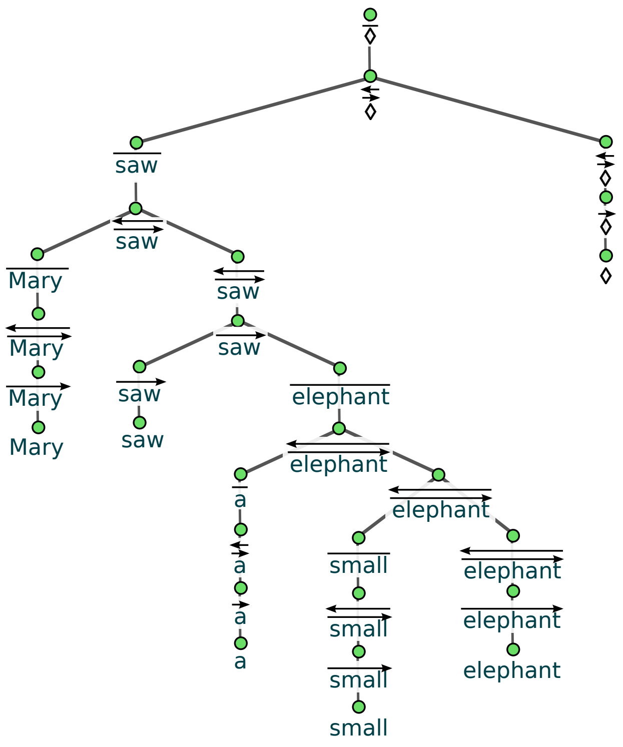


Figure 2.1: A lexicalized tree of the sentence "Mary saw a small elephant" in Dependency Model with Valence.

variational EM algorithm for the probability estimation and achieve a 59.4% directed attachment score on WSJ10.

Headden et al. (2009) extend the term of valence in DMV and call it Extended Valence Grammar (EVG). The main difference is that generating a new argument is conditioned by the fact whether it is the first one in the given direction or not. The probability $P_{ATTACH}(a|h, dir)$ is thus substituted by $P_{ATTACH}(a|h, dir, adj)$. This allows, for example, different distributions for the attachment of words "small" and "green" in the phrase "a small green apple". Another contribution of Headden et al. is the lexicalization (the generated arguments are conditioned not only the head part-of-speech but also its word form) and smoothing by interpolation:

$$P_{ATTACH}(a|h, dir, adj) = \lambda_1 P_1(a|h, dir, adj) + \lambda_2 P_2(a|dir, adj), \tag{2.2}$$

where λ_1 and λ_2 sum up to one. The PCFG rules are estimated using linearly interpolated probabilities by creating a "tied" PCFG which is extended by adding rules that select between the main distribution P_1 and the back-off distribution P_2. With these improvements, the attachment score on WSJ10 jumped almost 10% higher compared to previous results, reaching a directed attachment score of 68.9%.

Other improvements of DMV followed: Blunsom and Cohn (2010) use a tree

Description	Authors	undir. all	dir. $\leq \lvert 10 \rvert$	dir. all
Adjacent Baseline	–	53.2	33.6	25.4
Grammatical bigrams	Paskin (2001)	44.7	–	–
Simple PCFG, EM	Carroll and Charniak (1992)	39.7	–	–
DMV, EM	Klein and Manning (2004)	54.4	43.2	–
DMV, Contrastive Est.	Smith and Eisner (2005)	–	49.0	–
Dirichlet normal priors	Cohen et al. (2008)	–	59.4	40.5
EVG, lexicalization	Headden et al. (2009)	–	68.8	–
TSG DMV	Blunsom and Cohn (2010)	–	67.7	55.7
Splitting on punctuation	Spitkovsky et al. (2011b)	–	67.5	57.4
Unsupervised POS tags	Spitkovsky et al. (2011a)	–	–	59.1

Table 2.1: Directed (dir.) and undirected (undir.) attachment scores of different approaches measured on Penn Treebank. The column "dir. $\leq |10|$" shows scores measured on WSJ10, which is a subset containing sentences which are at most 10 words long.

substitution grammar which is capable of learning large dependency fragments and thereby allows for better text modelling. Spitkovsky et al. (2011b) observe a strong connection between English punctuation and phrase boundaries, split sentences at punctuation marks and impose parsing restrictions over their fragments.

For a completely unsupervised approach on dependency parsing, we should not use PoS tags. In case we have supervised PoS tags available, we could easily introduce some constraints on the dependencies, for example that a noun cannot depend on an adverb. Similar constraints were used by Carroll and Charniak (1992). However, this violates one of our main motivations, which is the independence of linguistic rules. The pure approach would be to use unsupervised PoS tags only as well. Such experiment was made by Spitkovsky et al. (2011a), who used Alexander Clark's *POSinduction* tool (Clark, 2003) for grammar induction and report better results for English than using the supervised tags.

We summarize all the results of the aforementioned methods in Table 2.1.

2.3 Other Approaches

A very interesting approach to unsupervised dependency parsing was described by Brody (2010). He formulates the parsing task as a problem of word alignment. Every sentence is aligned with itself with one constraint: no word can be attached to itself. Figure 2.2 shows one such alignment. He applied models similar to the IBM models (Brown et al., 1993), which are used for word-alignment induction: alignment model, distance model, and fertility model.

A disadvantage of this approach is the absence of the treeness constraint. The

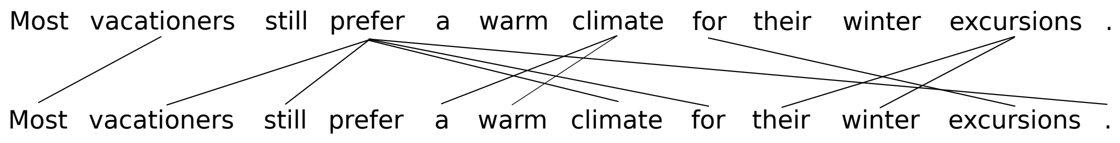

Most vacationers still prefer a warm climate for their winter excursions .
Most vacationers still prefer a warm climate for their winter excursions .

Figure 2.2: Dependency parsing via word alignment.

resulting structures may contain cycles. The directed attachment score on WSJ10 achieved 39.3%, which is less than that of basic DMV (Klein and Manning, 2004).

A completely different method of obtaining dependency structures for languages without any linguistically annotated resources can be a projection of dependencies using a parallel corpus with a resource-rich language (typically English). McDonald et al. (2011) showed that using such projection produces better structures than what current unsupervised parsers are capable of. However, our task is different. We would like to produce structures that are not burdened by any linguistic conventions.

Chapter 3

Statistical Background

In this chapter, we review the basic techniques of Bayesian statistics to provide a background on the algorithms we employ for unsupervised parsing. We start with the well-known Bayes formula, discuss the differences between maximum likelihood estimation (MLE) and Bayesian inference and the corresponding expectation-maximization (EM) and Gibbs sampling procedures respectively. More detailed description can be found for example in the works of Goldwater (2006), Knight (2009), and Resnik and Hardisty (2010).

Throughout the thesis, vectors are denoted in bold (e.g. $\boldsymbol{\alpha}$) and scalars in normal font (e.g. α). The equation mark "=" is used also for estimating probabilities.

3.1 Maximum Likelihood Estimation

Let us start with the Bayes rule, which defines the probability of a hypothesis h (in our case, h is a linguistic grammar) given a data D:

$$p(h|D) = \frac{p(D|h)p(h)}{p(D)} \propto p(D|h)p(h). \tag{3.1}$$

The posterior probability $p(h|D)$ is proportional to the product of the *likelihood* $p(D|h)$ (the probability of the data under the hypothesis h) and the *prior* probability $p(h)$. The likelihood evaluates how well h explains the observed data D, and the prior evaluates how well h conforms to expectations about what a good hypothesis should be like, regardless of the observed data. A hypothesis with a high prior probability requires less evidence in its favor in order to be accepted.

In Maximum-Likelihood Estimation (MLE), we select the hypothesis h with the highest likelihood:

$$\hat{h} = \arg\max_h P(D|h). \tag{3.2}$$

This is equivalent to assuming that all hypotheses are equally probable, i.e. the *prior* probability $p(h)$ is uniform, and then choosing the single hypothesis with the highest *posterior* probability. Maximum likelihood estimation in an unsupervised context can be performed by expectation-maximization (EM) algorithm (Dempster et al., 1977). An example of an EM algorithm is the inside-outside algorithm (Baker, 1979), which is useful for learning context-free grammars and is used also in the DMV grammar induction system (Klein and Manning, 2004). EM is an iterative procedure with a very nice property: The likelihood is guaranteed to converge. However, the substantial disadvantage of EM is the fact that it converges only to a local maximum of the likelihood function, not to the global maximum. Complex models such as those often found in linguistic applications generally have many local maxima. This can lead to poor results that are highly dependent on parameter initialization (Carroll and Charniak, 1992).

3.2 Categorical and Dirichlet Distribution

Hypotheses h in language learning systems have often the form of a categorical[1] distribution. For example, we want to know the distribution of PoS tags for a given word in PoS tagging or a distribution of a dependent word given a head word in dependency parsing. In all such tasks, we have a given number of possible outcomes $1, \dots, m$ and parameters $\boldsymbol{\theta} = \theta_1, \dots, \theta_m$, which correspond to the probabilities of the individual outcomes and sum up to 1. Let us have the outcomes $x_1, \dots, x_n$ distributed according to a categorical distribution with the parameter $\boldsymbol{\theta}$:

$$x_1, \dots, x_n \sim Cat(\boldsymbol{\theta}) \qquad p(X_i = j|\boldsymbol{\theta}) = \theta_j. \tag{3.3}$$

In Bayesian statistics, we use a prior probability distribution different from discrete uniform. The natural prior distribution for a categorical distribution is a Dirichlet distribution. We can say that it is a distribution of distributions, because each sample from a Dirichlet distribution is a set of parameter values $\boldsymbol{\theta}$ for the categorical distribution.

$$\boldsymbol{\theta} \sim Dir(\alpha), \tag{3.4}$$

where $\alpha = \alpha_1, \dots, \alpha_m$ are called hyperparameters. The definition of the Dirichlet distribution is as follows:

$$p(\boldsymbol{\theta}|\boldsymbol{\alpha}) \propto \prod_{j=1}^{m} \theta_j^{\alpha_j - 1}. \tag{3.5}$$

Assume that we have a data $D = x_1, \dots, x_n$, where $x_i \in \{1, \dots, m\} \, \forall i$, and we want to compute its probability given the parameters $\boldsymbol{\theta}$, i.e. likelihood of the data D. Then:

$$p(D|\boldsymbol{\theta}) = \prod_{i=1}^{n} p(X_i = x_i|\boldsymbol{\theta}) = \prod_{i=1}^{n} \theta_{x_i} = \prod_{i=1}^{n} \prod_{j=1}^{m} \theta_j^{I(x_i=j)}, \tag{3.6}$$

[1] In the field of natural language processing, it is sometimes spoken of a "multinomial distribution" when a categorical distribution is actually meant.

where $I(x_i = j)$ is an indicator function which is equal to one if the element x_i equals to j. Otherwise, it is zero. If we switch the two products, we get:

$$p(D|\boldsymbol{\theta}) = \prod_{j=1}^{m} \theta_j^{\sum_{i=1}^{n} I(x_i=j)} = \prod_{j=1}^{m} \theta_j^{c_j}, \tag{3.7}$$

where $c_j = \sum_{i=1}^{n} I(x_i = j)$ refers to the number of occurrences of the element j in our data D.

The posterior distribution $p(\boldsymbol{\theta}|D)$ is then:

$$p(\boldsymbol{\theta}|D) \propto p(D|\boldsymbol{\theta})\, p(\boldsymbol{\theta}) = \prod_{j=1}^{m} \theta_j^{c_j} \prod_{j=1}^{m} \theta_j^{\alpha_j-1} = \prod_{j=1}^{m} \theta_j^{c_j+\alpha_j-1}. \tag{3.8}$$

Now we can see that the posterior distribution $p(\boldsymbol{\theta}|D)$ is proportional to another Dirichlet distribution, in this case with the hyperparameter vector equals to $\boldsymbol{c} + \boldsymbol{\alpha}$.

$$p(\boldsymbol{\theta}|D) \propto Dir(\boldsymbol{c} + \boldsymbol{\alpha}). \tag{3.9}$$

This is called *conjugacy*. The posterior distribution has the same analytical form as the prior distribution and thus the Dirichlet distribution is a *conjugate* prior to the categorical distribution.

Now assume that we have a new element x and we want to estimate its probability with respect to our data D. The predictive distribution is then estimated by an integral over all possible parameters $\boldsymbol{\theta}$:

$$p(x|D) = \int p(x, \boldsymbol{\theta}|D)\, d\boldsymbol{\theta} = \int p(x|D, \boldsymbol{\theta})\, p(\boldsymbol{\theta}|D)\, d\boldsymbol{\theta}. \tag{3.10}$$

Furthermore, we assume that the new element x is conditionally independent of the data D, i.e. $p(x|\boldsymbol{\theta}, D) = p(x|\boldsymbol{\theta})$, and is identically distributed, i.e. $x \sim Cat(\boldsymbol{\theta})$. Then:

$$p(x|D) = \int p(x|\boldsymbol{\theta})\, p(\boldsymbol{\theta}|D)\, d\boldsymbol{\theta} = \int \theta_x\, p(\boldsymbol{\theta}|D) = E(\theta_x|D), \tag{3.11}$$

which is the conditional expectation of θ_x given the data D. Such expected value of the Dirichlet distribution can be expressed as:

$$p(x|D) = E(\theta_x|D) = \frac{c_x + \alpha_x}{n + \alpha_0}, \qquad \alpha_0 = \sum_{j=1}^{m} \alpha_x. \tag{3.12}$$

We leave this without proof. A detailed derivation can be found e.g. in Resnik and Hardisty (2010). In case we ignore α_x and α_0 in Equation 3.12, we get the empirical probability c_x/n estimated from the data D without any previous prior knowledge. The prior is expressed here by the vector α and the hyperparameters α_j are sometimes called *pseudocounts*, i.e. virtual counts pre-set before seeing the data.

3.3 Bayesian Inference

In unsupervised natural language problems, we typically want to induce latent variables T on our data D. Specifically in the task of unsupervised dependency parsing, we have a raw (or PoS tagged) corpus D consisting of sentences $D_1, \dots, D_n$ and want to induce dependency trees $T = T_1, \dots, T_n$. We want to get such trees T that maximize the probability $p(T|D, \hat{\boldsymbol{\theta}})$, where $\hat{\boldsymbol{\theta}}$ is the *maximum aposteriory* (MAP) solution for $\boldsymbol{\theta}$:

$$\hat{\boldsymbol{\theta}} = \arg\max_{\theta} p(D|\boldsymbol{\theta})\, p(\boldsymbol{\theta}). \tag{3.13}$$

Since the best $\hat{\boldsymbol{\theta}}$ is not known, the distribution over latent variables T given the observed data D is obtained by integrating over all possible values of $\boldsymbol{\theta}$:

$$p(T|D) = \int p(T|D, \boldsymbol{\theta})\, p(\boldsymbol{\theta}|D)\, d\boldsymbol{\theta}. \tag{3.14}$$

An advantage of integrating over all possible $\boldsymbol{\theta}$ is that it allows to use linguistically appropriate priors. In linguistic models, we often deal with categorical distributions with parameters $\boldsymbol{\theta} = \theta_1, \dots, \theta_m$. Natural prior is then a distribution conjugated to the categorical one, i.e. the aforementioned Dirichlet distribution with parameters $\boldsymbol{\alpha} = \alpha_1, \dots, \alpha_m$.

As the prior, we often use a symmetric Dirichlet distribution, where all the hyperparameters are equal. We denote their value by the scalar α. With increasing α, parameters $\boldsymbol{\theta}$ approaches towards having the uniform distribution.[2]

Conversely, very low α causes that some parameter values tend to be very high and other very low. Linguistic structures have typically very sparse distributions and thus we often set $\boldsymbol{\alpha} < 1$.

Assume that the outcomes (let us imagine them as individual dependency edges in dependency trees) on the data D are generated one by one. A probability of a new outcome $x_i = y$ can be then computed using all the previously generated outcomes. We will call them the history of x_i and denote them by $\mathbf{x}_{-i}$.

$$\mathbf{x}_{-i} = \{x_1 \dots x_{i-1}\}. \tag{3.15}$$

Then, using Equations 3.14 and 3.12, we have:

$$p(x_i = y|\mathbf{x}_{-i}) = \int \theta_{x_i}\, p(\boldsymbol{\theta}|\mathbf{x}_{-i})\, d\boldsymbol{\theta} = \frac{c_{x_i}^{-i} + \alpha_{x_i}}{i - 1 + \alpha_0}, \tag{3.16}$$

where $c_{x_i}^{-i}$ stands for the number of times the value x_i occurred in the history $\mathbf{x}_{-i}$. This equation gives us a simple guide for the estimation of a new outcome based on other outcomes. We will use its modifications in our dependency models.

[2]Theoretically, the uniform distribution of $\boldsymbol{\theta}$ would be reached by setting $\alpha = \infty$.

3.3.1 Relationship with Chinese Restaurant Process

Equation 3.16 can also be easily explained in terms of the so-called “Chinese restaurant process”. Let us imagine a restaurant with an infinite number of round tables, each with an infinite capacity. At time 1, the first customer is seated at an unoccupied table with probability 1. At time $n+1$, a new customer comes and chooses randomly (uniformly) one place to sit from the following options: directly to the left of one of the n customers already sitting at any occupied table, or at a new, unoccupied table.

Our outcomes are the customers sitting in the restaurant and their values correspond to the tables. Dirichlet hyperparameters (pseudocounts) can be imagined as a set of customers whose seating we determine. Those customers are seated before the Chinese restaurant process starts and we will call them prior customers. If there is a very low number of prior-customers or none at all ($\alpha < 1$), just a few tables will be very popular during the process and others will be empty in most cases. If we sit one prior customer at each table, the occupation of the tables will be more uniform. There will still be some extremes, but not as big as in the first case. Finally, imagine that we will seat a thousand prior customers at each table. The resulting occupation will be then very close to uniform for a long time and a lot of customers will be necessary to change it.

3.4 Gibbs Sampling

For stochastic searching for a distribution over the latent variables T, we make use of the Gibbs sampling, a standard Markov Chain Monte Carlo procedure (Gilks et al., 1996) that produces samples from the posterior distribution

$$p(T|D) \propto p(D|T,\boldsymbol{\alpha})\, p(T|\boldsymbol{\alpha}). \tag{3.17}$$

In this section, we will describe the Gibbs sampling algorithm generally. Its application on dependency parsing, in which the latent variables T over the data D are the dependency trees, is described in Chapter 6.

Assume the data $D = D_1, \dots, D_n$ on which we want to predict the latent variables $T = T_1, \dots, T_n$. The general schema of the Gibbs sampling procedure looks as follows:

1. We initialize the variables T randomly.

2. We keep going through the data D in a random order and iteratively changing the values of respective latent variables T (one by one) according to their conditional distribution given the current values of the latent variable of all other elements. Exchangeability (see Section 6.1) allows us to treat the currently resampled element as if it was the last element in the data and thus the *history* is then composed of all elements but this one.

3. We repeat the previous step in many iterations.

4. We obtain the final probability distributions of our latent variables based on the samples generated during the sampling.

Since the choice in Step 2 is not uniformly random, the more likely latent variables are sampled more often than the less likely ones. However, the algorithm never converges since there is always a possibility to make a small change that leads to a less probable sample than the previous one. This feature helps the sampler to escape local from optima.

Chapter 4

Data and Evaluation

One of the motivations for unsupervised language learning is the applicability to any language. Moreover, since these methods do not require annotated texts, we can use any texts available, for example documents from the web. This chapter describes the data used in this work and discusses possible evaluation methods.

We use two types of resources in our experiments with unsupervised dependency parsing. Large raw (not linguistically annotated) monolingual corpora and smaller manually annotated monolingual treebanks to be able to automatically evaluate the results and compare them to unsupervised parsers developed elsewhere.

4.1 Raw Corpora from W2C

Raw texts are easily available in vast amounts since they can be automatically downloaded from the web. Hoverer, a more tricky issue necessary to be solved is the language recognition. The multilingual corpus W2C created by Majliš and Žabokrtský (2012) is publicly available and suitable for our purposes. It consists of two sources: Wikipedia articles[1] and other texts downloaded from the web by a web crawler. The advantage of the Wikipedia articles is that the information about language is provided with them. A language recognizer was trained on Wikipedia (Majliš, 2012) and used for the language recognition of the web texts.

Although the W2C corpus contains texts written in 106 different languages (and for 100 languages, there are more than 10 GB of texts available), we use only data of 29 of them, because only for those 29 languages we have treebanks available for evaluation purposes (see Section 4.2). The statistics of the selected data are summarized in Table 4.1.

4.2 Treebanks

Treebanks are necessary to evaluate automatically to what degree the induced trees match linguistic conventions. Different treebanks exist for more than 30 languages,

[1]http://www.wikipedia.org

language	code	Wikipedia			Web		
		words (kw)	unique (kw)	avg. length	words (kw)	unique (kw)	avg. length
Arabic	ar	2,846	139	4.50	1,575	120	4.84
Basque	eu	9,716	440	6.61	64,498	1,370	6.38
Bengali	bn	2,611	199	5.66	35,434	659	5.09
Bulgarian	bg	14,346	526	5.26	60,088	1,373	4.99
Catalan	ca	22,538	463	4.58	98,690	748	4.50
Chinese	zh	6,242	4,289	7.99	867	354	7.54
Czech	cs	15,619	754	5.56	139,301	2,477	5.34
Danish	da	12,545	523	5.23	76,718	1,321	4.97
Dutch	nl	22,078	602	5.30	126,957	1,824	5.03
English	en	68,478	636	4.92	752,168	8,229	4.71
Estonian	et	8,469	779	6.65	80,153	2,193	6.04
Finnish	fi	13,657	1,211	7.56	93,191	3,837	7.24
German	de	44,699	1,340	6.15	97,305	2,212	5.71
Greek	el	16,834	571	5.50	99,467	1,622	5.32
Hindi	hi	15,231	459	4.20	42,134	535	3.82
Hungarian	hu	19,423	1,147	6.13	90,970	3,089	6.03
Italian	it	32,464	542	5.20	130,870	1,546	5.10
Japanese	ja	7,806	4,841	10.54	68,838	35,556	9.43
Latin	la	2,531	267	6.21	32,435	1,231	5.81
Persian	fa	16,142	290	3.67	104,540	913	3.69
Portuguese	pt	25,502	448	4.99	82,158	1,121	4.88
Romanian	ro	17,824	523	5.23	154,886	1,370	4.82
Russian	ru	25,445	1,042	6.18	36,178	1,607	6.04
Slovenian	sl	10,517	570	5.40	87,328	1,463	5.07
Spanish	es	44,509	635	4.96	223,776	2,295	4.87
Swedish	sv	15,397	678	5.49	93,963	1,661	4.95
Tamil	ta	5,806	696	8.13	48,367	2,113	7.51
Telugu	te	6,420	606	6.46	23,254	1,205	6.42
Turkish	tr	12,618	665	6.31	106,312	2,422	6.24

Table 4.1: Statistics of W2C Wikipedia and Web corpora for selected languages. They show the total number of words (kw = thousands of words), the number of unique words and the average word length. This table was extracted from the W2C “stat” files.

the problem is, however, that almost each of them uses their own annotation style and data format. For our purposes, we use our compilation of treebanks called HamleDT (HArmonized Multi-Language Dependency Treebanks) described in Zeman et al. (2012). We solved the problem of annotation style inconsistencies by developing a set of converters able to transform such heterogeneous dependency and phrase-structure treebanks into one common format. The list of treebanks currently present in the HamleDT collection follows.

- Arabic (ar): Prague Arabic Dependency Treebank 1.0 / CoNLL 2007; Smrž et al. (2008)[2]
- Basque (eu): Basque Dependency Treebank (larger version than CoNLL 2007 generously provided by IXA Group); Aduriz et al. (2003)
- Bengali (bn): *see Hindi*
- Bulgarian (bg): BulTreeBank; Simov and Osenova (2005)[3]
- Catalan (ca) and Spanish (es): AnCora; Taulé et al. (2008)
- Chinese (zh): Sinica treebank / CoNLL 2007; Chen and Hsieh (2004)[4]
- Czech (cs): Prague Dependency Treebank 2.0 / CoNLL 2009; Hajič et al. (2006)[5]
- Danish (da): Danish Dependency Treebank / CoNLL 2006; Kromann et al. (2004), now a part of the Copenhagen Dependency Treebank[6]
- Dutch (nl): Alpino Treebank / CoNLL 2006; van der Beek et al. (2002)[7]
- English (en): Penn TreeBank 2 / CoNLL 2009; Surdeanu et al. (2008)[8]
- Estonian (et): Eesti keele puudepank / Arborest; Bick et al. (2004)[9]
- Finnish (fi): Turku Dependency Treebank; Haverinen et al. (2010)[10]
- German (de): Tiger Treebank / CoNLL 2009; Brants et al. (2002)[11]

[2]http://padt-online.blogspot.com/2007/01/conll-shared-task-2007.html
[3]http://www.bultreebank.org/indexBTB.html
[4]http://godel.iis.sinica.edu.tw/CKIP/engversion/treebank.htm
[5]http://ufal.mff.cuni.cz/pdt2.0/
[6]http://code.google.com/p/copenhagen-dependency-treebank/
[7]http://odur.let.rug.nl/~vannoord/trees/
[8]http://www.cis.upenn.edu/~treebank/
[9]http://www.cs.ut.ee/~kaili/Korpus/puud/
[10]http://bionlp.utu.fi/fintreebank.html
[11]http://www.ims.uni-stuttgart.de/projekte/TIGER/TIGERCorpus/

- Greek (modern) (el): Greek Dependency Treebank; Prokopidis et al. (2005)
- Greek (ancient) (grc) and Latin (la): Ancient Greek Dependency Treebank; Bamman and Crane (2011)[12]
- Hindi (hi), Bengali (bn) and Telugu (te): Hyderabad Dependency Treebank / ICON 2010; Husain et al. (2010)
- Hungarian (hu): Szeged Treebank; Csendes et al. (2005)[13]
- Italian (it): Italian Syntactic-Semantic Treebank / CoNLL 2007; Montemagni et al. (2003)[14]
- Japanese (ja): Verbmobil; Kawata and Bartels (2000)[15]
- Latin (la): Latin Dependency Treebank; Bamman and Crane (2011)[16]
- Persian (fa): Persian Dependency Treebank; Rasooli et al. (2011)[17]
- Portuguese (pt): Floresta sint(c)tica; Afonso et al. (2002)[18]
- Romanian (ro): Romanian Dependency Treebank; Călăcean (2008)[19]
- Russian (ru): Syntagrus; Boguslavsky et al. (2000)
- Slovene (sl): Slovene Dependency Treebank / CoNLL 2006; Džeroski et al. (2006)[20]
- Spanish (es): *see Catalan*
- Swedish (sv): Talbanken05; Nilsson et al. (2005)[21]
- Tamil (ta): TamilTB; Ramasamy and Žabokrtský (2012)[22]
- Telugu (te): *see Hindi*
- Turkish (tr): METU-Sabanci Turkish Treebank; Atalay et al. (2003)[23]

Other characteristics of HamleDT treebanks, including the numbers of non-projective dependencies, are shown in Table 4.2.

[12]http://nlp.perseus.tufts.edu/syntax/treebank/greek.html
[13]http://www.inf.u-szeged.hu/projectdirs/hlt/index_en.html
[14]http://medialab.di.unipi.it/isst/
[15]http://www.sfs.uni-tuebingen.de/en/tuebajs.shtml
[16]http://nlp.perseus.tufts.edu/syntax/treebank/latin.html
[17]http://dadegan.ir/en/persiandependencytreebank
[18]http://www.linguateca.pt/floresta/info_floresta_English.html
[19]http://www.phobos.ro/roric/texts/xml/
[20]http://nl.ijs.si/sdt/
[21]http://www.msi.vxu.se/users/nivre/research/Talbanken05.html
[22]http://ufal.mff.cuni.cz/~ramasamy/tamiltb/0.1/
[23]http://www.ii.metu.edu.tr/content/treebank

Language	code	Pri. tree type	Used data source	Sents.	Tokens	Train / test [% snt]	Avg. sent. length	Non-proj. [%]
Arabic	ar	dep	CoNLL'07	3,043	116,793	96 / 4	38.38	0.37
Basque	eu	dep	primary	11,226	151,604	90 / 10	13.50	1.27
Bengali	bn	dep	ICON'10	1,129	7,252	87 / 13	6.42	1.08
Bulgarian	bg	phr	CoNLL'06	13,221	196,151	97 / 3	14.84	0.38
Catalan	ca	phr	CoNLL'09	14,924	443,317	88 / 12	29.70	0.00
Chinese	zh	dep	CoNLL'07	57,647	342,336	99 / 1	N/A	N/A
Czech	cs	dep	CoNLL'07	25,650	437,020	99 / 1	17.04	1.91
Danish	da	dep	CoNLL'06	5,512	100,238	94 / 6	18.19	0.99
Dutch	nl	phr	CoNLL'06	13,735	200,654	97 / 3	14.61	5.41
English	en	phr	CoNLL'09	40,613	991,535	97 / 3	24.41	0.39
Estonian	et	phr	primary	1,315	9,491	90 / 10	7.22	0.07
Finnish	fi	dep	primary	4,307	58,576	90 / 10	13.60	0.51
German	de	phr	CoNLL'09	38,020	680,710	95 / 5	17.90	2.33
Greek	el	dep	CoNLL'07	2,902	70,223	93 / 7	24.20	1.17
Greek	grc	dep	primary	21,160	308,882	98 / 2	14.60	19.58
Hindi	hi	dep	ICON'10	3,515	77,068	85 / 15	21.93	1.12
Hungarian	hu	phr	CoNLL'07	6,424	139,143	94 / 6	21.66	2.90
Italian	it	dep	CoNLL'07	3,359	76,295	93 / 7	22.71	0.46
Japanese	ja	dep	CoNLL'06	17,753	157,172	96 / 4	8.85	1.10
Latin	la	dep	primary	3,473	53,143	91 / 9	15.30	7.61
Persian	fa	dep	primary	12,455	189,572	97 / 3	15.22	1.77
Portuguese	pt	phr	CoNLL'06	9,359	212,545	97 / 3	22.71	1.31
Romanian	ro	dep	primary	4,042	36,150	93 / 7	8.94	0.00
Russian	ru	dep	primary	34,895	497,465	99 / 1	14.26	0.83
Slovene	sl	dep	CoNLL'06	1,936	35,140	79 / 21	18.15	1.92
Spanish	es	phr	CoNLL'09	15,984	477,810	90 / 10	29.89	0.00
Swedish	sv	phr	CoNLL'06	11,431	197,123	97 / 3	17.24	0.98
Tamil	ta	dep	primary	600	9,581	80 / 20	15.97	0.16
Telugu	te	dep	ICON'10	1,450	5,722	90 / 10	3.95	0.23
Turkish	tr	dep	CoNLL'07	5,935	69,695	95 / 5	11.74	5.33

Table 4.2: Statistics of individual treebanks included in HamleDT, adopted from Zeman et al. (2012). "Non-proj." stands for the percentage of non-projective dependencies; "Train / test" expresses the ratio between training and testing sets. In case a treebank was not originally divided into a training and a testing part, we determined the testing part ourselves by separating roughly 5000 tokens.

4.3 Evaluation Metrics

As in other unsupervised tasks (e.g. in unsupervised PoS induction), there is little consensus on evaluation measures. The performance of unsupervised methods is often measured by comparing the induced outputs with gold-standard manual annotations (Gelling et al., 2012). However, this approach causes a general problem: manual annotation is inevitably guided by a number of conventions. It is thus questionable, whether the unsupervised PoS tagging should adhere to the traditional PoS categories, or what conventions for local tree shapes representing e.g. complex verb forms, should be used to measure the performance of the unsupervised dependency parsing.

Different linguistic conventions used to capture particular linguistic phenomena in the form of dependencies across various treebanks were described by Zeman et al. (2012). Coordination structures are probably the most heterogeneous, since there

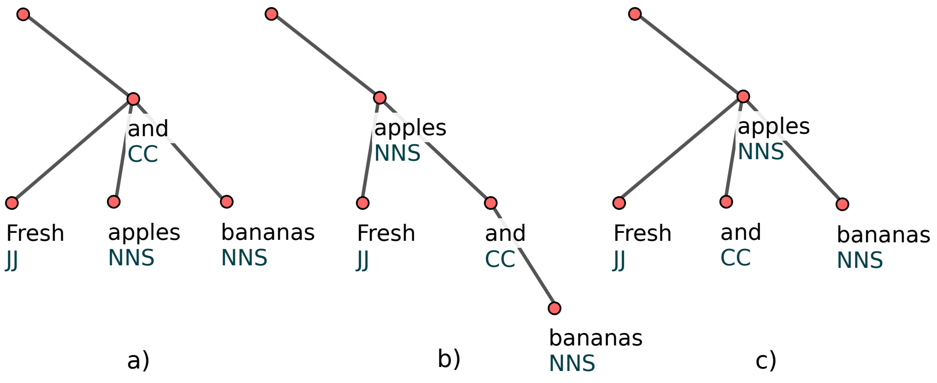


Figure 4.1: Three different annotations of coordination structures.

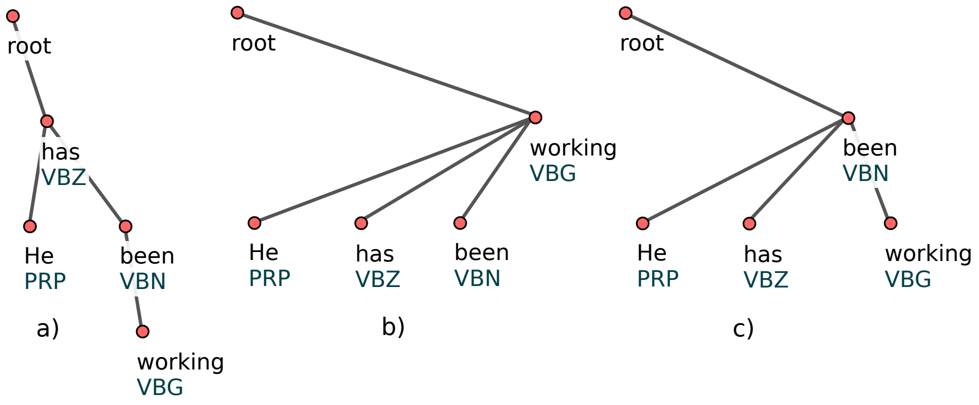


Figure 4.2: Three different annotations of the complex verb form “has been working”.

exist many linguistically motivated possibilities for them (see Figure 4.1). Relations between auxiliary verbs and finite verbs seem to be even more problematic for evaluation as they are very frequent. Some possibilities are listed in Figure 4.2. The problem here is to decide which verb should be the head, which one should be its child and on which of them the individual arguments should depend. Figure 4.3 shows two possibilities of attaching prepositions and subordinating conjunctions.

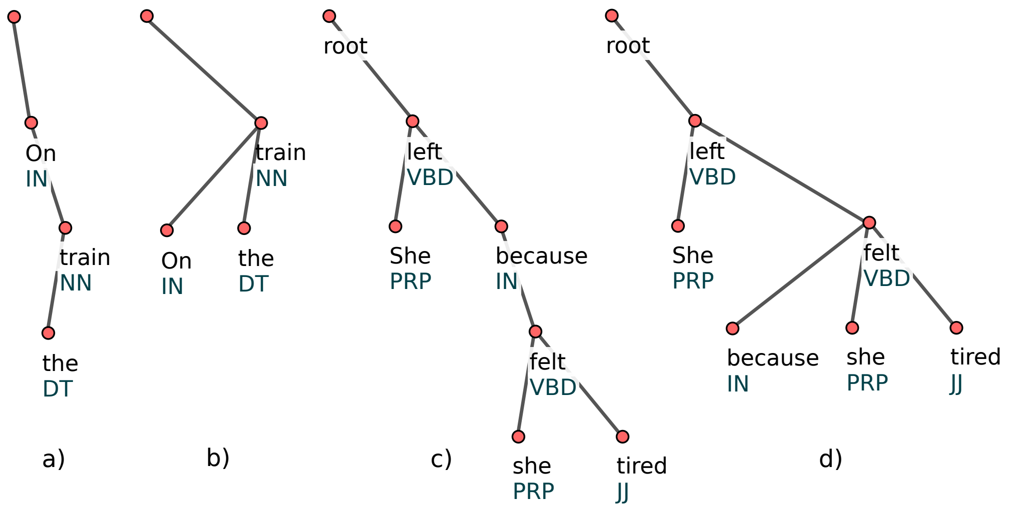


Figure 4.3: Two different annotation styles for prepositions and subordinating conjunctions. In a) and c), the function words are heads of the structures, whereas in b) and d) they are leaves.

Schwartz et al. (2011) discuss three different annotation schemes used to convert English phrase structures into dependencies and conclude that the differences between them are substantial. For instance, when evaluating two of the three annotation schemes on the Penn Treebank section 23, they discovered that 14.4% of edges were attached in a different way. In the following text, we will describe the main evaluation metrics[24] that have been used to measure the quality of unsupervised dependency parsers.

4.3.1 Directed Attachment Score

The *directed attachment score* (DAS) is the standard metric for dependency parsers. We simply calculate the percentage of words attached to a "correct" parent.

$$DAS(G, P) = \frac{1}{n} \sum_{i=1}^{n} I(g_i = p_i) \; \times 100 \quad [\%], \tag{4.1}$$

[24]All the proposed attachment scores ignore the dependency labels in the treebanks, since we predict only the structure and not individual types of the dependency relations.

where G and P are vectors of gold and predicted parents respectively. Although this metric does not allow the even slightest local structural differences, which might be caused just by more or less arbitrary linguistic or technical conventions, it is the most commonly used metric, probably because of its simplicity and the tradition in the field.

4.3.2 Undirected Attachment Score

It is obvious that using directed attachment scores leads to a strong bias towards such conventions and might not be a good indicator of unsupervised parsing improvements. The second metric, which is more tolerant and has the aim of reducing such bias, is called *undirected attachment score* (UAS). The direction of edges are disregarded here.

$$UAS(G,P) = \frac{1}{n}\sum_{i=1}^{n} I(g_i = p_i \vee g_{p_i} = i) \ \times 100 \quad [\%]. \tag{4.2}$$

Figure 4.4 shows that two different linguistic conventions for the attachment of modal verbs are more similar in case the undirected attachment score is used instead of the directed one. On the other hand, completely unwanted attachments, such as a noun depending on an adjective, are also judged as correct by this metric.

4.3.3 Neutral Edge Direction

The *neutral edge direction*[25] (NED) metric was proposed by Schwartz et al. (2011). It is even more tolerant in assessing parsing errors than the undirected attachment score. It treats not only node's gold-standard parent and child as the correct answer, but also its gold grandparent.

$$NED(G,P) = \frac{1}{n}\sum_{i=1}^{n} I(g_i = p_i \vee g_{p_i} = i \vee g_i = p_{p_i}) \ \times 100 \quad [\%]. \tag{4.3}$$

By definition, the NED metric completely ignores the edge flip. Figure 4.4 documents that the flipped edge between the words *must* and *have* is correct according to the NED. However, the NED greatly increases the number of false positives, i.e. incorrect attachments treated as correct, and yet it does not cover all the differences in linguistic conventions.

4.3.4 Removing Punctuation

Most works in unsupervised dependency parsing report their results ignoring the attachment of punctuation completely. This is justified because the attachment of

[25]`http://www.cs.huji.ac.il/~roys02/software/ned.html`

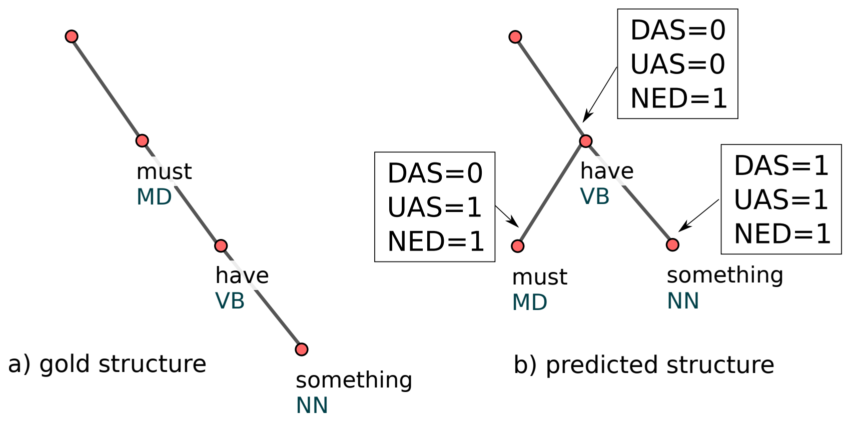


Figure 4.4: Evaluation using directed attachment score (DAS), undirected attachment score (UAS), and neutral edge direction (NED) on a predicted structure (b) that has the edge *must-have* flipped in comparison to the gold standard (a). We can see that DAS is the most strict one, whereas NED marks the predicted structure as completely correct.

punctuation is very arbitrary. For example, in the Russian treebank (Boguslavsky et al., 2000), punctuation is not treated as dependency tree nodes at all. Or, imagine that all the full stops of predicted Czech trees are "correctly" attached to the main verb. However, in the Czech treebank (Hajič et al., 2006), the full stops are attached to the technical roots. The DAS score would be then immediately lower by about 5% since the average sentence length is 20 words and a full stop appears in almost all of them.

Integrating punctuation removal into the evaluation as a preprocessing step before applying one of the aforementioned metrics thus appears as a possible solution. Most punctuation nodes are leaves in the trees and removing them is simple. If there are some words depending on a punctuation mark, they can be re-attached to its parent node.

We experiment with excluding punctuation from evaluation and excluding punctuation from learning in Section 7.3.4.

Chapter 5

Dependency Tree Models

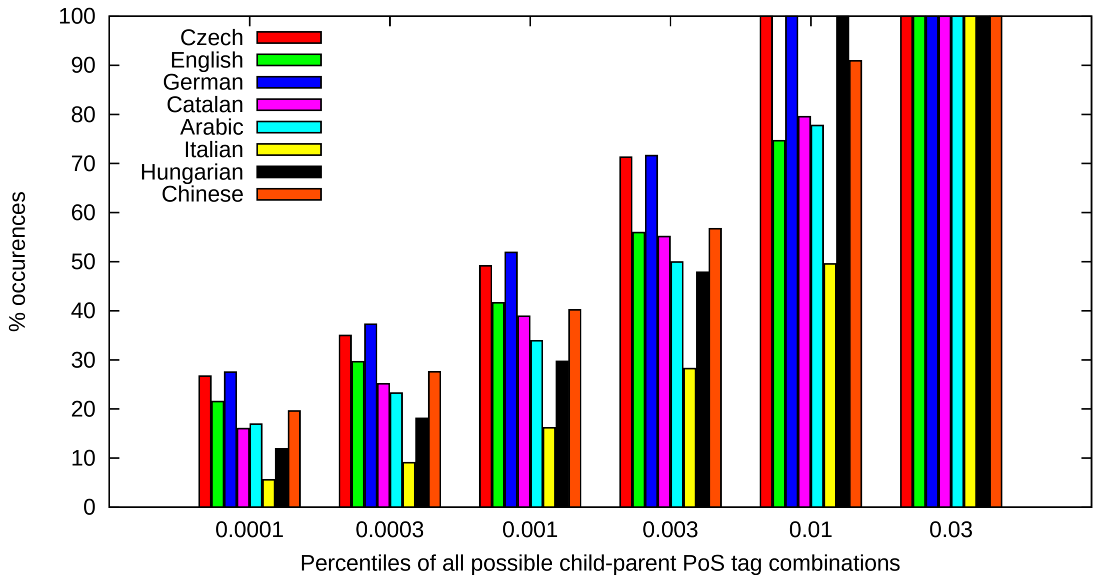
Developing a model that meets our requirements for the outcomes is the most important thing in unsupervised induction. We do not have any annotated training data and therefore we must rely on basic intuitions. In general, our task is to induce a linguistic structure, but there are various constraints that may be imposed upon this structure. We have already made one essential assumption, which is the dependency tree shape (see Section 1.5). Although we know several language phenomena for which the tree structure is not well-suited (e.g. coordination structures), we still decide to produce dependency trees. Such constraint is very helpful in unsupervised induction since it prohibits many unreasonable structures. Projectivity (see Section 5.6) can be used as another, stricter constraint.

In this chapter we will examine the models that reflect our basic intuitions about dependency trees, such as repeatability of dependency relations, short distances between dependents and governors, reducibility of dependents, fertility of words etc. Various combinations of these models are then described in Section 5.5.

5.1 Edge Models

Probably the most obvious feature which can be useful for the induction of dependencies is the fact that the distribution of dependency relations among the pairs of words in the corpus is not uniform. Particular words often relate to a very small subset of all possible words. For example, English adjectives depend very often on nouns and almost never on adverbs. The word “York” has a very common dependent, which is “New”.

Figure 5.1 shows that the majority of probability mass is concentrated on a relatively low number of dependency relations (out of all possible pairs of words or pairs of part-of-speech tags, respectively). We can see that in case of PoS tags, 10% of the most frequent dependency edge types cover almost the whole treebank. In case of word-forms, it is 0.03%. Czech, German, and Hungarian treebanks contain less then 0.01% of all possible pairs of word-forms. It must be noted that word form percentages are rather illustrative since they depend strongly on the corpus size.



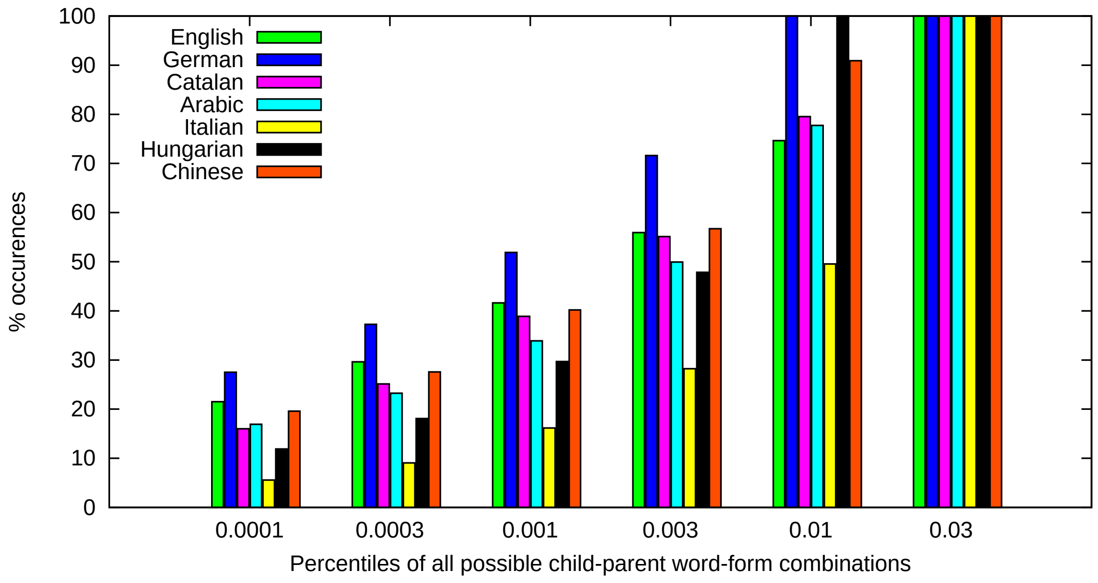


Figure 5.1: The percentage of edges in Czech, English, German, Catalan, Arabic, Italian, Hungarian, and Chinese treebanks that are covered by a particular percentile of all possible edge types ($|W|^2$) sorted according to their frequencies in the treebanks. The top figure shows statistics measured on part-of-speech-tags, the bottom figure shows statistics of word-forms. For example, if we take 1% of PoS tag pairs that constitute a dependency edge most frequently in the Chinese treebank, we cover about 90% of edges.

5.1.1 Naive Edge Model

Let us define a *dependency edge* as a pair $[w_d, w_g]$, where w_d is the dependent and w_g the governing word. We assume that the probability mass over all possible dependency edges in the corpus is concentrated into a relatively low number of types and the majority of types are very unlikely. In addition, we assume that the dependency relations in the corpus are mutually independent.

Let us first define a very naive model as a product of probabilities over all dependency edges in the treebank.

$$P_{join} = \prod_{d=1}^{n} P([w_d, w_{\pi(d)}]) = \prod_{d=1}^{n} \frac{c([w_d, w_{\pi(d)}])}{n}, \tag{5.1}$$

where n is the number of words in the corpus and $w_{\pi(d)}$ is the parent of the word w_d. The best dependency trees would be then found by maximizing the P_{join}.

However, this model is mathematically incorrect since we assume independence among the individual edges in a tree, which is not true. Note that every word plays the role of a dependent in a dependency edge just once, but could be in the role of a governor several times or not at all. Therefore, the maximization of the product of joined probabilities would cause that more frequent words tend to be the heads and less frequent words tend to be the leaves in the most probable dependency trees. Such structures are not desired.

5.1.2 Conditioning by Head

A better mathematical model which avoids this undesired behavior uses conditional probability instead of the joint probability in Equation 5.1. The probability of the dependency edge is now conditioned on the governing word.

$$P_{cond} = \prod_{d=1}^{n} P(w_d|w_{\pi(d)}) = \prod_{d=1}^{n} \frac{c([w_d, w_{\pi(d)}])}{c(w_{\pi(d)})}. \tag{5.2}$$

Conditioning on the head word ensures independence from $c(w_{\pi(d)})$, which represents the number of times $w_{\pi(d)}$ was a governing word. Note that $w_{\pi(d)}$ can be also the technical root, which is never in the role of a dependent. Number of such roots in the treebank equals the number of sentences. This basic model (or its variants) was used in the majority of works published in the field of unsupervised dependency parsing (Yuret, 1998; Klein and Manning, 2004).

Maximizing the conditional edge model in Equation 5.2 is the same as maximizing the sum over pointwise mutual information between dependent words. Pointwise mutual information in the context of dependency trees is computed as follows.

$$\mathrm{pmi}(d, g) = \log \frac{p(w_d, w_g)}{p(w_d)p(w_g)} \tag{5.3}$$

We define the pointwise mutual information of the whole tree as the sum of pointwise mutual information of individual edges:

$$\mathrm{pmi}(tree) = \sum_{d=1}^{n} \mathrm{pmi}(w_d, w_{\pi(d)}) = \log \prod_{d=1}^{n} \frac{p(w_d, w_{\pi(d)})}{p(w_d)p(w_{\pi(d)})}. \tag{5.4}$$

We can omit the probabilities of the dependent words for maximization since they are the same for all possible trees corresponding to a given sentence.

$$\arg\max_{tree} \mathrm{pmi}(tree) = \arg\max_{tree} \prod_{d=1}^{n} \frac{p(w_d, w_{\pi(d)})}{p(w_{\pi(d)})} = \arg\max_{tree} P_{cond}. \tag{5.5}$$

5.1.3 A switch to Bayesian Statistics

We want to estimate the conditional probabilities $p(w_d|w_g)$ from Equation 5.2. We assume that this probability has a categorical distribution with parameters $\boldsymbol{\theta}$. Since the Dirichlet distribution provides natural priors for the categorical distribution (see Section 3.2 for a detailed description), we add symmetric Dirichlet priors parameterized by $\boldsymbol{\alpha}$:

$$w_d|w_g \sim Cat(\boldsymbol{\theta}) \tag{5.6}$$
$$\theta|\alpha \sim Dir(\boldsymbol{\alpha}) \tag{5.7}$$

Given this model, Equation 3.16 yields the formula for the probability of a dependency edge $[w_d, w_g]$ given the parent word w_g and the history (all the preceding dependency edges[1]):

$$P_e([w_d, w_g]|w_g) = \frac{c^{-d}(\text{"}w_d, w_g\text{"}) + \alpha}{c^{-d}(\text{"}w_g\text{"}) + \alpha|W|}, \tag{5.8}$$

where w_d and w_g are words at positions d and g in the treebank and $c^{-d}(\text{"}w_d, w_g\text{"})$ indicates the number of edges $[w_d, w_g]$ in the history. Unlike in Equation 3.16, here we have the number of parents $c^{-d}(\text{"}w_g\text{"})$ in the denominator since we are conditioning on the parent word. Note, that the count $c^{-d}(\text{"}w_g\text{"})$ refers to the number of edges whose parent is w_g, not the number of words w_g. $|W|$ is the number of parameters, which is the number of distinct words is in this case. The term $\alpha|W|$ stands for α_0 from Equation 3.16.

5.1.4 Various Edge Models

The edge model in Equation 5.8 can be applied to word forms, to part-of-speech tags, or to a combination of both. It also proves useful to condition the probability

[1]We define the position of a dependency edge as the position of the dependent word in the corpus.

of a dependency edge on the word-order direction of the dependency, which is used in Dependency Models with Valence as well (Klein and Manning, 2004; Headden et al., 2009) (see Section 2.2). For example in English, adjectives are attached to their governing nouns from the left side, nouns appear to the right of their governing prepositions, etc.

In the following four Equations, we present possible variants of the edge model. For simplification, we will denote the dependency edges only as $[d, g]$, where d and g are the positions of the dependent and the governing word, respectively. We define the dependency direction $dir(d, g)$ to determine, whether the governing word is to the left ($d > g$) or to the right ($d < g$) from the dependent. While w_d and w_g are the word forms, t_d and t_g denote the respective PoS tags. All four models use symmetric Dirichlet priors.

Since the corpora on which we run the induction algorithm are not very large and the distributions of the individual word forms are thus very sparse, we first introduce the edge model based on part-of-speech tags (or alternatively, automatically induced word classes):

$$P_{et}(d, g) = \frac{c^{-d}(\text{``}t_d, t_g\text{''}) + \alpha_{et}}{c^{-d}(\text{``}t_g\text{''}) + \alpha_{et} \cdot |T|}, \tag{5.9}$$

where $|T|$ is the number of distinct tags (or unsupervised word classes) used in the corpus. A variant of this model, where the dependents are in addition conditioned on the edge direction, look as follows:

$$P_{etd}(d, g) = \frac{c^{-d}(\text{``}t_d, t_g, dir(d, t)\text{''}) + \alpha_{etd}}{c^{-d}(\text{``}t_g, dir(d, t)\text{''}) + \alpha_{etd} \cdot |T|}, \tag{5.10}$$

In lexicalized edge models, we suppose that we have already generated the part-of-speech tag of the dependent. The word form of the dependent is thus conditioned on the part-of-speech tags of the dependent and the governor and also on the word form of the governor:

$$P_{ew}(d, g) = \frac{c^{-d}(\text{''}t_d, t_g, w_d, w_g\text{''}) + \alpha_{ew}}{c^{-d}(\text{''}t_d, t_g, w_g\text{''}) + \alpha_{ew} \cdot |W|}, \tag{5.11}$$

And finally, the lexicalized edge model conditioned also by the edge direction:

$$P_{ewd}(d, g) = \frac{c^{-d}(\text{''}t_d, t_g, w_d, w_g, dir(d, g)\text{''}) + \alpha_{ewd}}{c^{-d}(\text{''}t_d, t_g, w_g, dir(d, g)\text{''}) + \alpha_{ewd} \cdot |W|}, \tag{5.12}$$

We mostly use the part-of-speech tag edge model conditioned on the edge direction (Equation 5.10) in our experiments (see Chapter 7), sometimes in a combination with one of the lexicalized models.

5.2 Fertility Models

By the *fertility* of a node in a dependency tree, we mean the number of its children (dependents). In Dependency Model with Valence (Klein and Manning, 2004), the fertility is modelled by the `STOP` sign (see Section 2.2). Every time we want to generate a new dependent, we first use the stop model $P_{STOP}(STOP|t_g, dir(d,g), adj)$ that determines whether the new dependent can be generated or not. The stop model is able to induce different fertilities for different heads because it is conditioned on the head's part-of-speech tag.

In this work, the fertility f_i of a node at the i-th position in our treebank models directly the number of its children conditioned on the PoS tag t_i. Similarly as with the edge models, we assume a categorical distribution of fertilities for a given PoS tag with a Dirichlet prior β:

$$f_i|t_i \sim Cat(\boldsymbol{\phi}), \tag{5.13}$$

$$\phi|\beta \sim Dir(\boldsymbol{\beta}). \tag{5.14}$$

Unlike in edge models, where the Dirichlet prior distribution was symmetric, here we assume that higher fertilities are less probable than lower ones. We introduce a prior probability P_0, which is estimated as follows:

$$P_0(f_i) = \frac{1}{2^{f_i+1}}. \tag{5.15}$$

The prior probability of a fertility f_i decreases exponentially with the fertility itself. A node is a leaf with a probability of $\frac{1}{2}$; has just one child with a probability $\frac{1}{4}$, etc. The base 2 was chosen so that all possible fertilities sum up to one.

Examples of fertility distributions extracted from English and German treebanks in Figures 5.2 and 5.3 show that for a majority of PoS tags, the zero fertility is dominant. In contrast to this, the fertility of verbs is almost never zero. See the English PoS tags with the prefix "VB" and the German PoS tags with the prefix "VV".

Following Equation 3.16, we derive a formula for the basic fertility model:

$$P_f(f_i|t_i) = \frac{c^{-i}(\text{“}t_i, f_i\text{”}) + \beta_0 P_0(f_i)}{c^{-i}(\text{“}t_i\text{”}) + \beta_0}, \tag{5.16}$$

where f_i is the number of children of the i-th word, β_0 is the Dirichlet hyperparameter, and $P_0(f_i)$ is the prior probability. In this case, the parameters of the Dirichlet prior distribution are $\beta = \beta_0 \, P_0(f_i)$.

The following slight modification of the fertility model distinguishes the numbers of left and right children. Instead of one number, it predicts a pair $[f^L, f^R]$. For example, fertility $[1,3]$ means that the node has one left and three right dependents, fertility $[0,0]$ indicates that the node is a leaf.

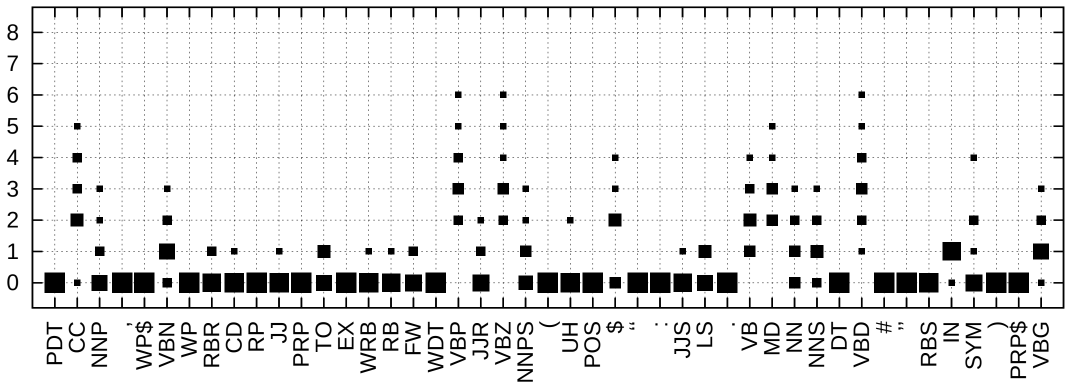


Figure 5.2: Fertility distribution conditioned by individual part-of-speech tags in the English treebank. Areas of squares are proportional to the counts of occurrences.

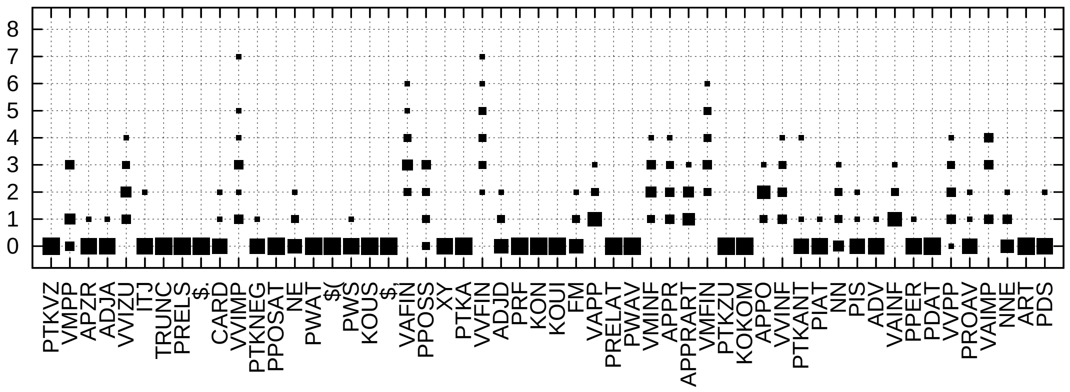


Figure 5.3: Fertility distribution conditioned by individual part-of-speech tags in the German treebank. Areas of squares are proportional to the counts of occurrences.

$$P_{fd}(f_i^L, f_i^R|t_i) = \frac{c^{-i}(\text{“}t_i, f_i^L, f_i^R\text{”}) + \beta_0 P_0(f_i^L + f_i^R)}{c^{-i}(\text{“}t_i\text{”}) + \beta_0}, \tag{5.17}$$

The prior probability P_0 is defined here in the same way as before, using the total number of children for each node.

Besides the basic fertility models, we also introduce a more complex model which uses the frequency of a given word form to generate the number of children. We assume that the most frequent words are mostly function words (e.g. determiners, prepositions, auxiliary verbs, conjunctions). Such words tend to have a stable number of children, for example (i) some function words are exclusively leaves, (ii) prepositions have just one child, and (iii) the attachment of auxiliary verbs depends on the annotation style, but the number of their children is also not very variable. The higher the frequency of a word form, the higher the probability mass concentrated on one specific number of children and the lower the Dirichlet hyperparameter β_0 in Equation 5.17 needed. The extended fertility is described by the following equation:

$$P_{fdx}(f_i^L, f_i^R | t_i) = \frac{c^{-i}(``t_i, f_i^L, f_i^R") + \frac{\beta_0}{F(w_i)} P_0(f_i^L + f_i^R)}{c^{-i}(``t_i") + \frac{\beta_0}{F(w_i)}}. \tag{5.18}$$

The relative word frequency $F(w_i)$ is computed by dividing the number of occurrences of the word form w_i in the corpus by the corpus size.

5.3 Distance Model

We define the *distance* between two words in a sentence as the difference between their word-order positions. Distances between two dependent words (edge lengths) are rather short in a typical case. Figure 5.4 shows the distributions of edge lengths in four different treebanks. We can see that the probability of a dependency edge between two words decreases rapidly with its length.

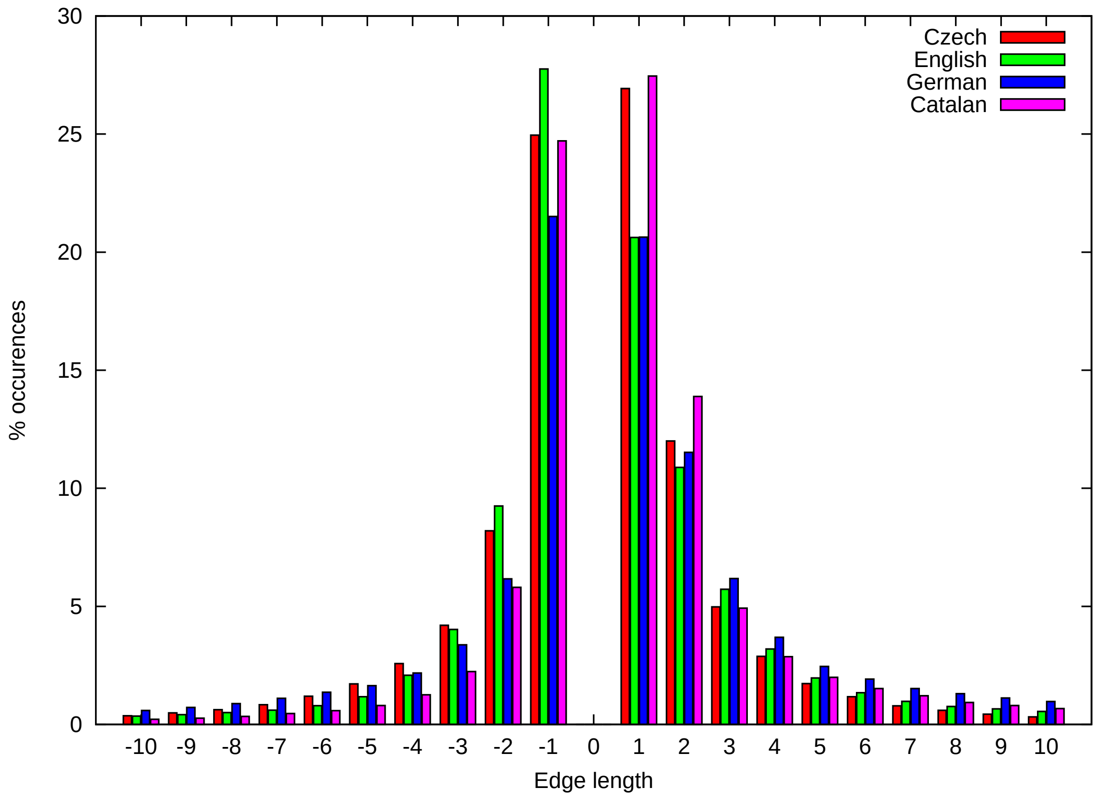


Figure 5.4: Distribution of edge lengths for various languages, as measured on Czech, English, German and Catalan treebanks included in the CoNLL 2006 and 2007 shared tasks.

In the *distance model*, we approximate the probability of the edge as the inverse value of the distance between the dependent word and its parent:[2]

$$P_d(d, g) = \frac{1}{\epsilon_d} \left(\frac{1}{|d - g|} \right)^{\gamma}, \tag{5.19}$$

where ϵ_d is the normalization constant and the hyperparameter γ determines the impact of this model.

5.4 Reducibility Model

The notion of *reducibility*, i.e. the possibility of deleting a word from a sentence without violating its syntactic correctness, belongs to traditionally known manifestations of syntactic dependency. As mentioned e.g. by Kübler et al. (2009), one of the traditional linguistic criteria for recognizing dependency relations (including their head-dependent orientation) is that a head H of a construction C determines the syntactic category of C and can often replace C. Or, in words of "Dependency Analysis by Reduction" of Lopatková et al. (2005), stepwise deletion of dependent elements within a sentence should preserve its syntactic correctness. A similar idea of dependency analysis by splitting the sentence into all possible acceptable fragments is used by Gerdes and Kahane (2011).

All the above works had obviously to respond to the notorious fact that there are many language phenomena precluding the ideal word-by-word) sentence reducibility (e.g. the case of prepositional groups, or English finite clause subjects). However, we disregard their solutions tentatively and borrow only the very core of the reducibility idea: if a word can be removed from a sentence without damaging it, then it is likely to depend on another word which is still present.

As is usual with dichotomies in natural languages, it seems more adequate to use a continuous scale instead of a reducible-irreducible opposition. That is why we introduce a simple reducibility measure based on n-gram corpus statistics.

5.4.1 Obtaining Reducible Words

We call a word (or a sequence of words) in a sentence *reducible* if the sentence remains grammatically correct after the removal of this word (or sequence). But here we face the problem that we cannot simply recognize whether a given sentence is grammatical or not. This might be possible in case we have a grammar; however, the grammar is the thing what we are trying to infer. We would need some negative feedback, similar to what children have when they learn their mother tongue (see Section 1.1). However, the only thing we have available are collections of many positive examples – the large monolingual corpora described in Section 4.1.

[2]We decided to use a reciprocal function here. The use of an exponential function would be also possible. However, we did not observe much differences on our experiments.

We determine the grammaticality of a newly created (i.e. reduced) sentence by searching for it in the corpus. If we find it, we assume that the removed word was reducible in the original sentence. This is certainly an improper solution since we suppose that all grammatically correct sentences occur in the corpus[3], but we are still able to recognize at least some words that are reducible. The experiments show that even the relatively row number of reducible PoS n-grams is sufficient for estimating PoS-ngram reducibility scores and improve the parsing quality for most of the languages (see Section 7.3.6).

The necessity of searching for whole sentences in the corpus and not only for smaller context,[4] which would lead to lower sparsity, is rationalized by the following example:

Their children went to school.
I took their children to school.

The verb *'went'* would be reducible in the context *'their children went to school'*, because the sequence *'their children to school'* occurs in the second sentence. One could find such examples frequently even for larger contexts. For instance, verbs in free word order languages can be placed almost at any position in the sentence. The following two Czech sentences are both correct:

Pavel s námi na výlet do Orlických hor v tomto hrozném počasí nepůjde.
[lit: Paul with us on a-trip to the-Eagle Mountains in this terrible weather will-not-come.]

Pavel nepůjde s námi na výlet do Orlických hor v tomto hrozném počasí.
[lit: Paul will-not-come with us on a-trip to the-Eagle Mountains in this terrible weather.]

They differ only in the position of the verb *nepůjde [will not come]*. The verb would be considered as reducible in this case, if we take shorter segments than whole sentences into account. This is not correct since the sentence does not make sense without the verb. In order to prevent such errors, we decided to work exclusively with the full sentence context instead of shorter contexts.

Another possibility of reaching a lower sparsity would be searching for sequences of part-of-speech tags instead of sequences of word forms. However, this also does not bring desired results. For instance, the following two sentence patterns

DT NNS VBD IN DT NN .
DT NNS VBD DT NN .

are quite frequent in English and we can deduce from them that the preposition IN is reducible. But this is of course a wrong deduction since the preposition cannot be

[3]Assume that natural languages have a possibly infinite number of grammatically correct sentences due to their recursivity. The fraction of sentences occurring in a corpus of any size is therefore close to zero.

[4]For example, we could consider using just the left and the right neighbor of the given word or using a trigram language model.

removed from the prepositional phrase. Using part-of-speech tags instead of word forms is thus not suitable for the reducibility score computation.

5.4.2 Computing Reducibility Scores

Our algorithm searches the corpus not only for reducible words but also for sequences of words. We compute the reducibility score for each part-of-speech tag (and sequences of part-of-speech tags) based on the number of occurrences of reducible words (sequences of words) with the particular part-of-speech tags. This requires a morphological disambiguation of the corpus. A sequence of part-of-speech tags will be denoted as a *PoS n-gram* in the following text.

Assume a PoS n-gram $g = [t_1, \ldots, t_n]$. We go through the corpus and search for all its occurrences. For each occurrence, we remove the respective words from the current sentence and check in the corpus whether the rest of the sentence occurs at least once elsewhere in the corpus.[5] If so, then this occurrence of the PoS n-gram is reducible, otherwise it is not. We denote the count of the reducible occurrences of the PoS n-gram g by $r(g)$. The number of all its occurrences is $c(g)$.

We compute the relative reducibility $R(g)$ of a PoS n-gram g as

$$R(g) = \frac{1}{N}\frac{r(g) + \sigma_1}{c(g) + \sigma_2}, \tag{5.20}$$

where the normalization constant N, which expresses the relative reducibility over all the PoS n-grams (denoted by G), causes that the mean of the scores is 1.

$$N = \frac{\sum_{g \in G}(r(g) + \sigma_1)}{\sum_{g \in G}(c(g) + \sigma_2)} \tag{5.21}$$

The smoothing constants σ_1 and σ_2, which prevent reducibility scores from being equal to zero, are set to

$$\sigma_1 = \frac{\sum_{g \in G} r(g)}{\sum_{g \in G} c(g)}, \qquad \sigma_2 = 1 \tag{5.22}$$

This setting causes that even if a given PoS n-gram is not reducible anywhere in the corpus, its reducibility score is $1/(c(g) + 1)$.

Tables 5.1, 5.2, and 5.3 show the reducibility scores of the most frequent PoS n-grams for three selected languages: English, German, and Czech. If we consider unigrams only, we can see that the scores for verbs are often among the lowest. Verbs are followed by prepositions and nouns, and the scores for adjectives and adverbs are very high for all three examined languages. This is desired since the

[5] We do not take sentences with less then 10 words into account because they could be nominal (without any verb) and might influence the reducibility scores of verbs.

unigrams	reduc.	bigrams	reduc.	trigrams	reduc.
VB	0.04	VBN IN	0.00	IN DT JJ	0.00
TO	0.07	IN DT	0.02	JJ NN IN	0.00
IN	0.11	NN IN	0.04	NN IN NNP	0.00
VBD	0.12	NNS IN	0.05	VBN IN DT	0.00
CC	0.13	JJ NNS	0.07	JJ NN .	0.00
VBZ	0.16	NN .	0.08	DT JJ NN	0.04
NN	0.22	DT NNP	0.09	DT NNP NNP	0.05
VBN	0.24	DT NN	0.09	NNS IN DT	0.14
.	0.32	NN ,	0.11	NNP NNP .	0.15
NNS	0.38	DT JJ	0.13	NN IN DT	0.23
DT	0.43	JJ NN	0.14	NNP NNP ,	0.46
NNP	0.78	NNP .	0.15	IN DT NNP	0.55
JJ	0.84	NN NN	0.22	DT NN IN	0.59
RB	2.07	IN NN	0.67	NNP NNP NNP	0.64
,	3.77	NNP NNP	0.76	IN DT NN	0.80
CD	55.6	IN NNP	1.81	IN NNP NNP	4.27

Table 5.1: Reducibility scores of the most frequent English PoS n-grams. (*V** are verbs, *N** are nouns, *DET* are determiners, *IN* are prepositions, *JJ* are adjectives, *RB* are adverbs, *CD* are numerals, and *CC* are coordinating conjunctions.)

unigrams	reduc.	bigrams	reduc.	trigrams	reduc.
VVPP	0.00	NN APPR	0.00	NN APPR NN	0.01
APPR	0.27	APPR ART	0.00	ADJA NN APPR	0.01
VFIN	0.28	ART ADJA	0.00	APPR ART ADJA	0.01
APPRART	0.32	NN VVPP	0.00	NN KON NN	0.01
VAFIN	0.37	NN $(	0.01	ADJA NN $.	0.01
KON	0.37	NN NN	0.01	NN ART NN	0.32
NN	0.43	NN ART	0.21	ART NN ART	0.49
ART	0.49	ADJA NN	0.28	NN ART ADJA	0.90
$(	0.57	NN $,	0.67	ADJA NN ART	0.95
$.	1.01	NN VAFIN	0.85	NN APPR ART	0.95
NE	1.14	NN VVFIN	0.89	NN VVPP $.	1.01
CARD	1.38	NN $.	0.95	ART NN APPR	1.35
ADJA	2.38	ART NN	1.07	ART ADJA NN	1.58
$,	2.94	NN KON	2.41	APPR ART NN	2.60
ADJD	3.54	APPR NN	2.65	APPR ADJA NN	2.65
ADV	7.69	APPRART NN	3.06	ART NN VVFIN	9.51

Table 5.2: Reducibility scores of the most frequent German PoS n-grams. (*V** are verbs, *N** are nouns, *ART* are articles, *APPR** are prepositions, *ADJ** are adjectives, *ADV* are adverbs, *CARD* are numerals, and *KON* are conjunctions.)

unigrams	reduc.	bigrams	reduc.	trigrams	reduc.
P4	0.00	RR AA	0.00	RR NN Z:	0.00
RV	0.00	Z: J,	0.00	NN RR AA	0.00
Vp	0.06	Vp NN	0.00	NN AA NN	0.16
Vf	0.06	VB NN	0.12	AA NN RR	0.23
P7	0.16	NN Vp	0.13	NN RR NN	0.46
J,	0.24	NN VB	0.18	NN J^ NN	0.46
RR	0.28	NN RR	0.22	AA NN NN	0.47
VB	0.33	NN AA	0.23	NN Z: Z:	0.48
NN	0.72	NN J^	0.62	NN Z: NN	0.52
J^	1.72	AA NN	0.62	NN NN NN	0.70
C=	1.85	NN NN	0.70	AA AA NN	0.72
PD	2.06	NN Z:	0.97	AA NN Z:	0.86
AA	2.22	Z: NN	1.72	NN NN Z:	1.38
Dg	3.21	Z: Z:	1.97	RR NN NN	2.26
Z:	4.01	J^ NN	2.05	RR AA NN	2.65
Db	4.62	RR NN	2.20	Z: NN Z:	8.32

Table 5.3: Reducibility scores of the most frequent Czech PoS n-grams. (*V** are verbs, *N** are nouns, *P** are pronouns, *R** are prepositions, *A** are adjectives, *D** are adverbs, *C** are numerals, *J** are conjunctions, and *Z** is punctuation.)

reducible unigrams are more likely to become leaves in the induced dependency trees. Considering bigrams, the couples [*determiner – noun*], [*adjective – noun*], and [*preposition – noun*] obtained reasonably high scores. However, there are also n-grams such as the German trigram [*determiner – noun – preposition*] (ART-NN-APPR) whose reducibility score is undesirably high.[6]

Figure 5.5 depicts the correlation between the unigram reducibility of the individual Czech PoS tags and the number of times these tags correspond to leaves in gold-standard dependency trees. We can see that the correlation is positive, which suggest that the reducibility feature can be useful.

5.4.3 Reducibility Model

The higher the reducibility score of a particular PoS n-ngram, the more likely the PoS n-gram constitutes a rooted subtree in the dependency tree. Let us define $desc(i)$ as the PoS n-gram (a sequence of part-of-speech tags $[t_l \cdots t_r]$) that corresponds to all the descendants of the word w_i including w_i, i.e. the whole rooted subtree of w_i.

[6]The high reducibility score of ART-NN-APPR was probably caused by German particles, which have the same PoS tag as prepositions.

We assume that the probability of such a subtree is proportional to the reducibility $R(desc(i))$.

$$P_r(i) = \frac{1}{\epsilon_r} R(desc(i))^{\delta}, \tag{5.23}$$

where ϵ_d is the normalization constant and the hyperparameter δ determines the impact of this model.

Note that the reducibility model is different from the previous three models, since it utilizes external large monolingual corpus to obtain the reducibility scores. The inference itself is done on a much smaller corpus.

5.5 Combining the Models

The previously described models are combined into a single one by multiplying them over all nodes in the treebank. The main configuration used in our experiments is a combination of models defined in Equations 5.10, 5.18, 5.19, and 5.23. The formula for computing probability of the whole treebank looks as follows:

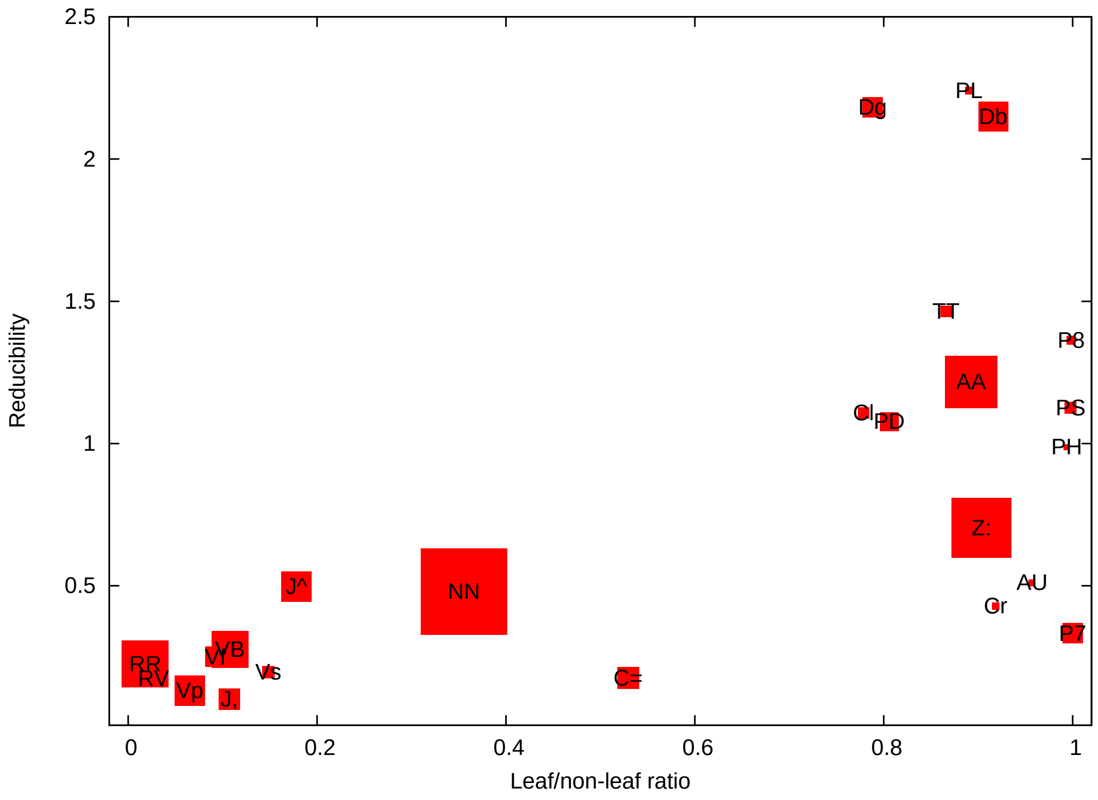


Figure 5.5: Correlation between unigram reducibility of individual Czech PoS tags and frequency of them being leaves in gold-standard dependency trees. The size of the squares corresponds to the frequencies of the individual PoS tags.

$$\begin{aligned}
P_{treebank} &= \prod_{i=1}^{n} P_{etd}(i, \pi(i))\, P_{fdx}(f_i, \pi(i))\, P_d(i, \pi(i))\, P_r(i) = \\
&= \prod_{i=1}^{n} \frac{c^{-i}(\text{“}t_i, t_{\pi(i)}, dir(i, \pi(i))\text{”}) + \alpha_{etd}}{c^{-i}(\text{“}t_{\pi(i)}, dir(i, \pi(i))\text{”}) + \alpha_{etd} \cdot |T|} \\
&\quad \frac{c^{-i}(\text{“}t_i, f_i^L, f_i^R\text{”}) + \frac{\beta_0}{F(w_i)} P_0(f_i^L + f_i^R)}{c^{-i}(\text{“}t_i\text{”}) + \frac{\beta_0}{F(w_i)}} \\
&\quad \frac{1}{\epsilon_d} \frac{1}{|i - \pi(i)|^{\gamma}} \\
&\quad \frac{1}{\epsilon_r} R(desc(i))^{\delta}.
\end{aligned} \tag{5.24}$$

The dependency function $\pi(i)$ returns the position of the parent of the word at the position i. In our experiments (Section 7.3), we will add, remove or substitute the individual submodels to inspect their positive and negative impacts for different configurations.

5.6 Projectivity

Projectivity is an important property of natural languages, even though there are many exceptions which violate this constraint. The notion of projectivity was established by Harper and Hays (1959), who mentioned that projections of dependency trees into sentences have a tendency to fill continuous intervals.

Generally, there are not many non-projective edges in manually annotated treebanks. Havelka (2007) studied non-projective constructions in treebanks included in CoNLL 2006 shared task and reported about 2.1% of non-projective edges for Czech, 2.4% for German and similar or lower percentages of non-projective edges for other languages. It is important to note that the number of non-projective edges depends not only on the chosen language but also on the selected annotation guidelines.

Edge projectivity can be also modeled, for example similarly to the distance between the governing and the dependent word by introducing a penalty for non-projective edges. However, such a feature is not convenient for our inference algorithm (see Section 6.2).

CHAPTER 6

Inference of Dependency Trees

In this chapter, the algorithm for dependency trees inference is described in detail. We employ the Gibbs sampling algorithm Gilks et al. (1996), a Monte Carlo method which allows us to solve the integral from Equation 3.14. In Section 6.1, we show the basic algorithm for dependency edge sampling without the "treeness" constraint, using only the simple edge model. The algorithm for projective dependency tree sampling is derived in Section 6.2. The decoding step (Section 6.3) is necessary to obtain the final dependency trees.

6.1 Basic Algorithm

We provide the basic algorithm first since we want to describe properly the sampling technique in a simple setting. For simplicity, we use just the edge model (Equation 5.9) and the task here is not to create a dependency tree but only to find a parent for each word. This means that the structures we are sampling may contain cycles and can be discontinuous.

The treebank probability, which we want to maximize, is then:

$$P_{treebank} = \prod_{i=1}^{n} P_{edge}(t_i|t_{\pi(i)}) = \prod_{i=1}^{n} \frac{c^{-i}(\text{“}t_i, t_{\pi(i)}\text{”}) + \alpha}{c^{-i}(\text{“}t_{\pi(i)}\text{”}) + \alpha|T|} \tag{6.1}$$

We follow the generic algorithm from Section 3.4:

1. The dependency edges are initialized randomly. Since our task is not constrained by the condition of "treeness", we simply assign a random parent word to each word in each sentence.

2. We keep going through all the words in the corpus in a random order in many iterations and changing their attachments using the *small change operator*.

In our case, the *small change operator* is a re-attachment of a chosen node. An example of such a small change is depicted in Figure 6.1. Assume that we have selected the word "lunch" in the dependency tree and want to make a small change

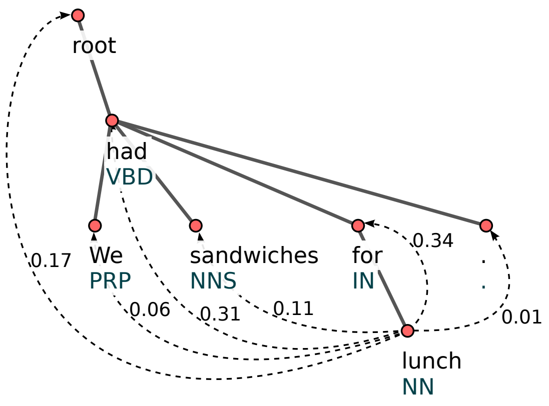


Figure 6.1: Performing a small change operator in basing sampling algorithm.

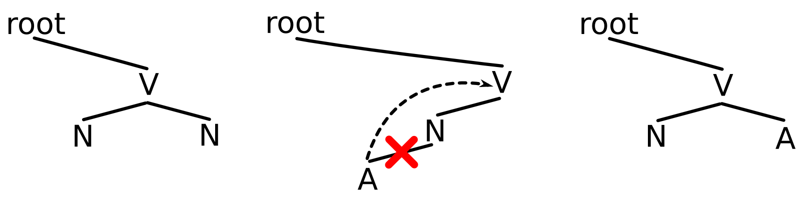


Figure 6.2: Exchangeability feature showed on a very small treebank containing only three sentences. Letters “N”, “V”, and “A” stand for nouns, verbs and adjectives respectively; “root” symbols represent the technical roots.

on it, in our case to change its parent. Since the sentence has six words, we have six possibilities of attaching it; the five other words and the technical root. Note that there is always the possibility of not changing anything, i.e. to choose the current parent “for” as the new parent. We compute the new overall probability of the treebank after each small change. These probabilities are then normalized (see the example numbers in Figure 6.1) and according to the obtained distribution, we randomly choose one candidate. We keep doing such small changes through the whole treebank. We go through all the sentences and make a small change on every word in a random order. One iteration is one pass through the whole corpus. A pseudo-code of this simple sampling is in Figure 6.3.

In a sense, we are sampling random treebanks one after another. Since the small changes are not uniformly random, the samples are slowly pushed towards the area with more probable treebanks. However, there is always a chance of moving to another area with different kinds of trees. This algorithm never converges by its definition, but if we sample long enough, we are most likely to get better and better samples, however, it is not guaranteed.

The overall treebank probability computation (on Line 8 in Figure 6.3), needed to compute the sampling distributions before each small change, poses a time complexity problem. Computing it according to the Equation 6.1 would be absolutely

```
for $i = 0; i < iterations; i{+}{+}$ do
    foreach $sentence \in corpus$ do
        foreach $node \in$ randomPermutation($sentence \rightarrow$getNodes()) do
            # estimate probability of node's parents
            foreach $parent \in sentence \rightarrow$getNodes() $\cup$ "root") do
                if $parent$ != $node$ then
                    $node \rightarrow$setParent($parent$);
                    $prob[parent]$ = estimateTreebankProbability();
                end
            end
            # choose parent w.r.t. the distribution
            $n\_prob$ = normalize($prob$);
            $parent$ = sampleFromDistribution($n\_prob$);
            $node \rightarrow$setParent($parent$);
        end
    end
end
```

Figure 6.3: Basic sampling algorithm.

impossible (it requires one pass through all words in the corpus). Instead, we use the fact that the probability of a sample after the small change differs only a little from the probability of the previous sample. Assume the following very simple treebank depicted in Figure 6.2 and the highlighted small change, where the adjective "A" in the second sentence changes its parent from the noun "N" to the verb "V". If we followed Equation 6.1 and computed the probability before the small change, we would end up with the following fractions:

$$P_{old} = \overset{\text{N}\leftarrow\text{V}}{\frac{0+\alpha}{0+4\alpha}} \overset{\text{root}\rightarrow\text{V}}{\frac{0+\alpha}{0+4\alpha}} \overset{\text{V}\rightarrow\text{N}}{\frac{1+\alpha}{1+4\alpha}} \overset{\text{A}\leftarrow\text{N}}{\frac{0+\alpha}{0+4\alpha}} \overset{\text{N}\leftarrow\text{V}}{\frac{2+\alpha}{2+4\alpha}} \overset{\text{root}\rightarrow\text{V}}{\frac{1+\alpha}{1+4\alpha}} \overset{\text{N}\leftarrow\text{V}}{\frac{3+\alpha}{3+4\alpha}} \overset{\text{root}\rightarrow\text{V}}{\frac{2+\alpha}{2+4\alpha}} \overset{\text{V}\leftarrow\text{A}}{\frac{0+\alpha}{4+4\alpha}} \qquad (6.2)$$

The corresponding edges are shown above the fractions. The first edge N←V has no history and therefore there are zero counts both in the numerator and the denominator. The third edge V→N has a "1" in the numerator, since there was one such edge in its history (we disregard the edge direction in this setting). The "1" in the denominator means that there was one edge with the same parent PoS tag in its history, etc. After the small change is done, the probability of the new treebank is as follows:

$$P_{new} = \overset{\text{N}\leftarrow\text{V}}{\frac{0+\alpha}{0+4\alpha}} \overset{\text{root}\rightarrow\text{V}}{\frac{0+\alpha}{0+4\alpha}} \overset{\text{V}\rightarrow\text{N}}{\frac{1+\alpha}{1+4\alpha}} \overset{\text{A}\leftarrow\text{V}}{\frac{0+\alpha}{2+4\alpha}} \overset{\text{N}\leftarrow\text{V}}{\frac{2+\alpha}{3+4\alpha}} \overset{\text{root}\rightarrow\text{V}}{\frac{1+\alpha}{1+4\alpha}} \overset{\text{N}\leftarrow\text{V}}{\frac{3+\alpha}{4+4\alpha}} \overset{\text{root}\rightarrow\text{V}}{\frac{2+\alpha}{2+4\alpha}} \overset{\text{V}\leftarrow\text{A}}{\frac{1+\alpha}{5+4\alpha}} \qquad (6.3)$$

Note that although only one edge has been changed, many more fractions must be updated because the histories of all the edges following the changed edge have changed. We changed the edge A←N to A←V and thus the last edge V→A now has one occurrence in its history and therefore the number 0 in the numerator must be changed to 1. The verb V has becomes a head one more time than it has been before and thus the number 4 in the last denominator must be changed to 5.

If we inspect the numerators and denominators separately, we can observe a regularity: the numbers in the numerators for a particular edge keep growing. For example, the numerators for the edge N←V contain the numbers 0, 1, and 2. The same holds for the numbers in the denominators any specific parent tag. Denominators for edges with the parent tag "V" contain the numbers 0, 1, 2, 3, and 4.

The difference between P_{old} and P_{new} lies only in the numerators and denominators associated with the changed edge:

$$P_{new} = P_{old} \frac{1+\alpha}{5+4\alpha} / \frac{0+\alpha}{0+4\alpha} \tag{6.4}$$

Therefore, when we compute the probability of the treebank by multiplying the fractions for the individual edges, the ordering of the edges is not important even though the edges have different histories. When computing the new probability, we can assume that the edge we are changing is the last edge in the treebank and all other edges are in its history. Then the histories of all other edges remain the same and we can change just the last fraction. This feature is called *exchangeability.*

In general, if we have a treebank with a probability P_{old} and change an edge X←Y to X←Z, the probability P_{new} of the changed treebank is:

$$P_{new} = P_{old} \frac{c^{others}(\text{``}Y\text{''}) + \alpha|T|}{c^{others}(\text{``}X,Y\text{''}) + \alpha} \frac{c^{others}(\text{``}X,Z\text{''}) + \alpha}{c^{others}(\text{``}Z\text{''}) + \alpha|T|}, \tag{6.5}$$

where c^{others} are the counts of the respective edges in the whole treebank, excluding the edge which is currently being changed.

Using this update, we can quickly estimate the new treebank probability after each small change in a constant time. The exchangeability property applies to all the models presented in Chapter 5 as well.

However, this *basic algorithm* has a crucial disadvantage: The sampled structures may contain cycles (and may be disconnected).[1] There are several ways of ensuring the acyclicity:

1. We can use the basic algorithm and simply allow only such samples that do not cause any cycles. However, this will not work very well since such an algorithm will deal differently with nodes in different positions in a tree. Although

[1]In the case of our task, which is to assign a parent word to each word in a sentence, the condition of acyclicity is equal to the condition of connectivity. The sampled structure contains a cycle if and only if it is disconnected.

the leaves could be attached to all other possible words in the sentence, the word attached to the technical root (e.g. the verb "had" in Figure 6.1) in a dependency tree has no other possibility of attachment in case it has no siblings since all other nodes are its descendants.

2. Our paper (Mareček and Žabokrtský, 2011) introduces a sampling algorithm where a tree with a cycle is fixed by re-attaching one of the node in the cycle to another node which is outside the cycle. The choice of the node in the cycle and the choice of its new parent is done by sampling as well. However, this algorithm cannot be applied to the reducibility model (see Section 5.4) since that model assumes that the current structure is always a tree. The cycles would cause that subtrees of certain nodes would not be defined and the reducibility could not be computed.

3. We define a more complex small change operator, where more nodes change their parents together at one step. This operator will be described in the following section.

6.2 Sampling Projective Trees

We introduce a Gibbs sampling algorithm that preserves the tree structure when re-attaching nodes in a tree. Moreover, the trees sampled using this algorithm are strictly projective, which is very useful from several points of view:

- Tree projectivity is a very valuable constraint for unsupervised parsing. Even though some of the language phenomena are certainly of a non-projective nature, they occur very rarely and it can be beneficial to disallow them completely. (See the percentages of non-projective edges in the individual treebanks in Table 4.2.)

- Our reducibility model (Section 5.4) requires that the subtree of a given node correspond to a continuous sequence of words in the sentence. It would be probably useful for discontinuous sequences as well, but it is more suitable for the continuous ones since the reducibility score is obtained from continuous sequences as well.

6.2.1 Initialization

Before the sampling starts, we initialize the projective trees randomly. We use the two following initializers for this step:

- *FlatInit* – For each sentence, we randomly choose one word as the head and attach all other words to it.

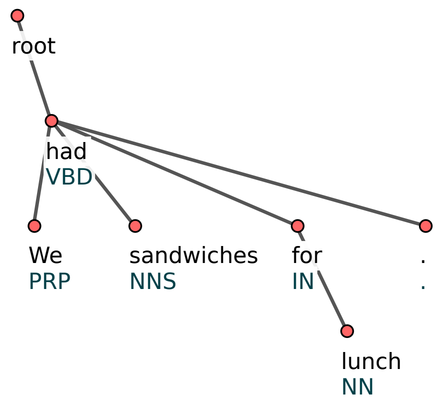
```
foreach $sentence \in corpus$ do
    foreach $node \in$ randomPermutation($sentence\rightarrow$getNodes()) do
        $left\_parent$ = $node\rightarrow$getPrevNode();
        while notAttached($left\_parent$) & $left\_parent$ do
            $left\_parent$ = $left\_parent\rightarrow$getPrevNode();
        end
        $right\_parent$ = $node\rightarrow$getNextNode();
        while notAttached($right\_parent$) & $right\_parent$ do
            $right\_parent$ = $node\rightarrow$getNextNode();
        end
        $parent$ = random($right\_parent$, $left\_parent$);
        if $parent$ then
            $node\rightarrow$setParent($parent$);
        end
        else
            if $right\_parent$ then
                $node\rightarrow$setParent($right\_parent$);
            end
            else if $left\_parent$ then
                $node\rightarrow$setParent($left\_parent$);
            end
            else
                $node\rightarrow$setParent($sentence\rightarrow$getRoot());
            end
        end
    end
end
```

Figure 6.4: Projective initializer.

- *RealInit* – We pick one word after another in a random order and attach it to the nearest left (or right) neighbor that has not been attached yet. The left-right choice is made randomly. If it is not possible to attach a word to one neighbor, we attach it to the other one. The last unattached word then becomes the head of the sentence. See the pseudo-code in Figure 6.4.

The *FlatInit* method generates only flat trees, whereas the more complex *RealInit* is able to generate all possible projective trees.[2] However, experiments showed that

[2]Note that the *RealInit* initializer does not generate all projective dependency trees with equal probability. It favors trees with shorter edges. We are not aware of any algorithm that would uniformly generate projective trees and be fast enough. Searching for all the possibilities is exponential and uncomputable for longer sentences.



((We) had (sandwiches) (for (lunch)) (.))

Figure 6.5: Edge and bracketing notation of a projective dependency tree.

((We) had sandwiches (for (lunch)) (.))

↓

(((We) had) sandwiches (for (lunch)) (.))

((We) (had) sandwiches (for (lunch)) (.))

((We) had (sandwiches) (for (lunch)) (.))

((We) had (sandwiches (for (lunch))) (.))

((We) had (sandwiches (for (lunch)) (.)))

Figure 6.6: An example of a small change in a projective tree. The bracket (*sandwiches*) is removed and there are five possibilities for replacing it.

the sampler converges to similar results for both the initializations. Therefore, we conclude that the choice of the initialization mechanism is not so important and choose the *FlatInit* initializer because of its simplicity.

6.2.2 Small Change Operator

We use a bracketing notation to illustrate the small change operator. Each projective dependency tree consisting of n words can be expressed by n pairs of brackets. Each bracket pair belongs to one node and delimits its descendants from the rest of the sentence. Furthermore, each bracketed segment contains just one word that is not embedded deeper; this node is the head of this segment head. An example of this notation is shown in Figure 6.5.

A small change that abides the projective tree constraint is very simple to accomplish in this notation. We remove one pair of brackets and add another so that the projective tree properties are not violated. Figure 6.6 illustrates all possible pairs of brackets in an example situation.

From the perspective of dependency structures, the small change can be described as follows:

1. Pick a random non-root word w (the word *sandwiches* in our example) and find its parent p (the word *had*).
2. Find all other children of w and p (the words *We*, *for*, and *.*) and denote this set of the nodes found by C.
3. Choose the new head out of w and p. Mark the new head as g and the second candidate as d. Attach d to g.
4. Select a neighborhood D, a continuous subset of C, adjacent to the word d. Attach all words from D to d. (D may be empty).
5. Attach the remaining words from C that were not in D to the new head g.

Figure 6.7 shows all possible dependency trees that can be created after the example small change shown in Figure 6.6. Obviously, one such small change typically leads to a re-attachments of more than one node.

6.3 Decoding

The Gibbs sampling algorithm never converges. The dependency trees which are sampled in the last few iterations always fluctuate around the best solution that fits our model by far. In a majority of applications, we will probably need to have the final trees fixed, not only to have a probabilistic distribution on them. Fixed trees are also more suitable for evaluation and for inspecting the resulting structures. The way of obtaining the final dependency trees from the sampling is called decoding. We introduce several possible decoding methods.

The easiest way would be to make no decoding at all and simply take the treebank as is after the last sampling iteration. If we do this, some edges will end up wrong due to an accidental choice of a low probability option in the previous iterations. Figure 6.8 shows the decreasing number of changes during a sampling experiment. We can see that no more than 15% of dependencies are changed after the twentieth iteration. If we assume that a half of these changes are for the worse and the second half for the better, we can expect that the trees after the last iteration may differ in 7% of edges compared to the trees that can be obtained by more sophisticated methods.

One of such methods of settling the final dependency trees is called "simulated annealing" or "decreasing the temperature" (Kirkpatrick et al., 1983). The term "temperature" is borrowed from metallurgy. Decreasing the temperature will lower

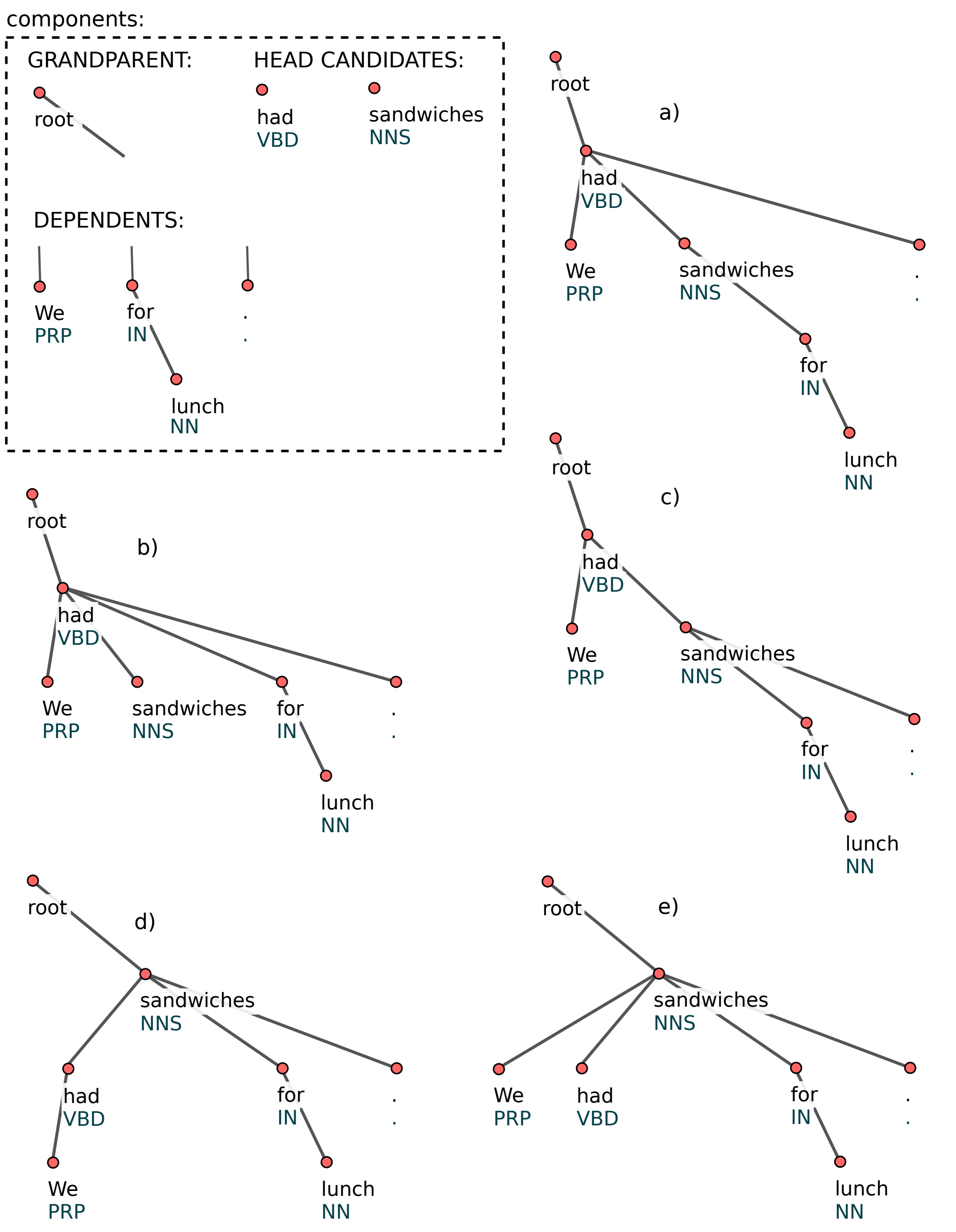


Figure 6.7: Small change example from Figure 6.6 in the perspective of dependencies. The two nodes *had* and *sandwiches* are the two candidates for the new head. Each of the three dependent subtrees is then attached to one of these candidates. All the possible trees that do not violate the projectivity constraint are depicted in a) to f).

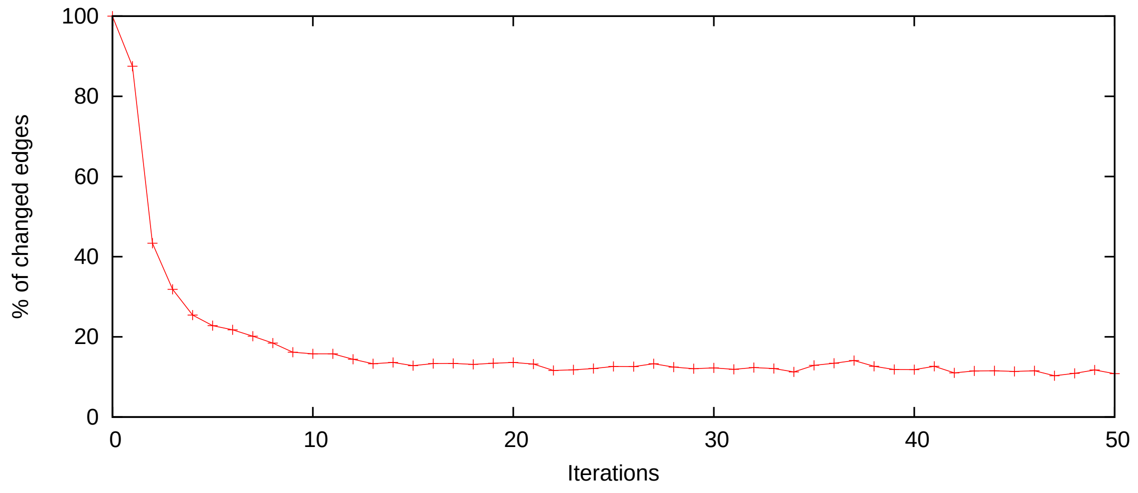


Figure 6.8: An example of the percentage of edges changed after each iteration. (Measured on English.)

the amount of "perturbations" in the trees. If the temperature is zero, the dependency trees become "frozen". During the sampling, each small change is chosen randomly according to a probability distribution proportional to the probabilities of individual changes. At the end of the sampling (in the last few iterations), we start decreasing the temperature, which means that the sampling is made on a sharper distribution than before. If we have n possible small changes with probabilities $p_1 \dots p_n$, we are sampling according to a new distribution $p_{1,T} \dots p_{n,T}$:

$$p_{x,T} = \frac{p_x^{1/T}}{\sum_{i=1}^{n} p_i^{1/T}}, \tag{6.6}$$

where the temperature T gradually decreases from one to zero. If $T = 1$, the original probability distribution is used. If $T \to 0$, the most probable change is always chosen since its probability in the new distribution is equal to one.

The third proposed method is to apply the maximum spanning tree (MST) algorithm (Chu and Liu, 1965) to create an "averaged" treebank based on the individual samples during the sampling. We skip so-called "burn-in" period, the first b iterations that change the initial random treebank very quickly to some more stable shape. In our experiments, we set b equal to 20. After this period, we count how many times the potential edges between all possible node pairs were sampled during the rest of the sampling. This counts are collected over the whole corpus with a *collection-rate* of 0.01, which means that we collect the counts roughly once per 100 small changes.

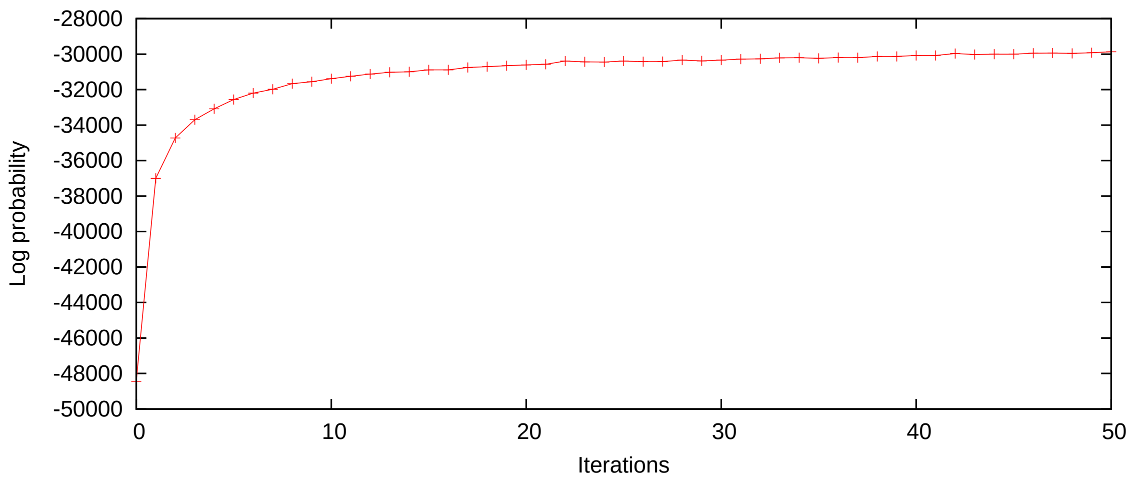


Figure 6.9: Increasing log-prob during iterations of the Gibbs sampler.

When the sampling is finished, we compute the scores of individual edges and build the most probable trees using MST. Since the MST maximizes the sum over the scores of used edges, it is natural to define the scores as logarithms of counts collected during the sampling. The MST maximization is then proportional to maximization of the individual counts.

$$T_{MST} = \arg\max_T \sum_{e \in T} \log count(e) = \arg\max_T \prod_{e \in T} count(e). \tag{6.7}$$

It is important to note that the MST algorithm may produce non-projective trees. Even if we average the strictly projective dependency trees, some non-projective edges may appear in the result. This might be an advantage since correct non-projective edges can be predicted, however, this relaxation may introduce mistakes as well.

Chapter 7

Experiments

7.1 Baselines

Before presenting experiments with our parser, we introduce several baselines for the unsupervised dependency parsing task.

The first baseline generates a completely random tree for a given sentence. An example of such random (generally non-projective) tree is depicted in Figure 7.1a. The algorithm proceeds as follows:

1. Select a random unattached word in the sentence.
2. Attach it to another word which is selected randomly, but only so as not to create a cycle.
3. Repeat the steps 1 and 2 until all words are attached.

This baseline is supposed to be very poor since the only knowledge used for constructing dependencies is the requirement of a tree shape.

The second baseline builds on the fact that the majority of dependencies in a tree should be projective. This fact follows from the properties of natural languages and treebank statistics in Table 4.2 confirm it as well. This baseline is identical to the projective initializer from Section 6.2.1, a pseudocode is shown in Figure 6.4. A random projective tree is depicted in Figure 7.1b.

The last two baselines are called "left chain" and "right chain". Here we make use of the fact that dependencies between words are rather short and many of them connect adjacent words. In the left chain baseline (see Figure 7.1c), each word is attached to the following neighbor and the last word is attached to the technical root. The right chain baseline (Figure 7.1d) works in the opposite way: Each word is attached to the previous one and the first word is attached to the root.

The average baseline scores computed on all our testing treebanks are shown in Table 7.1. It is interesting that some languages (or, more precisely, some manual annotations) are very left-oriented (e.g. Arabic, Italian, and Romanian), while other languages (Bengali, Japanese, Tamil, Telugu, Turkish) tend to have many more right-oriented dependencies. The scores for random projective trees are much

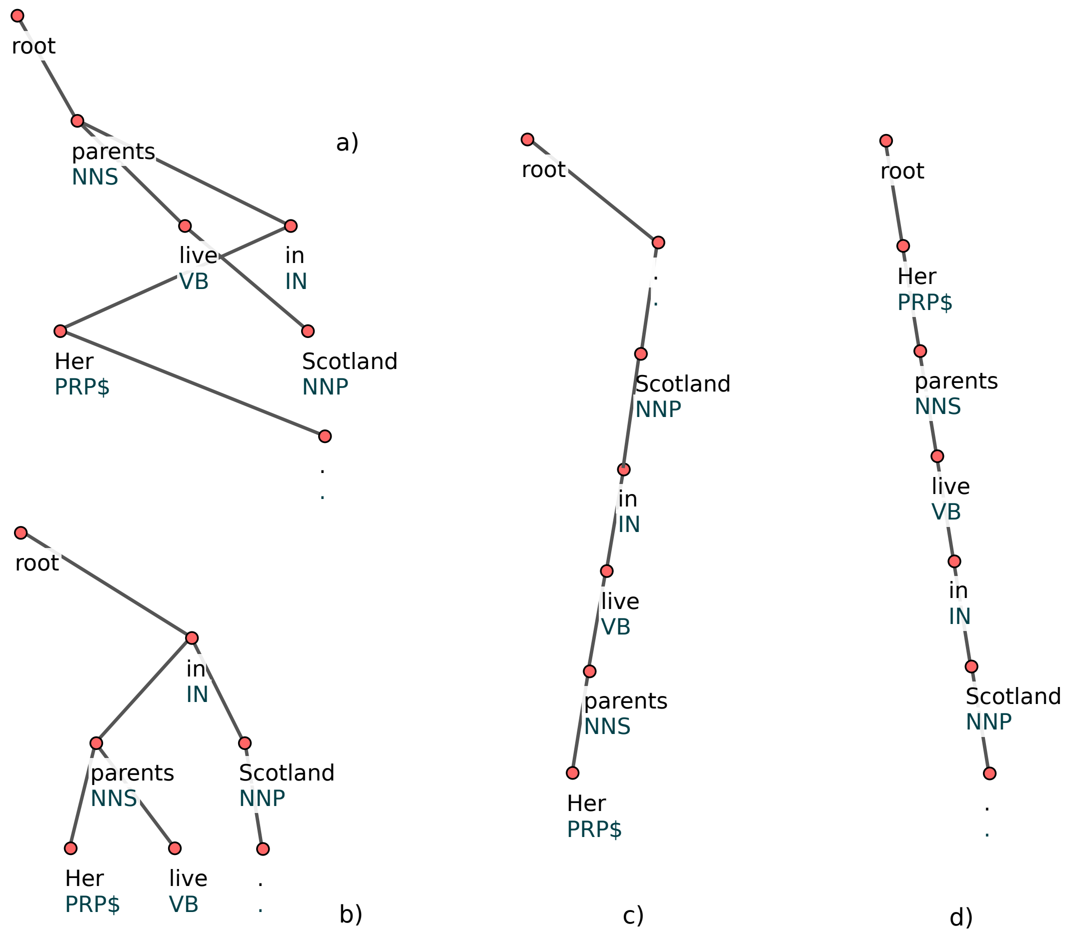


Figure 7.1: Examples of baselines for the unsupervised dependency parsing task: a) general non-projective random tree, b) projective random tree, c) left chain, d) right chain.

higher than for random non-projective trees not because of the projectivity, but due to shorter dependencies that are implicitly forced by the projectivity constraint. However, the best scoring baseline is always either the left chain or the right chain.

7.2 Preprocessing

7.2.1 Computing Reducibility Scores

One of the required preprocessing steps in order to use the dependency model defined in Section 5.5 is computing reducibility scores for the individual PoS n-grams as described in Section 5.4 so that we can use the reducibility model in the main

lang.	r.nproj	r.proj	left	right	lang.	r. nproj	r. proj	left	right
ar	2.5	17.5	7.9	**55.6**	hi	4.3	15.5	21.5	**27.8**
bg	6.9	15.9	18.8	**32.5**	hu	5.4	14.2	**35.9**	6.2
bn	18.9	29.4	**52.0**	5.3	it	4.8	18.0	20.4	**42.8**
ca	3.3	15.5	23.8	**25.9**	ja	12.6	22.4	**49.7**	24.5
cs	6.4	17.8	**26.9**	25.2	la	6.5	13.9	**18.6**	18.4
da	5.4	16.4	11.6	**41.9**	nl	7.1	17.8	24.7	**31.2**
de	6.3	14.0	**22.4**	18.0	pt	5.0	16.0	22.9	**27.0**
el	4.1	17.0	**33.6**	16.9	ro	9.5	21.2	18.0	**46.1**
en	4.0	15.2	23.7	**25.4**	ru	11.5	20.3	23.3	**35.5**
es	3.2	15.3	23.6	**25.7**	sl	6.6	16.0	**26.8**	19.5
et	13.3	16.9	**28.8**	15.5	sv	6.8	16.5	**24.8**	24.3
eu	7.8	18.3	24.6	**35.7**	ta	6.5	20.9	**48.4**	9.5
fa	4.9	16.3	17.5	**37.6**	te	25.1	42.4	**65.5**	2.5
fi	8.0	16.3	**34.3**	13.9	tr	6.8	23.4	**65.5**	1.6
grc	9.8	17.5	**26.6**	17.7	zh	13.3	22.1	**40.7**	14.0

Table 7.1: Directed attachment scores (DAS) for random non-projective baseline (r.nproj), random projective baseline (r.proj), left chain baseline (left), and right chain baseline (right). Random baselines were averaged over 10 runs.

inference procedure. We obtain the reducibility scores from Wikipedia monolingual raw corpora (Section 4.1), which must be first automatically tokenized and processed by a part-of-speech tagger.

The segmentation to sentences and tokenization is done by a simple rule-based script consisting of a sequence of regular expressions. Most of the rules are common for all the languages. Only some rules, such as separating *d'* and *l'* in Catalan, separating of *'re*, *'s*, *'m* in English, dealing with different quotation marks or with different full stop marks, are language specific. We developed such tokenization rules that best fit the evaluation treebanks; however, there still remain many character sequences that are tokenized differently. Note that we tokenize only the data that are used for computation of reducibility scores and induction of word classes. The inference itself is done on the same treebank as the evaluation.

For the assignment of supervised PoS tags, we used the TnT tagger (Brants, 2000). We trained it on the training part of the respective treebanks. The quality of the trained taggers is not very high since we do not use any lexicons[1] or pretrained models. However, we show that it is sufficient for obtaining useful reducibility scores (Tables 5.1, 5.2, and 5.3).

We were not able to do this preprocessing correctly for five languages: There are no Wikipedia articles written in Ancient Greek (grc). The words in Chinese (zh)

[1]Using lexicons or another pretrained models for tagging would mean using other sources of human annotated data, which is not allowed if we want to compare our results with others.

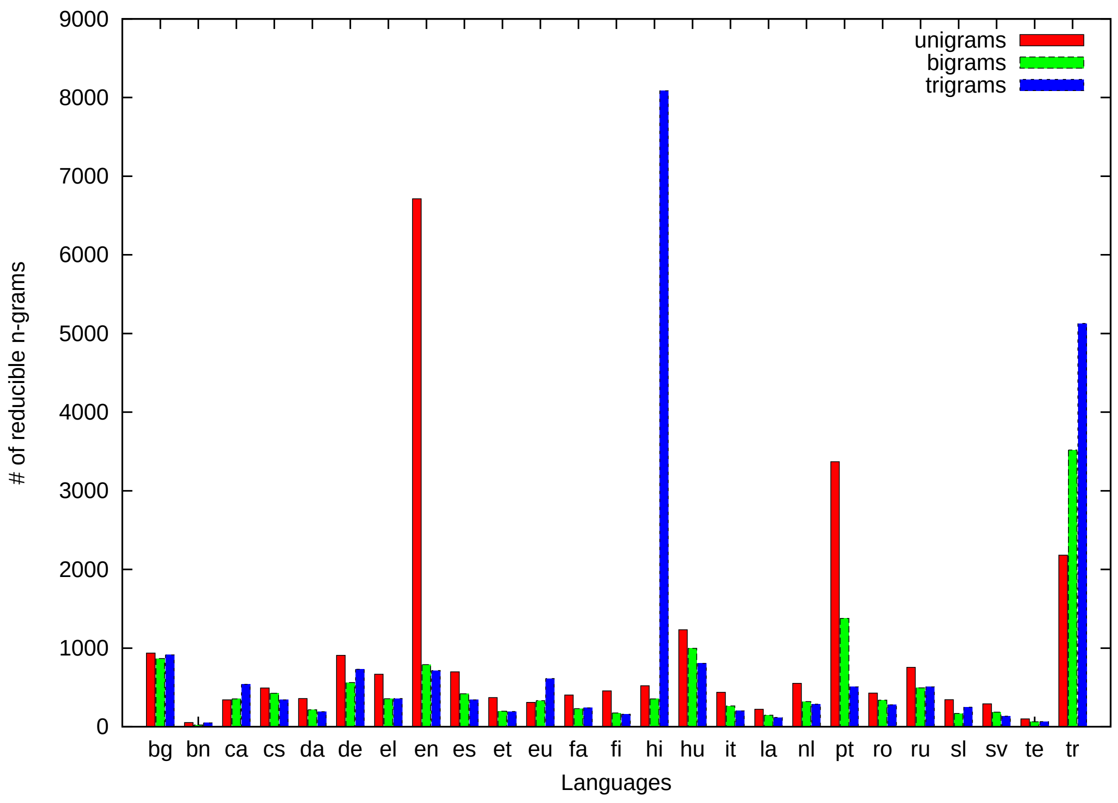


Figure 7.2: Numbers of unigrams, bigrams and trigrams that were considered as reducible in Wikipedia monolingual corpora.

and Japanese (ja) are not separated by spaces and we do not have an appropriate segmenter for them. The Tamil (ta) treebank is transcribed to Latin characters, whereas the Wikipedia articles are not. The TnT tagger cannot process Arabic (ar) texts properly.

All the 25 remaining languages were preprocessed correctly so that we could compute the reducibility scores of the individual PoS n-grams. The quality of the scores depends on the size of the corpus and on the number of word n-grams that were considered as reducible. The numbers of reducible unigrams, bigrams, and trigrams are summarized in Figure 7.2. We can see that for Bengali (bn) and Telugu (te), there were less than 50 reducible words found, which is probably not enough to determine reliable reducibility scores for all the PoS tags. We also have identified several repeating patterns of sentences that cause some abnormally high numbers of reducible words. For example, the majority of English (en) reducible unigrams are cardinal numbers in just one sentence type – "The per capita income for the city was \$$X$.", where X stands for a number. Such a sentence occurs in Wikipedia descriptions of all U.S. cities. Unfortunately, the number is missing in one such sentence and that is why all the numbers become reducible. Similar problems appeared in Hindi (hi) trigrams.

7.2.2 Unsupervised Part-of-speech Induction

Besides the standard supervised part-of-speech tags, we also experiment with unsupervised word classes. We use the same tokenized Wikipedia corpora as in the previous section and employ the best available word clustering tool. According to Christodoulopoulos et al. (2010) who compare the quality of various tools, one of the best is Alex Clark's *POSinduction* tool (Clark, 2003). This tool is used also by Spitkovsky et al. (2011a) in his DMV-based approach to dependency parsing.

The disadvantage of the *POSinduction* tool is the fact that it is limited to the ASCII encoding. Moreover, it employs individual characters in morphology prediction, so the simple substitution of characters by their UTF-8 codes is not sufficient. We have developed transliteration rules for converting European non-ASCII characters of European languages including the Cyrillic and Greek alphabet. However, other languages (zh, ja, fa, hi, ta, te, bn, ar) remain unconvertible.

The *POSinduction* tool requires to specify the number of word-classes we want to induce. We experiment with 25, 50, 100, and 200 word classes for all the languages. The induction is executed using the Wikipedia corpora together with training and testing sets of the corresponding treebanks.

7.3 Experimental Settings

In Chapters 5 and 6, we discussed possible models, methods, and procedures that might be helpful in the unsupervised parsing task. The variables we have for various experimental settings are summarized in the following list.

- *Languages/treebanks* – The experiments will be performed for all 30 languages for which we have their treebanks available in HamleDT (see Section 4.2).

- *Part-of-speech tags* – A majority of the treebanks in HamleDT have two available types of part-of-speech tags: the full tags (the fifth column in the CoNLL format), and coarse-grained tags (the fourth column in the CoNLL format). In addition, we can use word classes induced in an unsupervised way (see Section 7.2.2). For each language, we have four different sets of word classes differing by their numbers: 25, 50, 100, and 200. Therefore we have six different part-of-speech sets; we denote them as: POS, CPOS, WC25, WC50, WC100, and WC200.

- *Induction with included/excluded punctuation* – Many unsupervised parsing systems exclude punctuation marks from learning. We will also experiment with this option. This requires excluding punctuation from the evaluation as well.

- *Different models and their parameters* – We can experiment with various combinations of our models (edge model, fertility model, distance model, reducibility model) and their parameters.
- *Decoding procedure* – We have implemented two possible decoding procedures: Maximum spanning tree algorithm and annealing.
- *Evaluation* – We use three different evaluation metrics: directed attachment score (DAS), undirected attachment score (UAS) and neutral edge direction (NED). See section 4.3 for details. Moreover, a common practice in evaluation is to exclude punctuation.

We will not show here the results for all the combinations (30 languages $\times$ 6 part-of-speech sets $\times$ many model combinations with different hyperparameters $\times$ 2 decoding procedures $\times$ three different evaluation methods). Instead, we will go through the most interesting settings and compare the results by switching individual variables (PoS tags, models, evaluation methods, etc.).

All the inference experiments are performed and evaluated on the testing parts of the HamleDT treebanks (see Section 4.2 for their sizes) since the proposed sampling method is relatively slow.[2]

7.3.1 Standard setting

We have selected one of the possible settings as the base of our experiments and denote it as the *standard setting*. This setting, which emerged as the best one during the time of developing this parser, employs the default model defined in Equation 5.24. It uses fine-grained PoS tags (POS), decoding is done using maximum spanning tree algorithm, and punctuation marks are included both in the learning and in the evaluation.

All the following experiments compare this *standard setting* with other settings, where one of the variables is changed. In Section 7.3.3, we experiment with different tag sets. The impact of excluding punctuation is explored in Section 7.3.4. We have also examined the effects of: using unsupervised PoS tags (Section 7.3.5), removing one of the four models (Section 7.3.6), adding lexicalized model (Section 7.3.7), and using different evaluation metrics (Section 7.3.8).

7.3.2 Setting the Hyperparameters

First of all, we need to set the unknown hyperparameters of our models from Chapter 5. The basic model used in our *standard setting* has four hyperparameters: α, β,

[2]The inference of dependency structures using a larger data might be beneficial mainly for the lexicalized models (see Section 7.3.7). However, it would required a parallelization of the learning process. We leave this for the future research.

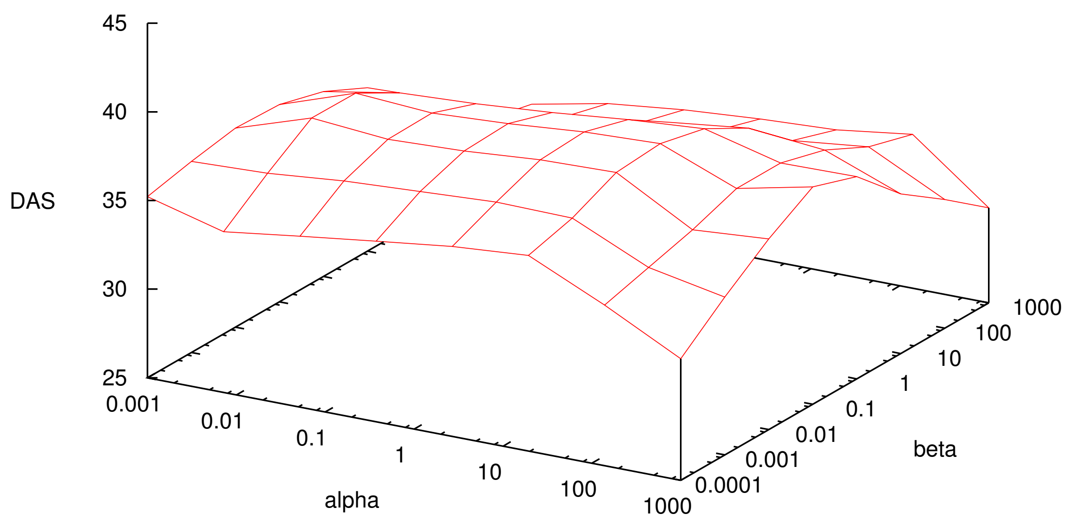


Figure 7.3: A graph of averaged directed attachment scores for different values of hyperparameters α and β. The other two hyperparameters are fixed: $\gamma = 1.5$, $\delta = 1$. Measured on English test data, punctuation is included.

γ, and δ. These numbers are the only parameters for tuning we have. Similarly to some previous papers on unsupervised parsing (Gillenwater et al., 2011; Spitkovsky et al., 2011c), the tuning experiments are performed on English only. The best parameters of the parser optimized for the English data will be then used for parsing all other languages. This simulates the situation in which we have only one treebank (English) on which we can tune our parser and we want to parse other languages for which we have no manually annotated treebanks.

This "tuning" is performed by exhaustive searching for the best four-tuple $[\alpha, \beta, \gamma, \delta]$ from the Cartesian product of the following sets:

$$\begin{aligned}\alpha &\in \{0.01,\ 0.1,\ 1,\ 10\},\\ \beta &\in \{0.001,\ 0.01,\ 0.1,\ 1\},\\ \gamma &\in \{1,\ 1.5,\ 2,\ 2.5\},\\ \delta &\in \{0.5,\ 1,\ 1.5,\ 2\}.\end{aligned}$$

The ranges for individual hyperparameters were set manually, based on our preliminary experiments with the individual parameters during the development. Moreover, the exact values of the hyperparameters are not crucial: The graphs showing directed attachment scores for different combinations of hyperparameter values are plotted in Figures 7.3 and 7.4. We can see that the "peak" area is very flat. Even the change of α or β by one order of magnitude does not affect the re-attachment

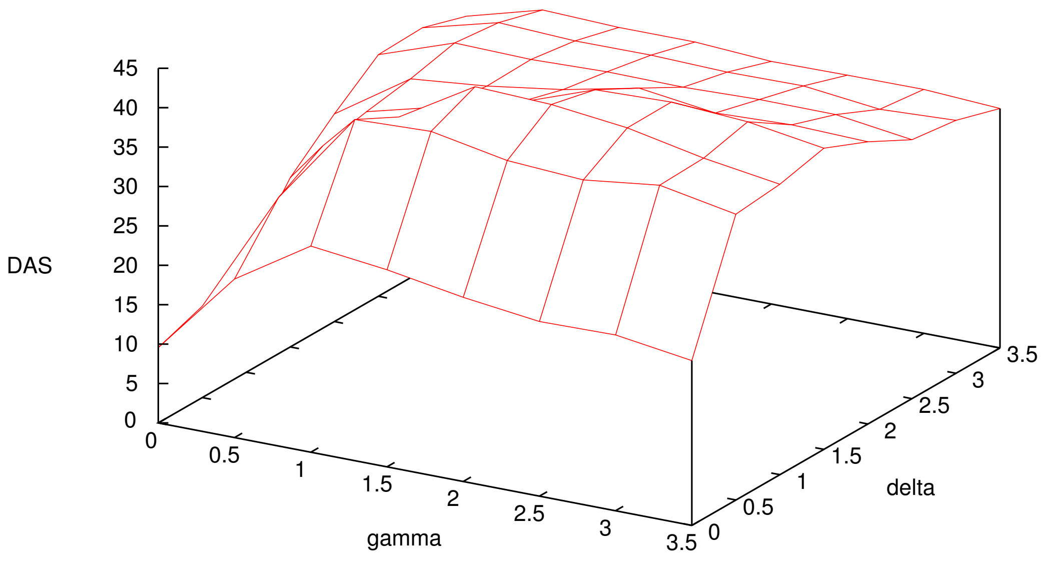


Figure 7.4: A graph of averaged directed attachment scores for different values of hyperparameters γ and δ. The other two hyperparameters are fixed: $\alpha = 1$, $\beta = 0.01$. Measured on English testing data, punctuation is included.

score very much. The hyperparameters γ and δ are expectedly more sensitive to their values, since the influence of respective models is directly connected with them.

We maximize the directed attachment score (including punctuation) on English test set and find the following optimal values:

$$\alpha = 1, \quad \beta = 0.01, \quad \gamma = 1.5, \quad \delta = 1$$

Note that here we make use of manually annotated trees, which might be considered as minimally supervised training. However, we use only the English treebank and we are setting only four numbers out of four previously manually estimated values. Similar searching could be done by changing the numbers and inspecting the outputs. So we believe such simple search does not violate the unsupervised nature of our experiments.

We experimented with searching for optimal parameters also for other treebanks and found out that they vary across languages. Therefore, adjusting the hyperparameters specifically for individual languages would improve their parsing quality.

7.3.3 Results on Supervised PoS Tags

The model with the hyperparameters tuned on English is now applied on all the testing languages. We perform this experiment both on fine-grained tags (POS) and coarse-grained tags (CPOS) and compare the results with the left and right baselines in Table 7.2. Since the sampling algorithm is not deterministic and every run can lead to different scores, we present the average scores and standard deviations based on 10 runs.

The parsing quality differs a lot across languages. Spanish (es), Catalan (ca), Portuguese (pt), Italian (it), Persian (fa), Swedish (sv) and other (bg, cs, en, nl, ru, sl) reached reasonably good parsing quality. It is noticeable that the best results were observed for languages from the Romance family. In the case of Bengali (bn), Estonian (et), or Romanian (ro), there are very big differences in parsing quality when using POS and CPOS tags. It is yet another evidence that the chosen tagset strongly influences the unsupervised parsing task. Very poor results were obtained for Hindi (hi), Latin (la), Telugu (te), and Turkish (tr). In a few cases, we were even not able to beat one of the left/right baselines.

lang	baselines	our		lang	baselines	our	
	left/right	CPOS	POS		left/right	CPOS	POS
ar	7.9/**55.6**	*22.8*±0.3	*26.4*±3.2	hi	21.5/**27.8**	22.3±0.3	18.3±0.4
bg	18.8/32.5	46.3±2.4	**46.5**±0.8	hu	35.9/6.2	35.6±9.8	**46.1**±4.2
bn	52.0/5.3	**54.0**±13.8	23.6±16.2	it	20.4/42.8	42.9±0.3	**53.0**±4.0
ca	23.8/25.9	**40.8**±1.5	40.5±1.9	ja	49.7/24.5	35.3±2.3	**36.1**±1.7
cs	26.9/25.2	42.7±0.7	**43.8**±0.7	la	18.6/18.4	**25.1**±0.3	24.6±0.3
da	11.6/**41.9**	38.1±1.4	37.0±2.1	nl	24.7/31.2	38.8±7.8	**43.2**±3.8
de	22.4/18.0	**38.0**±0.3	**38.0**±0.4	pt	22.9/27.0	47.9±2.7	**48.8**±3.3
el	33.6/16.9	22.5±0.7	**23.6**±3.0	ro	18.0/**46.1**	25.0±0.7	45.1±2.9
en	23.7/25.4	**44.1**±0.6	43.2±1.9	ru	23.3/35.5	**40.1**±0.3	40.0±0.6
es	23.6/25.7	**52.0**±0.4	**52.0**±0.3	sl	26.8/19.5	**40.6**±0.4	32.0±1.9
et	28.8/15.5	**47.8**±12.5	28.7±6.7	sv	24.8/24.3	**47.5**±1.0	47.3±0.9
eu	24.6/**35.7**	28.4±0.5	30.4±0.7	ta	**48.4**/9.5	*22.4*±0.7	*32.6*±4.7
fa	17.5/37.6	**51.1**±1.3	**49.0**±0.9	te	**65.5**/2.5	25.4±15.6	10.8±16.6
fi	34.3/13.9	30.4±0.4	**33.3**±1.0	tr	**65.5**/1.6	5.8±0.5	19.6±2.8
grc	**26.6**/17.7	*20.3*±2.4	*20.6*±1.7	zh	**40.7**/14.0	*14.8*±0.4	*31.1*±1.0

Table 7.2: Directed attachment scores and standard deviations for the basic settings with hyperparameters $\alpha = 1$, $\beta = 0.01$, $\gamma = 1.5$, and $\delta = 1$. The results of method using CPOS and POS tags are compared to the left/right baselines. The results in *italic* indicates that the reducibility model could not be properly applied due to the aforementioned problems in preprocessing. The **bold** results are the best ones for a particular language.

7.3.4 Learning and Evaluation Excluding Punctuation

In many previous works, the unsupervised parsers were evaluated excluding punctuation marks. The main reason for doing so was probably the fact that punctuation was harmful to the inference algorithms. In this experiment, we confirm this hypothesis. The learning without punctuation must be then evaluated without punctuation as well. We use the same setting as in the last experiment on POS tags. The only difference is that we exclude punctuation from evaluation (e-p) or even from learning (l-p). The results are compared with the standard learning (l+p e+p) in Table 7.3.

The experiment shows that if punctuation is included in learning but excluded in evaluation, the parsing quality increases for 24 out of 30 languages. This means that our parser makes more mistakes in (often very arbitrarily attached) punctuation marks than in attachment of other words. One such example is Czech (cs), where the score increased from 43.8% to 50.7%. This was caused by incorrectly attached full stops in all sentences, which depend on technical roots in the Czech treebank.

However, excluding punctuation from learning is beneficial only for 9 out of 30 languages. The biggest improvement in directed attachment score was achieved for Estonian (et), where the score increased from 34.2% to 42.2%.

lang	POS tags			lang	POS tags		
	l+p e+p	l+p e-p	l-p e-p		l+p e+p	l+p e-p	l-p e-p
ar	26.4 ±3.2	**27.7** ±3.5	23.5 ±4.2	hi	**18.3** ±0.4	17.4 ±0.3	10.8 ±0.7
bg	46.5 ±0.8	**49.0** ±0.8	45.7 ±1.3	hu	46.1 ±4.2	51.1 ±5.0	**52.0** ±1.8
bn	23.6 ±16.2	**23.8** ±17.1	22.6 ±16.4	it	**53.0** ±4.0	51.9 ±4.6	43.4 ±3.1
ca	40.5 ±1.9	42.6 ±2.1	**44.8** ±0.2	ja	36.1 ±1.7	**52.5** ±2.6	47.5 ±2.5
cs	43.8 ±0.7	**50.7** ±0.8	49.1 ±1.1	la	24.6 ±0.3	**27.7** ±0.4	26.6 ±0.3
da	37.0 ±2.1	**40.4** ±2.4	39.6 ±1.6	nl	43.2 ±3.8	41.7 ±4.4	**45.1** ±0.6
de	38.0 ±0.4	40.8 ±0.4	**40.9** ±0.8	pt	48.8 ±3.3	**54.9** ±3.6	52.8 ±3.2
el	23.6 ±3.0	**25.4** ±3.4	21.1 ±0.5	ro	45.1 ±2.9	45.1 ±2.9	**45.9** ±5.4
en	43.2 ±1.9	**48.0** ±1.5	42.5 ±2.8	ru	40.0 ±0.6	39.8 ±0.6	**41.7** ±3.8
es	52.0 ±0.3	**56.0** ±0.3	54.8 ±0.2	sl	32.0 ±1.9	**37.8** ±2.2	25.2 ±4.8
et	28.7 ±6.7	34.8 ±5.6	**42.2** ±4.6	sv	47.3 ±0.9	**49.9** ±1.0	48.9 ±0.9
eu	**30.4** ±0.7	27.2 ±1.0	25.2 ±0.2	ta	32.6 ±4.7	36.2 ±5.2	**36.9** ±5.4
fa	**49.0** ±0.9	47.9 ±1.0	36.5 ±4.0	te	10.8 ±16.6	**11.0** ±16.9	5.5 ±0.3
fi	33.3 ±1.0	**35.2** ±1.2	31.5 ±1.1	tr	19.6 ±2.8	**20.9** ±3.7	15.9 ±0.7
grc	20.6 ±1.7	**23.8** ±2.1	20.2 ±0.8	zh	31.1 ±1.0	31.2 ±1.1	**31.9** ±1.0

Table 7.3: Learning and evaluation excluding punctuation. "l" stands for learning, "e" for evaluation, "-p" and "+p" for excluding or including punctuation respectively.

7.3.5 Results on Unsupervised PoS Tags (Word Classes)

We run the same experiment as in Section 7.3.3 but for automatically induced word classes instead of the gold POS tags. Table 7.4 shows the results for 25 and 100 word classes. We can hardly determine which number of classes is better from these results. It seems that the ideal number can differ across languages. The results for 50 and 200 word classes were thoroughly similar but never outperform the standard setting results. Significant improvement when using word-classes instead of the manually designed POS tags was achieved on three languages: Greek (el; from 23.6% to 33.0%), Estonian (et; from 28.7% to 53.7%), and Turkish (tr; from 19.6% to 51.7%). Other seven languages reached similar scores at least for one number of the word-classes (bn, da, fi, ru, sv, te, zh). On the remaining languages, using word-classes instead of POS tags worsened the parsing quality.

We made a more detailed experiments for three languages (English, Estonian, and Swedish) to investigate the influence of various numbers of used word classes to the parsing quality. We induced unsupervised tag sets for 2, 4, 8, 16, 32, 64, 128, and 256 classes, recomputed the reducibility scores and tested our parser on these classes. The results, which are plotted in Figure 7.5, correspond with our expectations. The quality is low for very small number of word classes. It is necessary to distinguish at least the basic PoS tags, such as nouns, adjectives, verbs, prepositions etc. The results for 16 and 32 word classes are considerably better and the quality slightly decreases for higher numbers of word classes.

lang	standard setting	# of word-classes		lang	standard setting	# of word-classes	
		25	100			25	100
ar	**26.4** ±3.2	21.8 ±0.7	18.6 ±0.5	hi	**18.3** ±0.4	13.1 ±4.4	12.7 ±1.5
bg	**46.5** ±0.8	40.0 ±2.0	32.7 ±0.6	hu	**46.1** ±4.2	35.7 ±0.5	32.9 ±1.9
bn	23.6 ±16.2	23.8 ±1.5	**24.2** ±3.7	it	**53.0** ±4.0	37.6 ±2.1	36.4 ±0.6
ca	**40.5** ±1.9	17.5 ±0.2	17.5 ±0.2	ja	**36.1** ±1.7	27.3 ±5.1	30.4 ±4.7
cs	**43.8** ±0.7	16.2 ±0.3	25.9 ±0.6	la	**24.6** ±0.3	18.5 ±0.9	16.5 ±0.2
da	**37.0** ±2.1	35.7 ±1.3	28.5 ±0.5	nl	**43.2** ±3.8	17.6 ±0.4	23.7 ±1.1
de	**38.0** ±0.4	30.5 ±0.2	29.2 ±0.4	pt	**48.8** ±3.3	41.7 ±1.3	38.0 ±1.9
el	23.6 ±3.0	33.0 ±1.2	**37.8** ±1.2	ro	**45.1** ±2.9	41.4 ±0.6	28.3 ±1.1
en	**43.2** ±1.9	14.7 ±0.6	37.2 ±0.8	ru	40.0 ±0.6	44.7 ±0.5	**45.8** ±1.3
es	**52.0** ±0.3	20.3 ±0.3	26.0 ±0.7	sl	**32.0** ±1.9	16.4 ±0.2	20.9 ±0.2
et	28.7 ±6.7	**53.7** ±0.6	23.1 ±2.3	sv	**47.3** ±0.9	46.7 ±1.4	35.1 ±0.4
eu	**30.4** ±0.7	22.1 ±0.1	26.1 ±0.1	ta	**32.6** ±4.7	18.3 ±0.6	17.9 ±0.5
fa	**49.0** ±0.9	17.8 ±0.3	15.6 ±0.3	te	10.8 ±16.6	**17.6** ±6.1	15.2 ±5.9
fi	**33.3** ±1.0	32.5 ±0.3	26.9 ±3.3	tr	19.6 ±2.8	**51.7** ±0.2	26.1 ±1.8
grc	**20.6** ±1.7	17.7 ±1.0	16.4 ±0.7	zh	31.1 ±1.0	**31.7** ±6.2	25.5 ±9.9

Table 7.4: Directed attachment scores for unsupervised word classes compared to the basic setting results (PoS).

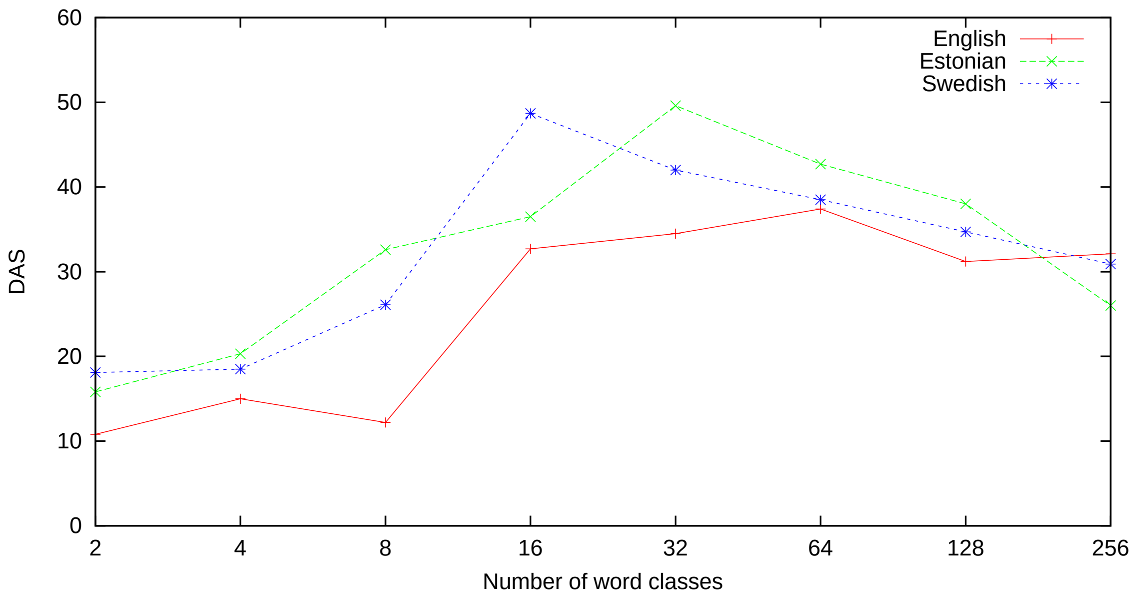


Figure 7.5: Directed attachment scores for different number of word classes.

7.3.6 Impact of Individual Models

To investigate the impact of individual components of the model, we experiment with removing one of them (ablation analysis). Table 7.5 shows the directed attachment scores when the distance model or the reducibility model is omitted. We can see that using all the four models is not always beneficial. For example, Bengali (bn), Russian (ru), or Telugu (te) performs better without the distance model. Eight other languages (ar, el en, hi, ja, ro, ta, tr) perform better without the reducibility model.[3]

Another experiment is focused on different fertility models. We compare the results of parser with the standard fertility model (Equation 5.18), with the simpler fertility model with symmetric Dirichlet prior (Equation 5.17), and with no fertility model at all. Table 7.6 shows that if we move from the symmetric Dirichlet prior to the prior derived from word frequency, we cannot make it significantly worse. Moreover, there are languages for which the non-uniform Dirichlet prior was very helpful: Spanish (es; from 43.6% to 52.0%), Hungarian (hu; from 33.4% to 46.1%). Complete removal of the fertility model usually means lower parsing quality. However, there are exceptions as well. The greatest improvement caused by removing the fertility model was achieved for Telugu (te; from 10.8% to 52.6%).

[3]It was caused probably by the fact, that reducibility scores were not computed properly due to some errors during preprocessing (see Section 7.2).

lang	standard setting	without dist. m.	without red. m.	lang	standard setting	without dist. m.	without red. m.
ar	26.4 ±3.2	17.4 ±1.1	**36.7** ±21.2	hi	18.3 ±0.4	20.6 ±0.5	**32.0** ±5.7
bg	**46.5** ±0.8	36.7 ±1.3	25.9 ±8.2	hu	**46.1** ±4.2	15.6 ±1.5	31.9 ±3.2
bn	23.6 ±16.2	**50.4** ±12.4	16.1 ±4.6	it	**53.0** ±4.0	20.3 ±2.4	42.9 ±8.8
ca	**40.5** ±1.9	30.1 ±0.6	26.6 ±0.6	ja	36.1 ±1.7	28.9 ±3.0	**61.8** ±4.3
cs	**43.8** ±0.7	22.4 ±6.0	22.4 ±1.8	la	**24.6** ±0.3	23.5 ±0.7	16.2 ±1.7
da	**37.0** ±2.1	21.2 ±5.5	18.3 ±9.9	nl	**43.2** ±3.8	21.1 ±5.9	34.3 ±2.7
de	**38.0** ±0.4	27.6 ±1.0	23.4 ±1.1	pt	**48.8** ±3.3	40.3 ±3.2	37.4 ±7.9
el	23.6 ±3.0	16.6 ±0.4	**35.6** ±3.2	ro	45.1 ±2.9	45.1 ±7.1	**55.7** ±11.0
en	**43.2** ±1.9	14.2 ±0.7	25.2 ±2.3	ru	40.0 ±0.6	**50.9** ±1.6	22.9 ±6.0
es	**52.0** ±0.3	5.4 ±6.9	28.2 ±4.1	sl	**32.0** ±1.9	23.3 ±1.1	24.1 ±1.4
et	**28.7** ±6.7	14.8 ±1.3	24.9 ±3.2	sv	**47.3** ±0.9	25.3 ±1.0	25.1 ±1.8
eu	30.4 ±0.7	10.4 ±0.6	**37.4** ±3.4	ta	32.6 ±4.7	26.0 ±4.9	**34.8** ±15.3
fa	**49.0** ±0.9	32.6 ±1.8	25.8 ±1.3	te	10.8 ±16.6	**64.5** ±19.2	18.5 ±16.2
fi	**33.3** ±1.0	29.3 ±1.7	25.8 ±6.1	tr	19.6 ±2.8	25.1 ±1.3	**36.0** ±7.2
grc	20.6 ±1.7	**27.1** ±2.5	20.2 ±1.8	zh	**31.1** ±1.0	24.2 ±0.3	29.0 ±1.5

Table 7.5: Comparison of directed attachment scores in case the distance and reducibility model is omitted.

l.	standard fert. m.	symmetric prior	without fert. m.	l.	standard fert. m.	symmetric prior	without fert. m.
ar	26.4 ±3.2	**27.7** ±0.6	23.2 ±0.5	hi	**18.3** ±0.4	**18.3** ±0.5	14.8 ±0.3
bg	**46.5** ±0.8	45.3 ±0.7	36.0 ±0.9	hu	**46.1** ±4.2	33.4 ±0.5	28.5 ±0.5
bn	23.6 ±16.2	30.5 ±18.1	**43.5** ±3.0	it	**53.0** ±4.0	42.2 ±2.0	37.7 ±0.9
ca	**40.5** ±1.9	36.8 ±5.1	31.4 ±0.3	ja	36.1 ±1.7	**37.6** ±2.7	31.9 ±0.4
cs	**43.8** ±0.7	42.2 ±1.1	39.3 ±0.5	la	24.6 ±0.3	**24.9** ±0.4	23.8 ±0.4
da	**37.0** ±2.1	34.7 ±1.2	30.2 ±0.5	nl	43.2 ±3.8	**44.7** ±2.4	33.1 ±0.5
de	**38.0** ±0.4	37.3 ±1.0	26.6 ±0.4	pt	**48.8** ±3.3	45.9 ±2.9	39.3 ±1.1
el	23.6 ±3.0	**25.2** ±2.8	25.0 ±0.5	ro	**45.1** ±2.9	42.4 ±5.2	41.8 ±0.6
en	**43.2** ±1.9	32.6 ±4.5	28.7 ±0.4	ru	40.0 ±0.6	45.6 ±5.9	**47.5** ±0.6
es	**52.0** ±0.3	43.6 ±10.4	31.6 ±0.2	sl	**32.0** ±1.9	30.5 ±4.3	30.7 ±0.5
et	**28.7** ±6.7	21.9 ±3.9	20.3 ±2.1	sv	**47.3** ±0.9	46.8 ±1.2	38.2 ±0.7
eu	30.4 ±0.7	**30.6** ±0.9	29.4 ±0.3	ta	**32.6** ±4.7	29.9 ±4.1	31.8 ±2.1
fa	**49.0** ±0.9	46.8 ±5.4	38.1 ±1.0	te	10.8 ±16.6	5.8 ±0.8	**52.6** ±1.0
fi	**33.3** ±1.0	33.0 ±0.9	30.1 ±0.3	tr	19.6 ±2.8	19.1 ±6.2	**24.5** ±0.2
grc	20.6 ±1.7	19.2 ±2.7	**24.1** ±0.7	zh	**31.1** ±1.0	30.7 ±0.7	27.2 ±0.5

Table 7.6: Comparison of directed attachment scores for different models of fertility.

7.3.7 Lexicalized Edge models

So far, the parser used only part-of-speech tags obtained in either supervised or unsupervised way. It has never looked on the words themselves. In this experiment, we add the lexicalized edge model, which was specified in Equation 5.12. This model supposes that the PoS tags are already generated and fills the word forms in the tree. A word form of a node is conditioned by the word form of its parent, by PoS tags of both the node and its parent and by the edge direction. We experiment with different settings of its Dirichlet hyperparameter α. However, this model does not improve the parsing quality for almost any language. The only exception is Bengali (bn), for which addition of the lexicalized model increased the directed attachment score from 23.6% to 35.8%.

7.3.8 Comparison of different metrics

In the last experiment, we do not change any parameters of the parser. We only evaluate its *standard setting* using three different evaluation measures described in Section 4.3. The majority of papers about unsupervised parsing report their results using DAS, but the UAS and NED are sometimes also mentioned (Gelling et al., 2012). So we provide the results of our standard parser setting using all these three metrics in Table 7.7. From the definitions of DAS, UAS and NED, it is obvious that the following inequalities hold:

$$DAS \leq UAS \leq NED$$

l.	DAS	UAS	NED	l.	DAS	UAS	NED
ar	26.4 ±3.2	42.3 ±0.4	53.0 ±0.4	hi	18.3 ±0.4	35.3 ±0.4	45.1 ±0.5
bg	46.5 ±0.8	53.6 ±0.8	63.8 ±0.6	hu	46.1 ±4.2	52.2 ±1.5	59.8 ±0.9
bn	23.6 ±16.2	44.1 ±9.9	56.0 ±9.5	it	53.0 ±4.0	60.7 ±2.1	68.6 ±0.8
ca	40.5 ±1.9	53.2 ±1.2	61.6 ±1.5	ja	36.1 ±1.7	54.9 ±0.8	70.2 ±0.7
cs	43.8 ±0.7	51.8 ±0.5	63.5 ±0.5	la	24.6 ±0.3	35.5 ±0.3	52.0 ±0.4
da	37.0 ±2.1	50.4 ±1.2	62.6 ±1.3	nl	43.2 ±3.8	52.9 ±1.8	69.4 ±1.6
de	38.0 ±0.4	46.8 ±0.3	56.9 ±0.3	pt	48.8 ±3.3	58.0 ±1.6	69.4 ±0.7
el	23.6 ±3.0	45.3 ±1.8	62.2 ±1.1	ro	45.1 ±2.9	55.6 ±1.7	67.2 ±1.1
en	43.2 ±1.9	52.7 ±1.2	67.6 ±0.7	ru	40.0 ±0.6	56.2 ±0.5	73.9 ±0.8
es	52.0 ±0.3	56.8 ±0.1	66.4 ±0.2	sl	32.0 ±1.9	44.9 ±0.8	57.3 ±0.6
et	28.7 ±6.7	44.0 ±5.1	58.0 ±3.7	sv	47.3 ±0.9	56.8 ±0.6	68.9 ±0.7
eu	30.4 ±0.7	47.2 ±0.4	58.6 ±0.7	ta	32.6 ±4.7	46.2 ±3.0	53.5 ±3.2
fa	49.0 ±0.9	55.9 ±0.4	65.0 ±0.5	te	10.8 ±16.6	40.5 ±9.7	58.1 ±6.9
fi	33.3 ±1.0	44.7 ±0.6	58.6 ±0.6	tr	19.6 ±2.8	47.1 ±0.5	50.1 ±0.7
grc	20.6 ±1.7	35.3 ±0.8	44.9 ±1.3	zh	31.1 ±1.0	46.8 ±0.7	59.2 ±0.8

Table 7.7: Evaluation of the standard parser setting using different metrics.

We will not provide here any specific analysis of influence of the errors in different treebanks on different metrics. We only remark that instead of finding an optimal metric for unsupervised parsing evaluation by comparing to the gold standard, we should rather focus on finding other evaluation methods which would not need manually annotated treebanks.

7.4 Error Analysis

So far, we have measured the quality of different parser settings by comparison with a gold standard treebank. Nevertheless, we are aware of the disadvantages of such evaluation, which were discussed in Section 4.3. In this section, we look into dependency trees we have induced by using the standard setting and analyze and explain some of the most substantial types of errors and interesting phenomena. Many such errors were caused due to the lack of any model using word forms (i.e. no lexicalized model was used in the standard setting).

- ***Prepositional phrases*** – Preposition (or apposition) should govern the noun in a prepositional group. One reason for this is that some verbs require particular prepositions. Our parser makes it sometimes reverse and attaches prepositions to nouns. Such errors appear for example in Slovene or in German prepositions with *am* and *im* (the tag APPRART, which is a determiner and a preposition in one word).

- ***Determiners*** – Determiners depend on nouns in the majority of treebanks. One exception is the Danish treebank, where it is reversed and nouns, together with their adjectival modifiers, depend on determiners. Interestingly, similar structures have been sometimes induced by our parser (e.g. in English and German) but all determiners were correctly placed as leaves in Danish, which is however incorrect according to the Danish treebank.

- ***Compound verbs*** – Compound verbs (e.g. *have been swimming* in English) that consists of one content (*swimming*) and one or more auxiliary verbs (*have* and *been*) occur very often in various languages. One of the verbs is often finite (*have*) and it may or may not be the content one. The ways how to structure compound verbs differ across treebanks. Our parser usually choose one verb as the head and attaches all other verbs and the verb arguments to it. This can cause large decrease in DAS, since if it chooses an incorrect verb, all the arguments are attached incorrectly too.

- ***Sequences of nouns*** – The structure of phrases that consist of more nouns is often induced badly. For example, the structure of English PoS tag sequence '*NN NN NN*' can be hardly recognized by our parser, since it does not look at word forms.

- ***Left/right chain*** – Our parser sometimes induced trees that were very close to right chain or left chain baselines. This happened e.g. for Turkish when experimenting with unsupervised PoS tags (Table 7.4). It was the best result for Turkish, since left-chain baseline is 65.5% of DAS and the induced trees were very close to it and achieved 51.7%.

- ***Attachment of prepositional and noun phrases*** – Attachment of such phrases makes problems to supervised parsers as well. Our parser makes many more such mistakes.

- ***Completely wrong trees*** – In cases when DAS falls below 20%, the dependency structure is often completely incorrect. For example, nouns depend on adjectives or verbs are leaves instead of being heads of the structures.

7.5 Comparison with Other Systems

7.5.1 Two Other Systems Evaluated on CoNLL Data

We compare our parser with two other systems that appeared in the last year NLP conference papers and reported very good results across various languages in CoNLL shared tasks (Buchholz and Marsi, 2006; Nivre et al., 2007). Note that the CoNLL datasets may differ from the HamleDT data sets, so the attachment scores might differ for some languages.

The two systems are described in Gillenwater et al. (2011) and Spitkovsky et al. (2011c) and we named them “Gillen.2011” and “Spitkov.2011” respectively. Both of them are based on Dependency Model with Valence (see Section 2.2). Since they provide results for several configurations of their parsers, we choose only the best one from each paper. We define the best configuration as the one with the highest average attachment score across all the tested languages. The results for “Gillen.2011” are taken from the best configuration in Table 7 in their paper. They provided only results on sentences of up to 10 tokens from CoNLL 2006 treebanks. The results for “Spitkov.2011” are taken from the best configuration in Table 6 in their paper.

The results are compared in Table 7.8. We can see that our parser outperforms the previously published approaches for majority of treebanks. In one case, it is better for all the ten data sets, in the other case, it is better for 15 out of 20 data sets. The average attachment scores, which are computed only from the results present for both compared parsers, also confirm the improvement.

However, it is important to note that we used an additional source of information, namely large raw corpora for computing reducibility scores, while the others probably used the CoNLL data only.

CoNLL			≤ 10 tokens		all sentences	
language	code	year	Gillen.2011	our parser	Spitkov.2011	our parser
Arabic	ar	06	–	40.5	16.6	**26.5**
Arabic	ar	07	–	42.4	**49.5**	27.7
Basque	eu	07	–	32.8	24.0	**27.2**
Bulgarian	bg	06	58.3	**59.0**	43.9	**49.0**
Catalan	ca	07	–	63.5	**59.8**	47.0
Czech	cs	06	53.2	**58.9**	27.7	**49.5**
Czech	cs	07	–	67.6	28.4	**50.7**
Danish	da	06	45.9	**52.8**	38.3	**40.4**
Dutch	nl	06	33.5	**42.4**	27.8	**41.7**
English	en	07	–	64.1	45.2	**49.2**
German	de	06	46.7	**60.8**	30.4	**44.8**
Greek	el	07	–	35.8	13.2	**25.4**
Hungarian	hu	07	–	63.2	34.7	**51.1**
Italian	it	07	–	50.5	**52.3**	43.3
Japanese	ja	06	57.7	**68.6**	50.2	**52.5**
Portuguese	pt	06	54.0	**66.0**	36.7	**54.9**
Slovenian	sl	06	50.9	**51.0**	32.2	**37.8**
Spanish	es	06	57.9	**67.3**	50.6	**51.9**
Swedish	sv	06	45.0	**62.9**	**50.0**	49.9
Turkish	tr	07	–	18.6	**35.9**	20.9
		Average:	50.3*	**59.0***	37.4	**42.1**

Table 7.8: Comparison of our parser with two other parsers "Gillen.2011" and "Spitkov.2011". The evaluation here is done on CoNLL data using directed attachment score (DAS) and excluding punctuation. The average score in the last line is computed across all comparable results, i.e. for comparison with "Gillen.2011" only the CoNLL'06 results are averaged (*).

7.5.2 Shared Task on Induction of Linguistic Structure

We have participated in the "PASCAL Challenge on Grammar Induction" shared tasks (Gelling et al., 2012). One of the tasks was unsupervised induction of dependency structures. Participants were given data sets extracted from ten different treebanks. Each data set consisted of three parts:

- *unlabeled training data* – These data are not provided with dependency structures and are intended for training the parsers.

- *labeled development data* – Data provided with dependency structures. Their purpose was parser quality checking.

- *unlabeled testing data* – The participants made their predictions on these data and submitted them for a central evaluation.

All the sets were provided with three types of part-of-speech tags: coarse grained (CPOS), fine-grained (POS), and universal (UPOS), a common tag set for all the languages.

The exact setting of our parser is described by Mareček and Žabokrtský (2012b) is very similar to the *standard setting* described here. We have tested our parser (marked as "Mar.") on all the three available tag sets (CPOS, POS, UPOS). Other participating systems (marked as "Bisk", "Blun.", "Søg", and "Tu") were submitted by Bisk and Hockenmaier (2012), Blunsom and Cohn (2010), Søgaard (2012), and Tu (2012) respectively.

file	Bisk std.	Blun. std.	Mar. CPOS	Mar. POS	Mar. UPOS	Søg. norul.	Søg. rul.	Tu std.
arabic_padt	23.5	48.7	12.7	**57.3**	52.0	33.9	46.5	54.1
basque_3lb	36.2	**45.9**	21.0	25.5	22.4	25.5	13.7	44.0
czech_pdt	32.1	38.0	**49.1**	42.9	44.1	32.9	40.9	48.8
danish_cdt	37.8	32.1	48.4	41.4	49.7	42.4	45.1	**50.2**
dutch_alpino	37.9	**49.2**	28.3	44.2	29.9	31.3	40.5	43.7
english_childes	**59.4**	45.8	54.2	44.2	49.3	48.1	51.9	53.8
english_ptb	50.4	**56.0**	41.0	50.3	37.5	32.8	42.5	55.5
portuguese_floresta	**65.2**	42.0	50.2	49.5	29.4	37.1	54.6	41.8
slovene_jos	35.4	52.8	30.4	40.8	26.7	28.4	37.7	**58.0**
swedish_talbanken	48.9	52.4	48.4	50.6	52.6	37.5	55.1	**57.3**
average	42.7	46.3	38.4	44.7	39.4	35.0	42.9	**50.7**

Table 7.9: Results of the "PASCAL Challenge on Grammar Induction" shared task. Directed attachment scores computed on all testing sentences excluding punctuation marks.

The results are summarized in Table 7.9.[4] According to the average scores across all languages, the best system is the one developed by Tu (50.7%). However, Tu's system was tuned for each language separately using the development data and therefore is not comparable to other unsupervised systems, which used the same setting for all the languages. Disregarding Tu's results, the best average score (46.3%) was achieved by Blunsom and Cohn (2010). Our system using the POS tagset was the second-best one, with the average directed attachment score of 44.7%. Søgaard (2012) submitted a baseline system based on universal hand-specified rules and reached a DAS of 42.9%, which was enough for the third place.

[4]This table was taken from `http://wiki.cs.ox.ac.uk/InducingLinguisticStructure/ResultsDep`, where the results computed on all sentences regardless of their length are available. Gelling et al. (2012) provide results computed only on sentences not exceeding 10 words.

Chapter 8

Conclusions

We have described and implemented a novel method for unsupervised induction of dependency trees. Our dependency parser uses a model that consists of four submodels: the edge model, the fertility model, the distance model, and the reducibility model. For the inference itself, we have designed a Gibbs sampling method capable of sampling dependency structures that adhere at all times to the restrictions of projective dependency trees. The main ideas of this thesis have been also published in NLP conference papers: the Gibbs sampling of dependency trees in (Mareček and Žabokrtský, 2011), the dependency models in (Mareček and Žabokrtský, 2012a) and the parser evaluation in the PASCAL Challenge shared task (Section 7.5.2) in (Mareček and Žabokrtský, 2012b).

The main asset of this work lies in the reducibility model, which is based on the fact that words which can be removed from a sentence without damaging its grammaticality are very often leaves in dependency trees. Similarly, reducible sequences of words are very often subtrees. In fact, we have combined two rather complementary views on dependency:

- frequent co-occurrence of head-dependent pair, which is expressed by edge model, versus
- reducibility of the dependent, which is expressed by the reducibility model.

No other published work on unsupervised parsing employs reducibility or a similar idea. Dominating approaches in unsupervised parsing are typically based on repeated patterns, and not on the possibility of a deletion inside a pattern.

Our parser has been tested on 30 languages. Thanks to the HamleDT collection of treebanks, we were able to evaluate the parser also on more "exotic" languages, such as Ancient Greek, Persian, Tamil, or Telugu. We have tested various settings of the model and the algorithm and confirm the conclusions of previous work in this field (Spitkovsky et al., 2011c), (Gillenwater et al., 2011): even a small change in the parser setting can drastically decrease or increase the parsing quality for a particular language. There is no ideal setting which would work reasonably well across all the languages. Some models are very helpful for some languages but very harmful for other languages.

If we look at the resulting trees induced by our unsupervised parser, we can see that the crucial dependency relations were mostly determined correctly. Adjectives depend on nouns, verbs are in the role of heads of sentences with nouns as their arguments. The syntactic positions of prepositions, articles, and other functional words are also induced correctly in many testing languages. Most errors stem from the lack of lexicalization, i.e. from not using the word forms. Unfortunately, the lexicalized model we have proposed is not able to improve the delexicalized one and rather makes the overall quality worse.

Almost for each language, we would probably be able to find a better set of hyperparameters for a model combination which would lead to a better parsing quality. However, we did not do that. Our goal was to find a language-independent universal method for induction of dependency trees based on a text corpus only, without any tuning on a treebank of a particular language. The research progress in this area over the last ten years promises that it could be possible and we believe that our work is one of the steps on this long journey.

Bibliography

Itzair Aduriz, Mara Jess Aranzabe, Jose Mari Arriola, Aitziber Atutxa, Arantza Daz de Ilarraza, Aitzpea Garmendia, and Maite Oronoz. Construction of a Basque dependency treebank. In *Proceedings of the 2nd Workshop on Treebanks and Linguistic Theories*, 2003.

Susana Afonso, Eckhard Bick, Renato Haber, and Diana Santos. "Floresta sintá(c)tica": a treebank for Portuguese. In *Proceedings of the 3rd International Conference on Language Resources and Evaluation (LREC)*, pages 1968–1703, 2002.

Nart B. Atalay, Kemal Oflazer, Bilge Say, and Informatics Inst. The annotation process in the Turkish treebank. In *Proceedings of the 4th International Workshop on Linguistically Interpreteted Corpora (LINC)*, 2003.

James K. Baker. Trainable grammars for speech recognition. In *Speech communication papers presented at the 97th Meeting of the Acoustical Society*, pages 547–550, 1979.

David Bamman and Gregory Crane. The Ancient Greek and Latin dependency treebanks. In Caroline Sporleder, Antal Bosch, and Kalliopi Zervanou, editors, *Language Technology for Cultural Heritage*, Theory and Applications of Natural Language Processing, pages 79–98. Springer Berlin Heidelberg, 2011. ISBN 978-3-642-20227-8.

Eckhard Bick, Heli Uibo, and Kaili Mrisep. Arborest – a VISL-style treebank derived from an Estonian constraint grammar corpus. In *Proceedings of Treebanks and Linguistic Theories*, 2004.

Yonatan Bisk and Julia Hockenmaier. Induction of Linguistic Structure with Combinatory Categorial Grammars. In *Proceedings of the NAACL-HLT Workshop on the Induction of Linguistic Structure*, pages 90–95, Montréal, Canada, June 2012. Association for Computational Linguistics.

Phil Blunsom and Trevor Cohn. Unsupervised induction of tree substitution grammars for dependency parsing. In *Proceedings of the 2010 Conference on Empirical*

Methods in Natural Language Processing, EMNLP'10, pages 1204–1213, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

Rens Bod. An all-subtrees approach to unsupervised parsing. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, ACL-44, pages 865–872, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics. doi: 10.3115/1220175.1220284.

Igor Boguslavsky, Svetlana Grigorieva, Nikolai Grigoriev, Leonid Kreidlin, and Nadezhda Frid. Dependency treebank for Russian: Concept, tools, types of information. In *Proceedings of the 18th conference on Computational linguistics-Volume 2*, pages 987–991. Association for Computational Linguistics Morristown, NJ, USA, 2000.

Gideon Borensztajn and Willem Zuidema. *Bayesian Model Merging for Unsupervised Constituent Labeling and Grammar Induction.* ILLC scientific publications. Institute for Logic, Language and Computation (ILLC), University of Amsterdam, 2007.

Sabine Brants, Stefanie Dipper, Silvia Hansen, Wolfgang Lezius, and George Smith. The TIGER treebank. In *Proceedings of the Workshop on Treebanks and Linguistic Theories*, Sozopol, 2002.

Thorsten Brants. TnT - A Statistical Part-of-Speech Tagger. *Proceedings of the sixth conference on Applied natural language processing*, pages 1–8, 2000.

Samuel Brody. It depends on the translation: unsupervised dependency parsing via word alignment. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP'10, pages 1214–1222, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

Peter F. Brown, Vincent J. Della Pietra, Stephen A. Della Pietra, and Robert. L. Mercer. The Mathematics of Statistical Machine Translation: Parameter Estimation. *Computational Linguistics*, 19:263–311, 1993.

Sabine Buchholz and Erwin Marsi. CoNLL-X shared task on multilingual dependency parsing. In *Proceedings of the Tenth Conference on Computational Natural Language Learning*, CoNLL-X '06, pages 149–164, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics.

Glenn Carroll and Eugene Charniak. Two Experiments on Learning Probabilistic Dependency Grammars from Corpora. In *Working Notes of the Workshop Statistically-Based NLP Techniques*, pages 1–13. AAAI, 1992.

Keh-Jiann Chen and Yu-Ming Hsieh. Chinese Treebanks and Grammar Extraction. In *Proceedings of the First International Joint Conference on Natural Language Processing (IJCNLP 2004)*, pages 655–663, March 2004.

Noam Chomsky. *Syntactic Structures*. Mouton classic. Mouton De Gruyter, 2002. ISBN 9783110172799.

Christos Christodoulopoulos, Sharon Goldwater, and Mark Steedman. Two decades of unsupervised POS induction: How far have we come? In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2010.

Y. J. Chu and T. H. Liu. On the Shortest Arborescence of a Directed Graph. *Science Sinica*, 14:1396–1400, 1965.

Alexander Clark. Combining distributional and morphological information for part of speech induction. *Proceedings of 10th European Chapter of Association of Computational Linguistics (EACL'03)*, pages 59–66, 2003.

Shay B. Cohen, Kevin Gimpel, and Noah A. Smith. Logistic normal priors for unsupervised probabilistic grammar induction. In *Neural Information Processing Systems*, pages 321–328, 2008.

Dóra Csendes, János Csirik, Tibor Gyimóthy, and András Kocsor. The Szeged treebank. In Václav Matoušek, Pavel Mautner, and Tomáš Pavelka, editors, *TSD*, volume 3658 of *Lecture Notes in Computer Science*, pages 123–131. Springer, 2005. ISBN 3-540-28789-2.

Mihaela Călăcean. Data-driven dependency parsing for Romanian. Master's thesis, Uppsala University, August 2008.

Hang Cui, Renxu Sun, Keya Li, Min-Yen Kan, and Tat-Seng Chua. Question answering passage retrieval using dependency relations. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR'05, pages 400–407, Salvador, Brazil, 2005. ACM.

Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal statistical society, Series B*, 39(1):1–38, 1977.

Sašo Džeroski, Tomaž Erjavec, Nina Ledinek, Petr Pajas, Zdeněk Žabokrtský, and Andreja Žele. Towards a Slovene dependency treebank. In *Proceedings of the Fifth International Language Resources and Evaluation Conference, LREC 2006*, pages 1388–1391, Genova, Italy, 2006. European Language Resources Association (ELRA).

Nicholas Evans and Stephen C. Levinson. The myth of language universals: Language diversity and its importance for cognitive science. *Behavioral and Brain Sciences*, 32(05):429–448, 2009.

Douwe Gelling, Trevor Cohn, Phil Blunsom, and Joao Graca. The PASCAL Challenge on Grammar Induction. In *Proceedings of the NAACL-HLT Workshop on the Induction of Linguistic Structure*, pages 64–80, Montréal, Canada, June 2012. Association for Computational Linguistics.

Kim Gerdes and Sylvain Kahane. Defining dependencies (and constituents). In *Proceedings of Dependency Linguistics 2011*, Barcelona, 2011.

W. R. Gilks, S. Richardson, and D. J. Spiegelhalter. *Markov chain Monte Carlo in practice*. Interdisciplinary statistics. Chapman & Hall, 1996. ISBN 9780412055515.

Jennifer Gillenwater, Kuzman Ganchev, João Graça, Fernando Pereira, and Ben Taskar. Posterior Sparsity in Unsupervised Dependency Parsing. *Journal of Machine Learning Research*, 12:455–490, February 2011. ISSN 1532-4435.

Sharon Goldwater. *Nonparametric Bayesian Models of Lexical Acquisition*. PhD thesis, Brown University, 2006.

Jan Hajič, Jarmila Panevová, Eva Hajičová, Petr Sgall, Petr Pajas, Jan Štěpánek, Jiří Havelka, Marie Mikulová, Zdeněk Žabokrtský, and Magda Ševčíková-Razímová. Prague Dependency Treebank 2.0. CD-ROM, Linguistic Data Consortium, LDC Catalog No.: LDC2006T01, Philadelphia, 2006.

Kenneth E. Harper and David G. Hays. The Use of Machines in the Construction of a Grammar and Computer Programm for Structural Analysis. In *Proceedings of the IFIP. Information Processing*, pages 188–194, Paris, France, 1959.

Jiří Havelka. Beyond Projectivity: Multilingual Evaluation of Constraints and Measures on Non-Projective Structures. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*, pages 608–615, 2007.

Katri Haverinen, Timo Viljanen, Veronika Laippala, Samuel Kohonen, Filip Ginter, and Tapio Salakoski. Treebanking Finnish. In Markus Dickinson, Kaili Mrisep, and Marco Passarotti, editors, *Proceedings of the Ninth International Workshop on Treebanks and Linguistic Theories (TLT9)*, pages 79–90, 2010.

William P. Headden, III, Mark Johnson, and David McClosky. Improving unsupervised dependency parsing with richer contexts and smoothing. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, NAACL '09, pages 101–109, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics. ISBN 978-1-932432-41-1.

Samar Husain, Prashanth Mannem, Bharat Ambati, and Phani Gadde. The ICON-2010 tools contest on Indian language dependency parsing. In *Proceedings of ICON-2010 Tools Contest on Indian Language Dependency Parsing*, Kharagpur, India, 2010.

Yasuhiro Kawata and Julia Bartels. Stylebook for the Japanese treebank in Verbmobil. In *Report 240*, Tbingen, Germany, September 29 2000.

Scott Kirkpatrick, Daniel C. Gelatt, and Mario P. Vecchi. Optimization by Simulated Annealing. *Science, Number 4598, 13 May 1983*, 220, 4598:671–680, 1983.

Dan Klein. *The Unsupervised Learning of Natural Language Structure.* PhD thesis, Stanford University, 2005.

Dan Klein and Christopher D. Manning. Corpus-based induction of syntactic structure: models of dependency and constituency. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, ACL '04, Stroudsburg, PA, USA, 2004. Association for Computational Linguistics.

Kevin Knight. Bayesian Inference with Tears. A tutorial workbook for natural language researchers, September 2009. URL `http://www.isi.edu/natural-language/people/bayes-with-tears.pdf`.

Philipp Koehn. *Statistical Machine Translation.* Cambridge University Press, 2009. ISBN 978-0521874151.

Matthias T. Kromann, Line Mikkelsen, and Stine Kern Lynge. Danish dependency treebank, 2004. URL `http://code.google.com/p/copenhagen-dependency-treebank/`.

Sandra Kübler, Ryan T. McDonald, and Joakim Nivre. *Dependency Parsing.* Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers, 2009.

Tom Kwiatkowski, Sharon Goldwater, Luke Zettelmoyer, and Mark Steedman. A Probabilistic Model of Syntactic and Semantic Acquisition from Child-Directed Utterances and their Meanings. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, 2012.

Karim Lari and Steve J. Young. The estimation of stochastic context-free grammars using the Inside-Outside algorithm. *Computer Speech and Language*, 4:35–56, 1990.

Markéta Lopatková, Martin Plátek, and Vladislav Kuboň. Modeling syntax of free word-order languages: Dependency analysis by reduction. In Václav Matoušek, Pavel Mautner, and Tomáš Pavelka, editors, *Lecture Notes in Artificial Intelligence, Proceedings of the 8th International Conference, TSD 2005*, volume 3658 of *Lecture Notes in Computer Science*, pages 140–147, Berlin / Heidelberg, 2005. Springer. ISBN 3-540-28789-2.

David M. Magerman and Mitchell P. Marcus. Parsing a natural language using mutual information statistics. In *Proceedings of the eighth National conference on Artificial intelligence - Volume 2*, AAAI'90, pages 984–989. AAAI Press, 1990. ISBN 0-262-51057-X.

Martin Majliš. Yet Another Language Identifier. In *Proceedings of the Thirteenth Conference of the European Chapter of the Association for Computational Linguistics (EACL 2012).* The Association for Computer Linguistics, April 2012.

Martin Majliš and Zdeněk Žabokrtský. Language Richness of the Web. In *Proceedings of the eighth international conference on Language Resources and Evaluation (LREC 2012)*, Istanbul, Turkey, May 2012. ELRA, European Language Resources Association.

Gary F. Marcus. Negative evidence in language acquisition. *Cognition*, 46:53–85, 1993. doi: 10.1016/0010-0277(93)90022-N.

Mitchell P. Marcus, Beatrice Santorini, and Mary A. Marcinkiewicz. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330, 1994.

David Mareček and Zdeněk Žabokrtský. Gibbs Sampling with Treeness constraint in Unsupervised Dependency Parsing. In *Proceedings of RANLP Workshop on Robust Unsupervised and Semisupervised Methods in Natural Language Processing*, pages 1–8, Hissar, Bulgaria, 2011.

David Mareček and Zdeněk Žabokrtský. Exploiting Reducibility in Unsupervised Dependency Parsing. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 297–307, Jeju Island, Korea, July 2012a. Association for Computational Linguistics.

David Mareček and Zdeněk Žabokrtský. Unsupervised Dependency Parsing using Reducibility and Fertility features. In *Proceedings of the NAACL-HLT Workshop on the Induction of Linguistic Structure*, pages 84–89, Montréal, Canada, June 2012b. Association for Computational Linguistics.

David Mareček, Martin Popel, and Zdeněk Žabokrtský. Maximum entropy translation model in dependency-based MT framework. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR*, WMT'10, pages 201–206, Uppsala, Sweden, 2010. Association for Computational Linguistics. ISBN 978-1-932432-71-8.

Ryan McDonald, Slav Petrov, and Keith Hall. Multi-source transfer of delexicalized dependency parsers. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 62–72, Edinburgh, Scotland, UK., July 2011. Association for Computational Linguistics.

Simonetta Montemagni, Francesco Barsotti, Marco Battista, Nicoletta Calzolari, Ornella Corazzari, Alessandro Lenci, Antonio Zampolli, Francesca Fanciulli, Maria Massetani, Remo Raffaelli, Roberto Basili, Maria Teresa Pazienza, Dario Saracino, Fabio Zanzotto, Nadia Mana, Fabio Pianesi, and Rodolfo Delmonte. Building the Italian syntactic-semantic treebank. In Anne Abeill, editor, *Building and using Parsed Corpora*, Language and Speech series, pages 189–210, Dordrecht, 2003. Kluwer.

Jens Nilsson, Johan Hall, and Joakim Nivre. MAMBA meets TIGER: Reconstructing a Swedish treebank from antiquity. In *Proceedings of the NODALIDA Special Session on Treebanks*, 2005. URL `http://www.msi.vxu.se/users/nivre/research/Talbanken05.html`.

Joakim Nivre, Johan Hall, Sandra Kübler, Ryan McDonald, Jens Nilsson, Sebastian Riedel, and Deniz Yuret. 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 2007. Association for Computational Linguistics.

Mark Paskin. Grammatical bigrams. In *Advances in Neural Information Processing Systems 14*. MIT Press, 2001.

Sharon G. Penner. Parental Responses to Grammatical and Ungrammatical Child Utterances. *Child Development*, 58(2):376–384, April 1987.

Prokopis Prokopidis, Elina Desipri, Maria Koutsombogera, Harris Papageorgiou, and Stelios Piperidis. Theoretical and practical issues in the construction of a Greek dependency treebank. In *In Proc. of the 4th Workshop on Treebanks and Linguistic Theories (TLT)*, pages 149–160, 2005.

Loganathan Ramasamy and Zdeněk Žabokrtský. Prague dependency style treebank for Tamil. In *Proceedings of LREC 2012*, Istanbul, Turkey, 2012.

Mohammad Sadegh Rasooli, Amirsaeid Moloodi, Manouchehr Kouhestani, and Behrouz Minaei-Bidgoli. A syntactic valency lexicon for Persian verbs: The first steps towards Persian dependency treebank. In *5th Language & Technology Conference (LTC): Human Language Technologies as a Challenge for Computer Science and Linguistics*, pages 227–231, Pozna, Poland, 2011.

Philip Resnik and Eric Hardisty. Gibbs Sampling for the Uninitiated. Technical Report LAMP-TR-153, University of Maryland, College Park, 2010.

Roy Schwartz, Omri Abend, Roi Reichart, and Ari Rappoport. Neutralizing linguistically problematic annotations in unsupervised dependency parsing evaluation. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 663–672, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.

Petr Sgall, Eva Hajičová, and Jarmila Panevová. *The Meaning of the Sentence in Its Semantic and Pragmatic Aspects.* D. Reidel, 1986. ISBN 9789027718389.

Kiril Simov and Petya Osenova. Extending the annotation of BulTreeBank: Phase 2. In *The Fourth Workshop on Treebanks and Linguistic Theories (TLT 2005)*, pages 173–184, Barcelona, December 2005.

Noah A. Smith and Jason Eisner. Guiding unsupervised grammar induction using contrastive estimation. In *Proceedings of IJCAI Workshop on Grammatical Inference Applications*, pages 73–82, 2005.

Otakar Smrž, Viktor Bielický, Iveta Kouřilová, Jakub Kráčmar, Jan Hajič, and Petr Zemánek. Prague Arabic dependency treebank: A word on the million words. In *Proceedings of the Workshop on Arabic and Local Languages (LREC 2008)*, pages 16–23, Marrakech, Morocco, 2008. European Language Resources Association. ISBN 2-9517408-4-0.

Anders Søgaard. Two baselines for unsupervised dependency parsing. In *Proceedings of the NAACL-HLT Workshop on the Induction of Linguistic Structure*, pages 81–83, Montréal, Canada, June 2012. Association for Computational Linguistics.

Valentin I. Spitkovsky, Hiyan Alshawi, Angel X. Chang, and Daniel Jurafsky. Unsupervised dependency parsing without gold part-of-speech tags. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP 2011)*, 2011a.

Valentin I. Spitkovsky, Hiyan Alshawi, and Daniel Jurafsky. Punctuation: Making a point in unsupervised dependency parsing. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning (CoNLL-2011)*, 2011b.

Valentin I. Spitkovsky, Hiyan Alshawi, and Daniel Jurafsky. Lateen EM: Unsupervised training with multiple objectives, applied to dependency grammar induction. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP 2011)*, 2011c.

Mihai Surdeanu, Richard Johansson, Adam Meyers, Llus Mrquez, and Joakim Nivre. The CoNLL-2008 shared task on joint parsing of syntactic and semantic dependencies. In *Proceedings of CoNLL*, 2008.

Mariona Taulé, Maria Antònia Martí, and Marta Recasens. AnCora: Multilevel annotated corpora for Catalan and Spanish. In *LREC*. European Language Resources Association, 2008.

Kewei Tu. Combining the Sparsity and Unambiguity Biases for Grammar Induction. In *Proceedings of the NAACL-HLT Workshop on the Induction of Linguistic Structure*, pages 105–110, Montréal, Canada, June 2012. Association for Computational Linguistics.

Leonoor van der Beek, Gosse Bouma, Jan Daciuk, Tanja Gaustad, Robert Malouf, Gertjan van Noord, Robbert Prins, and Begoa Villada. Chapter 5. the Alpino dependency treebank. In *Algorithms for Linguistic Processing NWO PIONIER Progress Report*, Groningen, The Netherlands, 2002.

Antony van der Mude and Adrian Walker. On the inference of stochastic regular grammars. *Information and Control*, 38(3):310–329, September 1978.

Deniz Yuret. *Discovery of Linguistic Relations Using Lexical Attraction*. PhD thesis, Massachusetts Institute of Technology, 1998.

Daniel Zeman, David Mareček, Martin Popel, Loganathan Ramasamy, Jan Štěpánek, Zdeněk Žabokrtský, and Jan Hajič. HamleDT: To Parse or Not to Parse? In *Proceedings of the eighth international conference on Language Resources and Evaluation (LREC 2012)*, Istanbul, Turkey, May 2012. European Language Resources Association.

Appendix A

Examples of Induced Trees

We provide examples of dependency trees induced by our parser using the standard setting, unless stated otherwise.

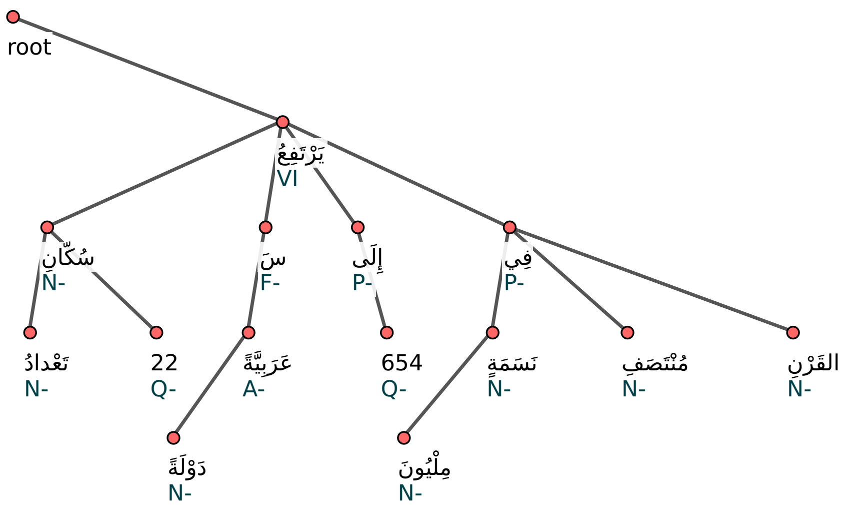


Figure A.1: Example of an induced Arabic dependency tree.

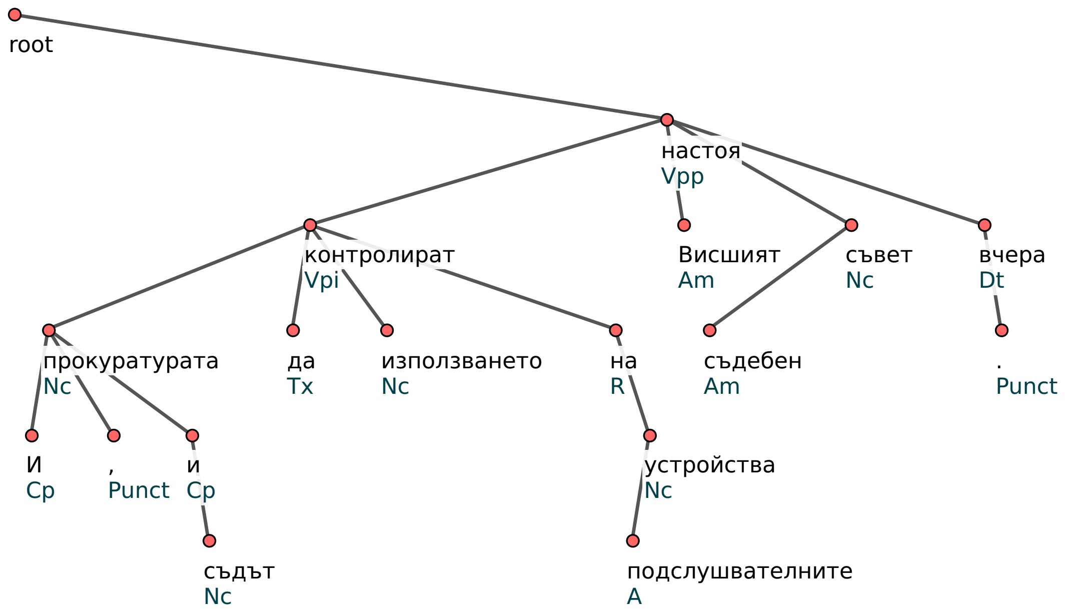


Figure A.2: Example of an induced Bulgarian dependency tree.

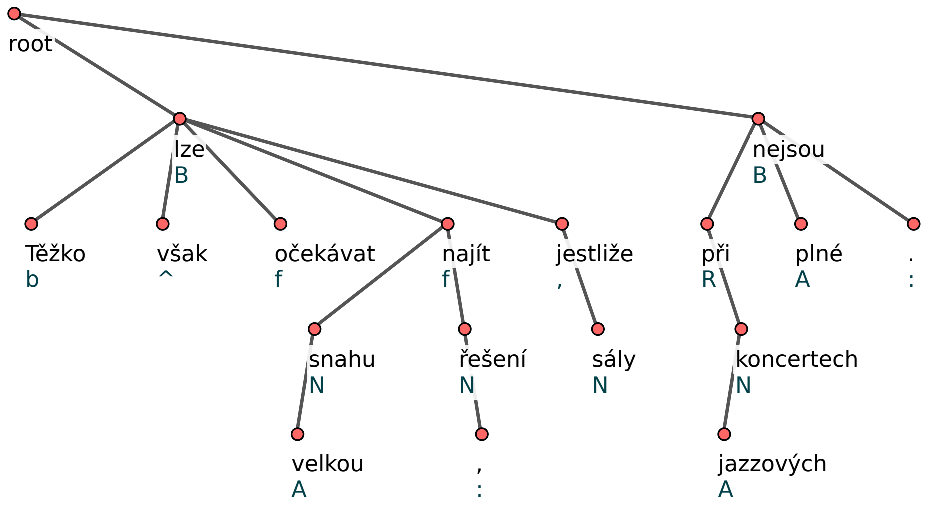


Figure A.3: Example of an induced Czech dependency tree.

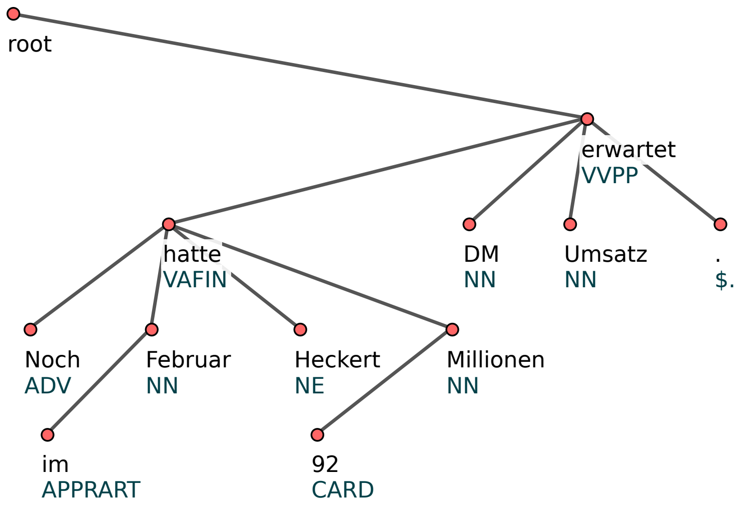


Figure A.4: Example of an induced German dependency tree.

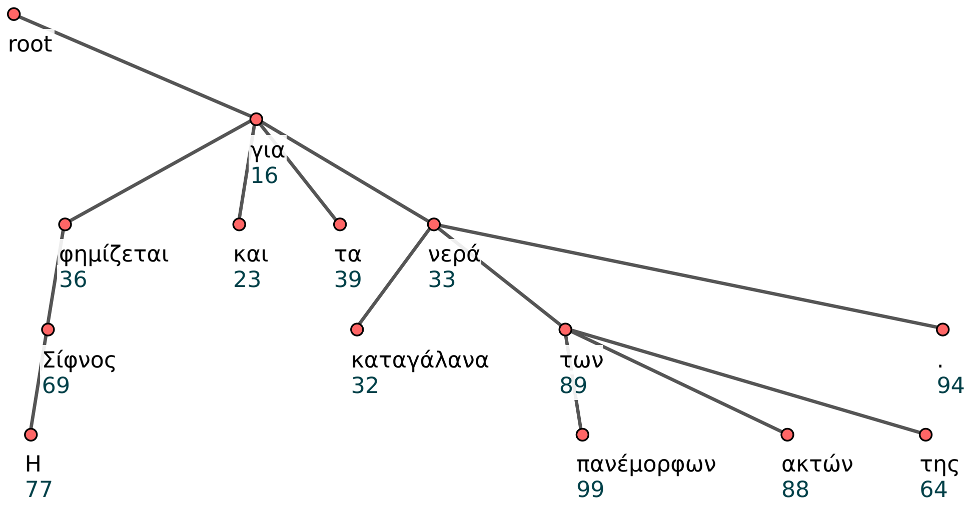


Figure A.5: Example of an induced Greek dependency tree using unsupervised part-of-speech tags (100 classes).

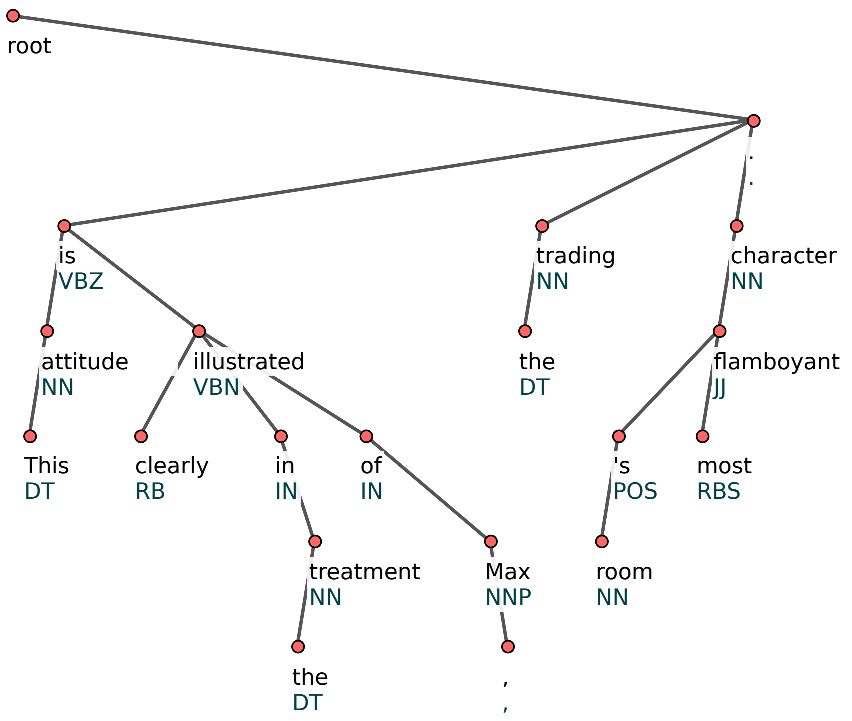


Figure A.6: Example of an induced English dependency tree.

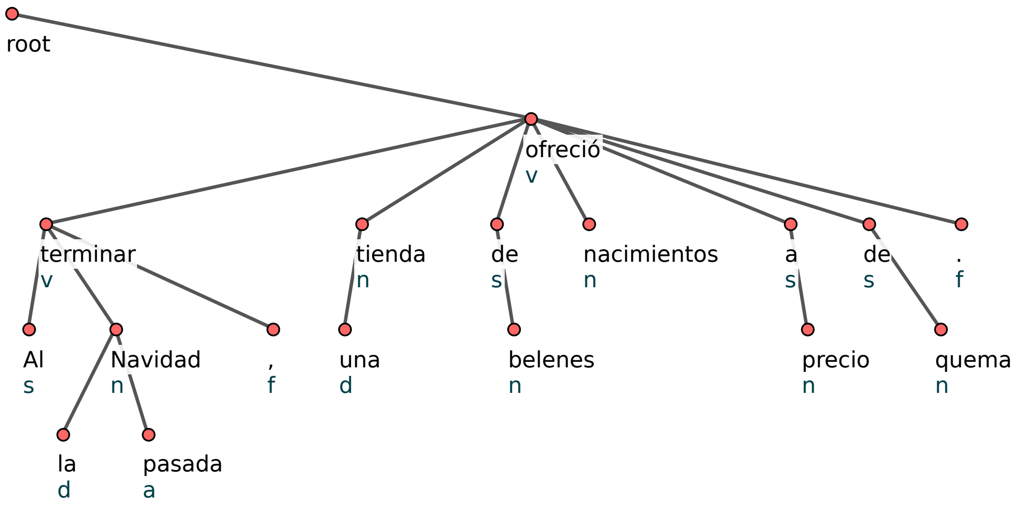


Figure A.7: Example of an induced Spanish dependency tree.

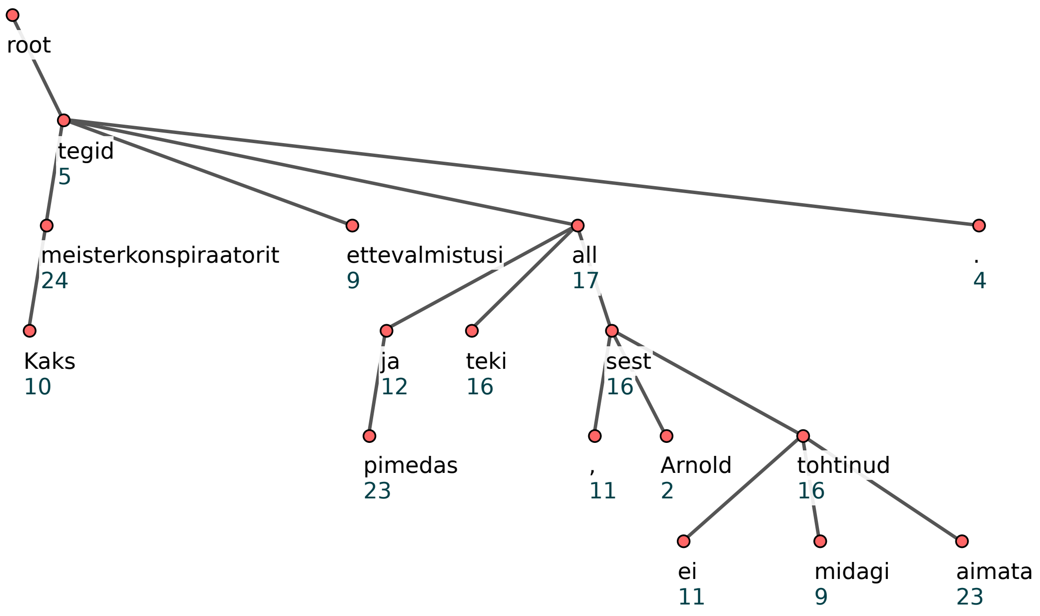


Figure A.8: Example of an induced Estonian dependency tree using unsupervised part-of-speech tags (25 classes).

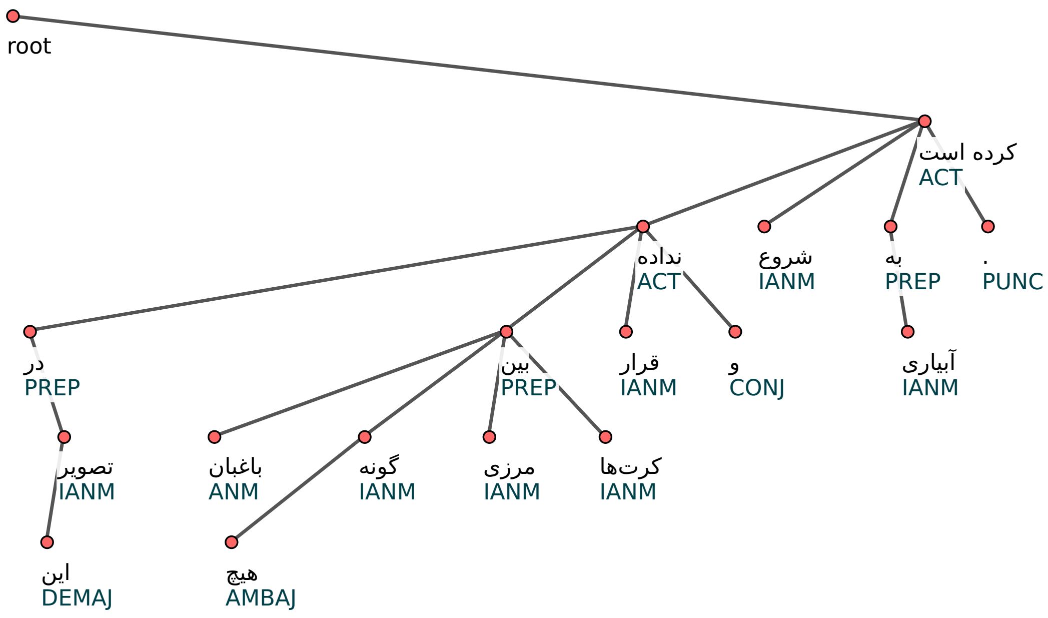


Figure A.9: Example of an induced Persian dependency tree.

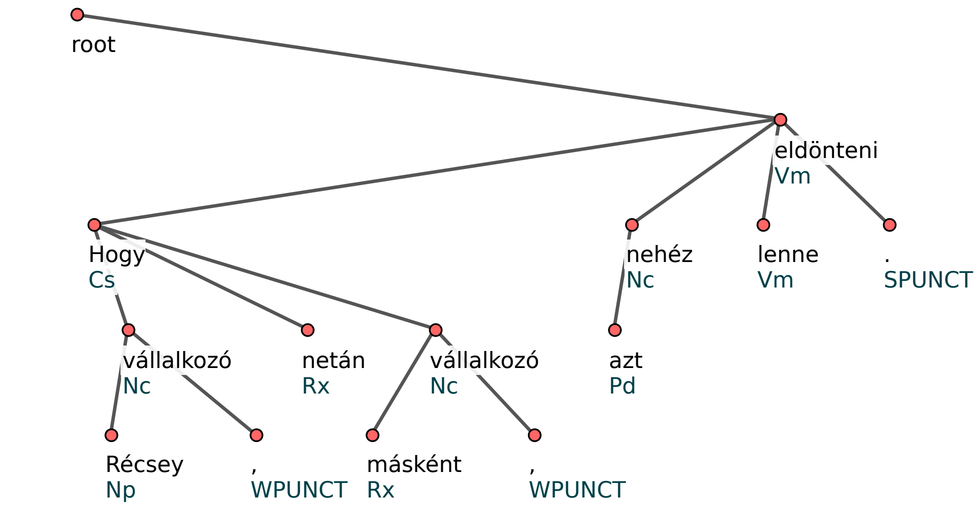


Figure A.10: Example of an induced Hungarian dependency tree.

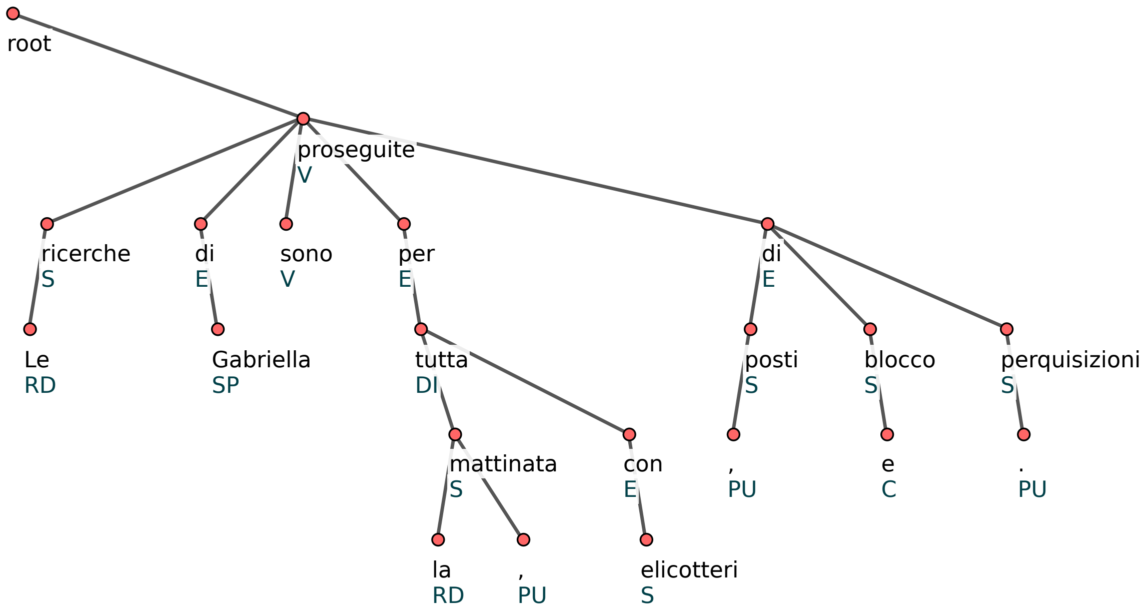


Figure A.11: Example of an induced Italian dependency tree.

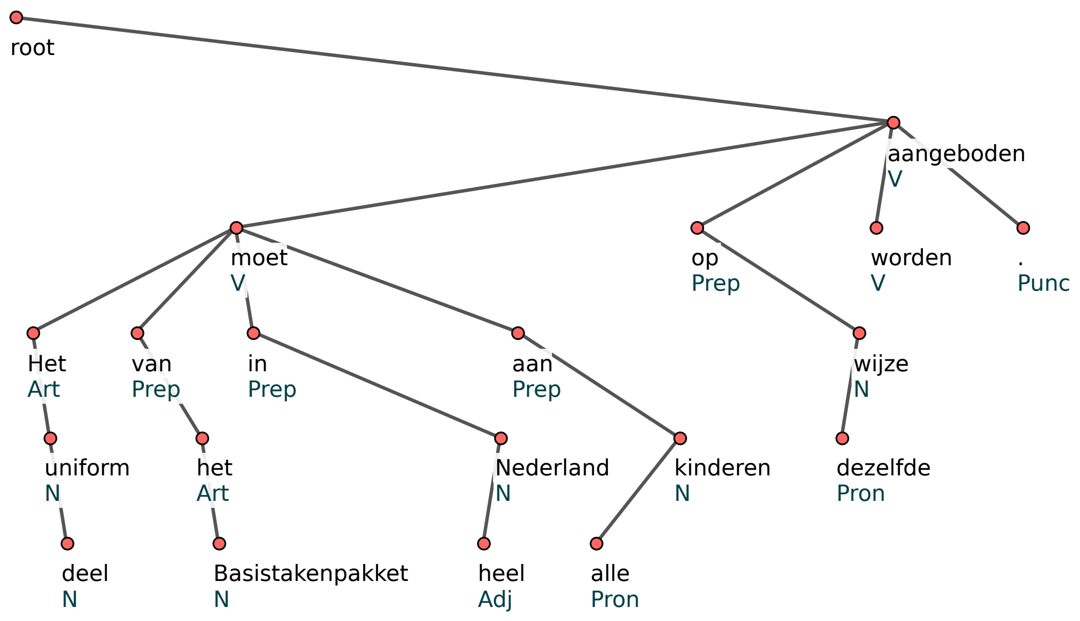


Figure A.12: Example of an induced Dutch dependency tree.

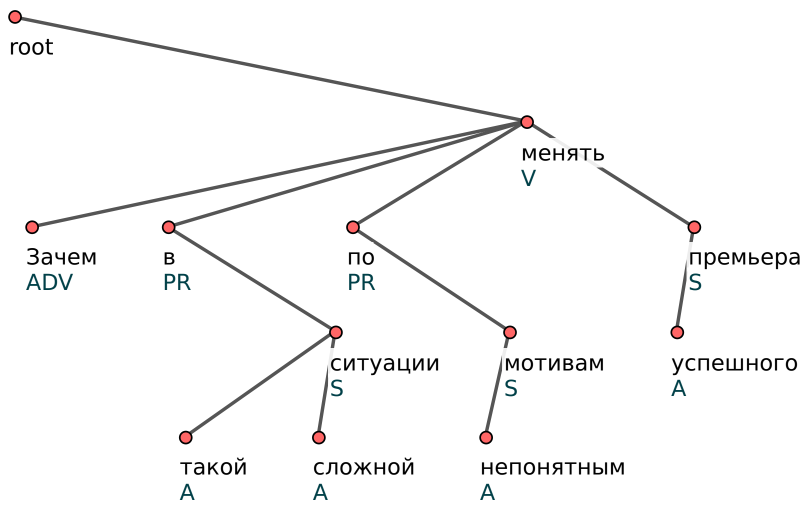


Figure A.13: Example of an induced Russian dependency tree.

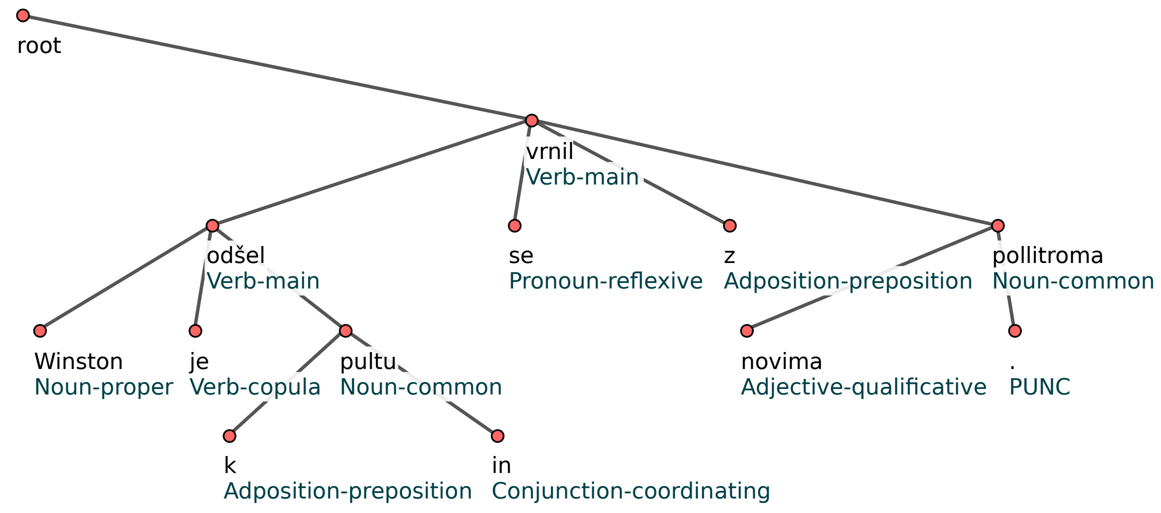


Figure A.14: Example of an induced Slovene dependency tree.

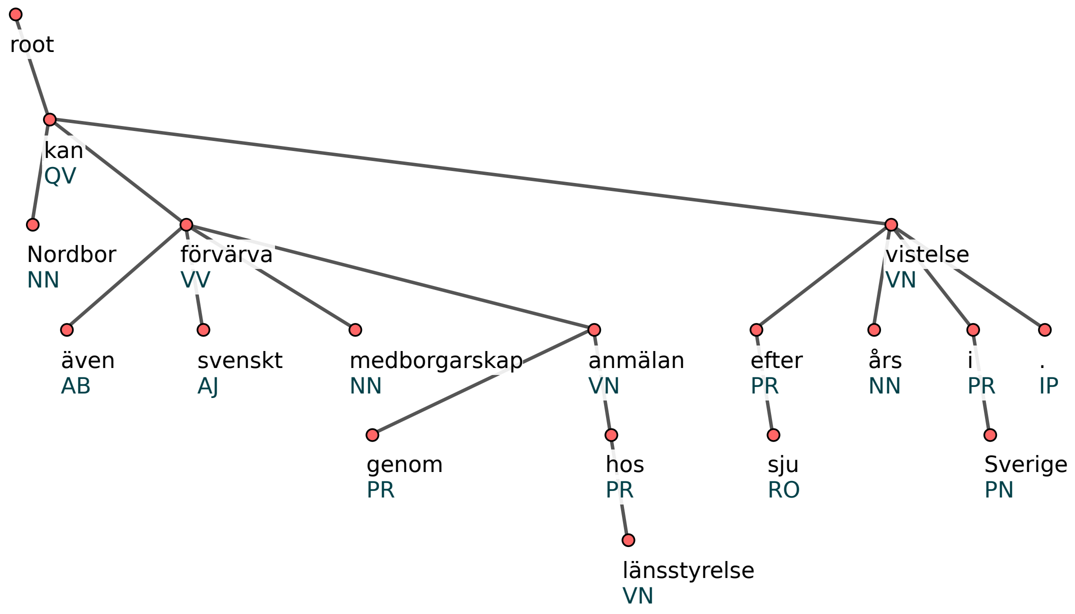


Figure A.15: Example of an induced Swedish dependency tree.

Appendix B

“UDP” – Software Documentation

The software that has been developed for the purpose of this thesis is publicly available and can be downloaded from `http://ufal.mff.cuni.cz/~marecek/udp/`.

Installation

The unsupervised dependency parser “`UDP`” is implemented in Perl and does not need to be installed. However, it requires the Moose module, which can be installed from CPAN using following command:

```
cpan Moose
```

Running `UDP`

The parser can be run either with configuration file or without it using default parameters.

```
./parse.pl --reducibility reducibility_file.red < input_file.conll
    > output_file.conll
```

```
./parse.pl --config config.cfg < input_file.conll > output_file.conll
```

Input and output files are in CoNLL-X format, which is specified e.g. by Buchholz and Marsi (2006). The file `reducibility_file.red` contains reducibility scores for individual PoS n-grams. Such file can be obtained by running the following script:

```
scripts/ngram_reducibility.pl < big_corpus.conll > reducibility.red
```

Precomputed reducibility files for Czech, English and German are also available in the directory `extracted_reducibilities`.

List of Options

- `--reducibility` – File containing reducibility scores in the line-oriented format (see e.g the file `extracted_reducibilities/en.red`).
- `--iterations` – Number of Gibbs sampling iterations (passes through data); default is 20.
- `--tag-column` – PoS tag column used in CoNLL input file, numbered from 0; default is 4.
- `--form-column` – Word form column used in CoNLL input file, numbered from 0; default is 1.
- `--config` – Config file with model parameters (see the file `config.cfg`).

Configuration file

Each model which is used in the parser is specified on one line in the configuration file. An example of such configuration file (particularly the “standard” setting of the parser defined in 7.3) follows:

```
edge alpha=1
distance alpha=1.5
subtree alpha=1 default_score=0.03 score_file=english.red
fertility alpha=0.01 freq_contribution=linear
```

The name of the model is followed by its parameters. Parameter `alpha` is the Dirichlet hyperparameter.

Evaluation

Evaluation script takes gold and predicted CoNLL files and returns scores of the three evaluation metrics: DAS (directed attachment score), UAS (undirected attachment score), and NED (neutral edge direction).

```
scripts/eval.pl gold.conll generated.conll
```