UNSUPERVISED DEPENDENCY PARSING

DAVID MAREČEK, ZDENĚK ŽABOKRTSKÝ

ÚFAL/CKL Technical Report
TR-2011-45
Copies of ÚFAL/CKL Technical Reports can be ordered from:

Institute of Formal and Applied Linguistics (ÚFAL MFF UK)
Faculty of Mathematics and Physics, Charles University
Malostranské nám. 25, CZ-11800 Prague 1
Czech Republic

or can be obtained via the Web: http://ufal.mff.cuni.cz/techrep
Unsupervised Dependency Parsing

David Mareček, Zdeněk Žabokrtský

This technical report summarizes the research on unsupervised dependency parsing at the Institute of Formal and Applied Linguistics, Faculty of Mathematics and Physics, Charles University in Prague, in the year 2011. It describes projective and non-projective approaches of sampling of dependency trees, possibility to employ reducibility feature of dependent words, and reports results obtained across various languages.

This work has been supported by the grants GAČR 201/09/H057, MSM0021620838, and GAUK 116310.
# Contents

1 Introduction ........................................ 5

2 Related Work ........................................ 7

3 Basic Intuitions ..................................... 8
   3.1 Tree structure .................................. 8
   3.2 Projectivity ................................... 8
   3.3 Short dependencies ............................... 10
   3.4 Edge Repeatability .............................. 10
   3.5 Reducibility of dependents ...................... 11

4 Models ................................................. 16
   4.1 Edge model ..................................... 16
   4.2 Distance model .................................. 17
   4.3 Subtree model ................................... 17
   4.4 Overall probability of the treebank ............ 18

5 Sampling algorithms ................................. 19
   5.1 Non-projective tree sampler ...................... 19
      5.1.1 Basic sampling algorithm .................. 19
      5.1.2 Hard Constraints ............................ 19
   5.2 Projective tree sampler ......................... 20
      5.2.1 Initialization ................................ 20
      5.2.2 Small Change Operator ...................... 22
      5.2.3 Building “average” trees ................. 23

6 Experiments ......................................... 24
   6.1 Data ............................................. 24
   6.2 Evaluation metrics ............................... 25
   6.3 Results for non-projective parsing ............... 25
   6.4 Results for projective parsing ................... 27
      6.4.1 Error Analysis ............................... 28
      6.4.2 Ablation Analysis ........................... 29
7 Conclusions
Chapter 1

Introduction

Unsupervised approaches receive considerably growing attention in NLP in the last years, and dependency parsing is not an exception.

The advantages of such approaches are obvious. We do not need any human annotated data for training and therefore we are able to syntactically analyze the texts even in languages, for which there is no formal description of their morphology nor syntax.

Another advantage is more speculative. It is the fact that formal grammars produced by people based on their linguistic intuition may not be adequate 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 may look like from the pure statistical point of view, the only possibility is to employ completely unsupervised parser with no language dependent prior knowledge.

On the other hand, the quality of unsupervised dependency parsers is still much lower than the quality of the supervised approaches, if we compare their results against manually created treebanks. However, such comparison is not very fair since the supervised parsers are trained on a similarly annotated data and therefore it would be quite surprising if the unsupervised methods were doing better here. Instead, the parsers should be compared in an extrinsic way, for example in a final application, such as in machine translation.

Nevertheless, the results in this report are measured intristically, because we have not attempted to engage them in any application and, in addition, it allows us to easily compare our method with other approaches.

The report is structured as follows. Section 2 briefly outlines the state of the art in unsupervised dependency parsing. Section 3 describes the basic intuitions about dependency trees and verify these intuitions on available manually annotated treebanks. Section 4 shows our models which serve for generating probability estimates for edge sampling described in Section 5.
Experimental parsing results across various languages are summarized in Section 6. Section 7 concludes.
Chapter 2

Related Work

The popular approach in unsupervised dependency parsing of the recent years is to employ Dependency Model with Valence (DMV), which was introduced by Klein and Manning (Klein and Manning, 2004). The inference algorithm was further improved by Smith (Smith, 2007) and Cohen et al. (Cohen et al., 2008). (Headden et al., 2009) introduced the Extended Valency Model (EVG) and added lexicalization and smoothing. Blunsom and Cohn (Blunsom and Cohn, 2010) use tree substitution grammars, which allow learning larger dependency fragments.

Unfortunately, many of these works show results only for English.\footnote{The state-of-the-art unsupervised parsers achieve more than 50% of attachment score measured on the Penn Treebank.} However, the main feature of unsupervised methods should be their applicability across a wide range of languages. Such experiments were done by Spitkovsky (Spitkovsky et al., 2011c), where the parsing algorithm was evaluated on all 19 languages included in CoNLL 2006 (Buchholz and Marsi, 2006) and 2007 (Nivre et al., 2007) shared tasks. The fully unsupervised linguistics analysis in (Spitkovsky et al., 2011a) shows that the unsupervised part-of-speech tags may be more useful for this task than the supervised ones.

Brody (Brody, 2010) discovers resemblances between unsupervised parsing and word alignment and introduces the IBM Models 1, 2, and 3 also for dependency trees.

In this paper, we describe a new approach to unsupervised dependency parsing. Unlike the dominating DMV, we will use a combination of three smaller models, and a different inference procedure.
Chapter 3

Basic Intuitions

This chapter describes some basic properties of syntactic structures, which we believe are generally applicable across various natural languages.

3.1 Tree structure

The first such property is the treeness itself. We assume that a syntactic structure of a sentence can be represented by a rooted directed tree. For the formal definition of tree, we will use the following definitions that were taken from (Havelka, 2007).

Definition 1 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 so represents a directed, rooted tree on \(V\). Relation \(\rightarrow^*\) is the reflexive transitive closure of \(\rightarrow\) and is usually called subordination.

Definition 2 A rooted subtree \(S_i\) of a dependency tree \(T = (V, \rightarrow, \preceq)\) is a set of nodes subordinated by \(i \in V\), that is \(S_i = \{v \in V; i \rightarrow^* v\}\).

3.2 Projectivity

Projectivity is one of the important properties of natural languages, even though there are many exceptions, which violate the condition of projectivity. 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.

We will use the definition of tree projectivity introduced by Harper and Hays:

Definition 3 A dependency edge \(i \rightarrow j\) is projective if and only if \(\forall v \in V : v \in (i, j) \implies v \in S_i\).

\(^1\)In surface syntax, each node corresponds to one word in the sentence.
Definition 4 A dependency tree $T = (V, \rightarrow, \preceq)$ is projective if and only if all its edges are projective.

Generally, there are not many non-projective edges in manually annotated treebanks. Havelka (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 even less non-projective edges for other languages. It is important to note that the number of non-projectivities depends not only on the chosen language but also on the chosen annotation guidelines.

![Figure 3.1: Distribution of edge lengths for various languages. They were measured on Czech, English, German and Catalan treebanks included in CoNLL 2006 and 2007 shared tasks.](image)

In this report, we will describe and compare two different algorithms. The first one does not take the tree projectivity into account at all. Conversely, the second one generates strictly projective trees.
3.3 Short dependencies

Naturally, distances between two related words are rather short. Figure 3.1 shows the distributions of lengths of dependencies in four different treebanks. We can see that the probability of a dependency edge between two words decreases rapidly with its length.

![Figure 3.1: Normalized probability mass distribution of edge types for Czech, English, German, and Catalan. All possible edge types (the squared number of POS tags) are ordered according to their frequency and projected to the interval (0,1) on the x-axis. The area under each graph is equal to one. The characteristics were measured using treebanks from CoNLL 2006 and 2007 shared tasks.]

3.4 Edge Repeatability

Assume all possible types of dependency edges, defined as doubles of child and parent part-of-speech (POS) tag. We state that the edge probability mass is concentrated into quite a low number of types and the remaining types are less likely. The measurements on various treebanks (Figure 3.2)
showed the Zipfian distributions.

### 3.5 Reducibility of dependents

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 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 preserves its syntactic correctness. A similar idea of dependency analysis by splitting a sentence into all possible acceptable fragments is used in (Gerdes and Kahane, 2011).

Of course, all the above works had to respond to the notorious fact that there are many language phenomena precluding the ideal (word by word) sentence reducibility (e.g. in the case of prepositional groups, or in the case of subjects in English finite clauses). But 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 be dependent on some other (still present) word.

More generally, if a sequence of words \( < i, j > \) can be removed from a sentence, then this sequence more likely forms a subtree in the dependency tree.

We will compute a reducibility score for each possible sequence of words (n-grams). The obtained scores will be then useful for parsing. The most important are certainly the shortest sequences (i.e. unigrams, bigrams, and possibly trigrams). We faced the two following issues:

1. What size of the context might be taken into account? This is the trade-off between insufficiency and data sparseness.

2. Could be the data sparseness problem solved by word clustering, for example by using part-of-speech tags instead of word forms?

The small context is not sufficient. Consider the two following sentences:

\[
\begin{align*}
\text{Their children} & \text{ went to school}.
\text{I took their children} & \text{ to school.}
\end{align*}
\]

Then the verb ‘went’ is reducible in the context ‘children went to school’, because the sequence ‘children to school’ occurs in the second sentence. There are much more such examples even for larger context mainly for free word-order languages. To prevent this, we decided to take the whole sentences as a context instead of a shorter sequences.
Using the part-of-speech tags instead of word forms also does not bring the proper results. For instance, the two following sentence patterns

\[
\begin{align*}
\text{DT NNS VBD IN DT NN .} \\
\text{DT NNS VBD DT NN .}
\end{align*}
\]

are quite frequent in English. Therefore we could deduce that the preposition \text{IN} can be reduced. But this is a wrong deduction, since the preposition cannot be removed from the prepositional phrase. Based on these observations, we decided to use the full word forms for computing reducibilities.

In the following text, we will use the word \text{n-gram} exclusively for a sequence of part-of-speech tags, not for a sequence of words.

For each possible n-gram, we want to find its score saying how likely this n-gram can be removed from a sentence so that the rest of the sentence remains grammatically correct. This is performed on a large corpus.

\begin{center}
\begin{tabular}{|c|c|c|c|c|}
\hline
unigrams & R & bigrams & R & trigrams & R \\
\hline
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 \\
\hline
\end{tabular}
\end{center}

Table 3.1: Reducibility scores of the most frequent English n-grams. (\text{V*} are verbs, \text{N*} are nouns, \text{DET} are determiners, \text{IN} are prepositions, \text{JJ} are adjectives, \text{RB} are adverbs, \text{CD} are numerals, and \text{CC} are coordinating conjunctions)

Given an n-gram, we go through the corpus\textsuperscript{2} and find all its occurrences. For each such occurrence, we remove the appropriate words from the current sentence and search through the corpus whether the rest of the sentence

\textsuperscript{2}We assume that the corpus is morphologically analyzed by a POS-tagger.
occurs at least once elsewhere in the corpus.\textsuperscript{3} If so, then the n-gram is reducible in the current context, otherwise it is not.

The reducibility $R$ of an n-gram $[t_1 \cdots t_n]$, where $n \in \mathbb{N}$ is the number of words covered by this n-gram, is computed following the Equation (3.1). We define it as the number of times this n-gram was reducible ($r$) divided by all its occurrences in the corpus ($c$). It is then normalized\textsuperscript{4} by the reducibility of all possible n-grams ($G$).

$$R(t_1 \cdots t_n) = \frac{r(t_1 \cdots t_n) + \sigma}{c(t_1 \cdots t_n) + \sigma} \cdot \frac{\sum_{g \in G} r(g)}{\sum_{g \in G} c(g)} \quad (3.1)$$

The parameter $\sigma$ is a smoothing constant ensuring that even the n-grams that could not be reduced anywhere in the corpus get some small score. Moreover, such score is higher for less frequent n-grams.

Tables 3.1, 3.2, and 3.3 show reducibility scores of the most frequent n-grams in English, German, and Czech. If we consider only unigrams, we can see that the scores for verbs are often among the lowest. Verbs are

\begin{table}
\begin{center}
\begin{tabular}{|c|c|c|c|}
\hline
unigrams & bigrams & trigrams & \\
\hline
VV\textsubscript{PP} & 0.00 & NN APPR & 0.00 & NN APPR NN & 0.01 \\
APPR & 0.27 & APPR ART & 0.00 & ADJA NN APPR & 0.01 \\
VV\textsubscript{FIN} & 0.28 & ART ADJA & 0.00 & APPR ART ADJA & 0.01 \\
APPR\textsubscript{ART} & 0.32 & NN VV\textsubscript{PP} & 0.00 & NN KON NN & 0.01 \\
VA\textsubscript{FIN} & 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 \\
NE & 1.14 & NN VV\textsubscript{FIN} & 0.89 & NN VV\textsubscript{PP} $ & 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 \\
ADJ\textsubscript{D} & 3.54 & APPR NN & 2.65 & APPR ADJA NN & 2.65 \\
ADV & 7.69 & APPR\textsubscript{ART} NN & 3.06 & ART NN VV\textsubscript{FIN} & 9.51 \\
\hline
\end{tabular}
\end{center}
\caption{Reducibility scores of the most frequent German n-grams. (\textit{V*} are verbs, \textit{N*} are nouns, \textit{ART} are articles, \textit{APPR*} are prepositions, \textit{ADJ*} are adjectives, \textit{ADV} are adverbs, \textit{CARD} are numerals, and \textit{KON} are conjunctions)}
\end{table}

\textsuperscript{3}We do not take into account sentences that have less than 10 words, because they could be nominal (without any verb) and might influence the reducibility scores of verbs.

\textsuperscript{4}This normalization causes the scores are not too small. Note that the reducibility scores are not probabilities.
Table 3.3: Reducibility scores of the most frequent Czech 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)

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

followed by prepositions and nouns, and the scores for adjectives and adverbs are among the highest for all three examined languages. That is what we want, because the reducible unigrams will more likely become leaves in the 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] whose score is undesirably high.

In Figure 3.3, there is a graph presenting the correlation between unigram reducibility of individual Czech POS tags and how many times these tags were leaves in dependency trees. We can see that the correlation is positive and thus the reducibility feature can be useful.
Figure 3.3: Correlation between unigram reducibility of individual Czech POS tags and how many times they were a leaf in dependency tree. The size of squares corresponds to the POS tag frequencies. The logarithm of reducibility is on the y-axis.
Chapter 4

Models

In this section, we introduce three submodels matching the basic intuitions we have proposed: edge model, distance model and subtree model. All the models are based on part-of-speech tags only; the models dealing with word forms have been left for the future work.

4.1 Edge model

For the purposes of the edge model, we define a dependency edge between the words $w_d$ and $w_g$ as a triple

$$[t_d, t_g, \text{dir}(d, g)],$$

where $d$ is the position of the dependent word, $g$ is the position of the governing word, $t_d$ and $t_g$ are their part-of-speech tags, and $\text{dir}(d, g)$ is the direction in which the word $w_d$ lies related to the word $w_g$. The direction can have two values: left (L) and right (R). For completeness, the part-of-speech tag of the technical root is set to root and the direction in which a word lies form the technical root is set to root as well.\(^1\)

We want to maximize the pointwise mutual information over all edges in our treebank. We add the direction term to the joint probability, so the pointwise mutual information of the edge between the words $w_d$ and $w_g$ is defined as

$$\text{pmi}(d, g) = \log \frac{p(t_d, t_g, \text{dir}(d, g))}{p(t_d)p(t_g)} \quad (4.1)$$

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

\(^1\)All the edges between a word $w_i$ and the technical root have the form $[t_i, \text{root}, \text{root}]$. They are used for modelling ability of a part-of-speech tag to be head of a sentence.
\[ pmi(tree) = \sum_{i=1}^{n} pmi(d_i, g_i) = \log \prod_{i=1}^{n} \frac{p(t_{d_i}, t_{g_i}, \text{dir}(d_i, g_i))}{p(t_{d_i})p(t_{g_i})} \quad (4.2) \]

We can omit the probabilities of the part-of-speech tags of the dependent words, because they are the same for all possible trees.

\[ \arg \max_{tree} pmi(tree) = \arg \max_{tree} \prod_{i=1}^{n} \frac{p(t_{d_i}, t_{g_i}, \text{dir}(d_i, g_i))}{p(t_{g_i})} \quad (4.3) \]

The edge model is based on the Chinese restaurant process. The probability of a dependency edge on the position\(^2\) \(d\) depends on the number of times it occurred before in the corpus.

\[ P_e(d, g) = \frac{c^{-d}("t_{d}, t_{g}, \text{dir}(d, g)")}{c^{-d}(t_{g}^n) + \alpha \cdot 2|T|^2}, \quad (4.4) \]

The edge model is defined in Equation (4.4), where \(c^{-d}\) stands for the count of edges in the history. The count \(c^{-d}("t_{g}^n")\) refers to the number of edges whose parent tag is \(t_{g}\). (Not the number words with the tag \(t_{g}\).) The hyperparameter \(\alpha\) here is the Dirichlet prior.

In some configurations, we use also joined edge model in which the probability of an edge is not conditioned by its parent. Here \(c^{-d}(\ast)\) stands for the number of all edges in history.

\[ P_{je}(d, g) = \frac{c^{-d}("t_{d}, t_{g}, \text{dir}(d, g)") + \alpha}{c^{-d}(\ast) + \alpha \cdot 2|T|^2}, \quad (4.5) \]

### 4.2 Distance model

In the distance model, we define the probability of the edge as the inverse value of the distance between the word and its parent.

\[ P_d(d, g) = \frac{1}{\epsilon \left(\frac{1}{|d - g|}\right)^\beta}, \quad (4.6) \]

where \(\epsilon\) is the normalization constant and hyperparameter \(\beta\) determines the weight of this model.

### 4.3 Subtree model

The subtree model brings the reducibility feature. Let’s define \(\text{desc}(i)\) as the sequence of tags \([t_l \cdots t_r]\) that corresponds to all the descendants of the

\(^2\)We define the position of the edge by the position of its dependent word in the corpus.
word \( w_i \) including \( w_i \), i.e. the whole subtree of \( w_i \). The probability of such subtree is proportional to the reducibility \( R(desc(i)) \). Hyperparameter \( \gamma \) determines the weight of the model.

\[
P_s(i) = \frac{1}{\epsilon} R(desc(i))^\gamma
\] (4.7)

### 4.4 Overall probability of the treebank

The probability of the whole treebank is a product of the probabilities \( P_e \), \( P_d \), and \( P_s \) over all the words in the corpus.

\[
P_{treebank} = \prod_{i=1}^{n} (P_e(i, \pi(i))P_d(i, \pi(i))P_s(i)),
\] (4.8)

where \( \pi(i) \) denotes the parent of the word \( i \).
Chapter 5

Sampling algorithms

For stochastic searching for the most probable dependency trees, we employ Gibbs sampling, a standard Markov Chain Monte Carlo technique (Gilks et al., 1996). We present two different samplers. The first one is generally non-projective, the second one generates strictly projective trees.

5.1 Non-projective tree sampler

The non-projective tree sampling algorithm simply go through all the words in the corpus in random order and choose their parents from all other words in the sentence.

5.1.1 Basic sampling algorithm

The easiest variant of this algorithm does not preserve the tree structure. Its pseudocode is shown in Figure 5.1. It may create cycles and discontinuous directed graphs; such graphs are also accepted as the algorithm’s initial input.

5.1.2 Hard Constraints

The problem of the basic sampling algorithm is that it does not sample trees. It only chooses a parent for each word but does not guarantee the acyclicity. We introduce and explore two hard constraints:

- **Tree** – for each sentence, the set of assigned edges constitutes a tree in all phases of computation,
- **SingleRoot** – the technical root can have only one child.

Tree-sampling algorithm with pseudocode in Figure 5.2 ensures the tree-ness of the sampled structures. It is more complicated, because it checks acyclicity after each edge is sampled. If there is a cycle, it chooses one edge
iterate {
    foreach sentence {
        foreach node in rand_permutation_of_nodes {

            # estimate probability of node's parents
            foreach parent in (0 .. |sentence|) {
                next if parent == node;
                node->set_parent(parent);
                prob[parent] = estimate_edge_prob();
            }

            # choose parent w.r.t. the distribution
            parent = sample from prob[parent];
            node->set_parent(parent);
        }
    }
}

Figure 5.1: Pseudo-code of the basic sampling approach (cycles are allowed).

which will be deleted and the remaining node is then hanged on another
node so that no other cycle is created. This deletion and rehanging is done
using the same sampling method.

The second hard constraint represents the fertility of the technical root,
which is generally supposed to be low. Ideally, each sentence should have
one word which dominates all other words. For this reason, we allow only
one word to depend on the technical root. If the root acquires two children
during sampling, one of them is immediately resampled (a new parent is
sampled for the child).

5.2 Projective tree sampler

The algorithm for projective sampling is completely different, since the
projectivity constraint is hard to employ in the previously described non-
projective algorithm.

5.2.1 Initialization

Before the sampling starts, we initialize the projective trees randomly. For
doing so, we tried the following two initializers:

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

- We are picking one word after another in a random order and we attach
  it to the nearest left (or right) neighbor that has not been attached
  yet. The left-right choice is made by a coin flip. If it is not possible
iterate {
  foreach sentence {
    foreach node in rand_permutation_of_nodes {

      # estimate probability of node's parents
      foreach parent in (0 .. |sentence|) {
        next if parent == node;
        node->set_parent(parent);
        prob[parent] = estimate_edge_prob();
      }

      # choose parent w.r.t. the distribution
      parent = sample from prob[parent];
      node->set_parent(parent);

      if (cycle was created) {

        # choose where to break the cycle
        foreach node2 in cycle {
          parent = node2->parent;
          node2->unset_parent();
          prob[node2] = estimate_edge_prob();
          node2->set_parent(parent);
        }
        node2 = sample from prob[node2];

        # choose the new parent
        foreach parent {
          next if node2->parent creates a cycle
          node2->set_parent(parent);
          prob[parent] = estimate_edge_prob();
        }
        parent = sample from prob[parent];
        node2->set_parent(parent);
      }
    }
  }
}

Figure 5.2: Pseudo-code of the tree-sampling approach (cycles are not allowed).

to attach a word to one side, we attach it to the other side. The last unattached word is then the head of the sentence.

While the first method generates only flat trees, the second one can generate all possible projective trees. However, the sampler converges to similar results for both the initializations. Therefore we conclude that the choice of the initialization mechanism is not so important here.
5.2.2 Small Change Operator

We use the bracketing notation for illustrating 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 segment head. An example of this notation is in Figure 5.3.

\[
((\text{The dog}) \text{ was } \text{in} \ ((\text{the}) \text{ park}) \ (.))
\]

Figure 5.3: Arrow and bracketing notation of a projective dependency tree.

The small change is then very simple. We remove one pair of brackets and add another, so that the conditions defined above are not violated. The example of such change is in Figure 5.4.

\[
(((\text{The}) \text{ dog}) \text{ was } \text{in} ((\text{the}) \text{ park}) \ (.))
\]

Figure 5.4: An example of small change in a projective tree. The bracket (in the park) was removed and there are three possibilities how to replace it.

From the perspective of the dependencies, the small change is following:

1. Pick a random non-root word \( w \) (the word \text{ in} in our example) and find its parent \( p \) (the word \text{ was}).
2. Find all other children of \( w \) and \( p \) (the words \text{ dog}, \text{ park}, and .) and denote this set as \( C \).
3. Choose the new head from \( w \) and \( p \). Mark the new head as \( g \) and the second candidate as \( d \). Attach \( d \) to \( g \).
4. Select the neighborhood \( D \) of the word \( d \) as a continuous subset of \( C \) and attach all words from \( D \) to \( d \).
5. Attach the remaining words from \( C \) that were not in \( D \) to the new head \( g \).
5.2.3 Building “average” trees

The “burn-in” period is set to 10 iterations. After this period, we begin to count how many times an edge occurs at a particular location in the corpus. This counts are updated over the whole corpus with the probability 0.01 after each small change is made.

When the sampling is finished, the final dependency trees are built using such edges that were the most frequent during the sampling. We employed the maximum spanning tree (MST) algorithm (Chu and Liu, 1965) to find them.\(^1\) Tree projectivity is not guaranteed by the MST algorithm.

---

\(^1\)The weights of edges needed in MST algorithm correspond to the number of times they were present during the sampling.
Chapter 6

Experiments

6.1 Data

We need two kinds of data for our experiments: a smaller treebank, which is used for sampling and for evaluation, and a large corpus, from which n-gram reducibility scores are computed.

The treebanks were taken from the CoNLL shared task 2006 and 2007 (Buchholz and Marsi, 2006; Nivre et al., 2007). The Czech treebank is a subset of Prague Dependency Treebank (Hajič et al., 2006), the German treebank was derived from the Tiger treebank (Brants and Hansen, 2002), and the English treebank comes from the Penn Treebank (Marcus et al., 1994), where the constituents were converted to dependencies by Pennconverer (Johansson and Nugues, 2007). We use only the testing parts of the treebanks\(^1\) and as a source of the part-of-speech tags, we used the \texttt{POS} column, which is the fifth column in the CoNLL format. The CoNLL tagset size and data statistics for each language are shown in Table 6.1. In some experiments that do not require large corpus for computing n-gram reducibilities, we do the evaluation on all the 19 languages included in CoNLL data.

<table>
<thead>
<tr>
<th>language</th>
<th>CoNLL</th>
<th>sentences</th>
<th>tokens</th>
<th>tagset size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Czech</td>
<td>2007</td>
<td>364</td>
<td>5760</td>
<td>59</td>
</tr>
<tr>
<td>German</td>
<td>2006</td>
<td>357</td>
<td>5694</td>
<td>54</td>
</tr>
<tr>
<td>English</td>
<td>2007</td>
<td>377</td>
<td>9529</td>
<td>45</td>
</tr>
</tbody>
</table>

Table 6.1: CoNLL testing data statistics. Note that the Czech POS tags were shortened in CoNLL (compared to the original treebank), and thus the tagset size is only 59.

For obtaining reducibility scores, we downloaded the texts from Czech, German, and English Wikipedia articles. Their statistics are showed in Table 6.2. To make them useful, the necessary preprocessing steps must

\(^1\)The file \texttt{test.conll} for the year 2006 and the file \texttt{dtest.conll} for the year 2007.
have been done. After the rule-based segmentation and tokenization\textsuperscript{2}, the texts were automatically POS tagged\textsuperscript{3} using the pretrained models.

<table>
<thead>
<tr>
<th>language</th>
<th>sentences</th>
<th>tokens</th>
</tr>
</thead>
<tbody>
<tr>
<td>Czech</td>
<td>998,000</td>
<td>19.1 millions</td>
</tr>
<tr>
<td>German</td>
<td>935,000</td>
<td>18.9 millions</td>
</tr>
<tr>
<td>English</td>
<td>3,149,000</td>
<td>80.9 millions</td>
</tr>
</tbody>
</table>

Table 6.2: Wikipedia texts statistics

6.2 Evaluation metrics

As in other unsupervised tasks (e.g. in unsupervised POS induction), there is a little consensus on evaluation measures. Performance of unsupervised methods is often measured by comparing the induced outputs with gold standard manual annotations. However, this approach causes a general problem: manual annotation is inevitably guided by a number of conventions, such as the traditional POS categories in unsupervised POS tagging, or varying (often linguistically controversial) conventions for local tree shapes representing e.g. complex verb forms in unsupervised dependency parsing. It is obvious that using unlabeled attachment scores (UAS) leads to a strong bias towards such conventions and it might not be a good indicator of unsupervised parsing improvements. Therefore we estimate parsing quality by two additional metrics:

- UUAS - undirected UAS (edge direction is disregarded),
- NED - neutral edge direction, introduced in (Schwartz et al., 2011), which treats not only a node’s gold parent and child as the correct answer, but also its gold grandparent.

6.3 Results for non-projective parsing

In the non-projective parsing algorithm, we employed only \textit{joined edge model} and \textit{distance model}\textsuperscript{4} The hyperparameters were set as follows:

\textsuperscript{2}The segmentation to sentences and tokenization was performed using the TectoMT framework (Popel and Zabokrtský, 2010)

\textsuperscript{3}We used Morče tagger (Spoustová et al., 2007) for English and Czech, and TreeTagger (Schmid, 1995) for German. The tagsets of the pretrained models differs only in small details from the tagset used in CoNLL data. The differences were removed.

\textsuperscript{4}The \textit{subtree model} has not been employed in non-projective algorithm, because the projections of subtrees may contain gaps and reducibility scores can be computed only on continuous sequences of words so far.

25
\[\alpha = 0.01, \quad \beta = 2\]

We applied our unsupervised dependency parser on all languages included in 2006 and 2007 CoNLL shared tasks. We used the configuration that was the best for Czech. The parsing was run on concatenated training and development sets after removing punctuation, but the final attachment scores were measured on the development sets only, so that they were comparable to the previously reported results. There is no sentence length limit and the evaluation is done for all the sentences and only the POS (fifth column in the CoNLL format) is used for the inference.

<table>
<thead>
<tr>
<th>Language</th>
<th>Baselines</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>code</td>
<td>CoNLL</td>
</tr>
<tr>
<td>Arabic</td>
<td>ar</td>
<td>2007</td>
</tr>
<tr>
<td>Bulgarian</td>
<td>bg</td>
<td>2006</td>
</tr>
<tr>
<td>Catalaun</td>
<td>ca</td>
<td>2007</td>
</tr>
<tr>
<td>Czech</td>
<td>cs</td>
<td>2007</td>
</tr>
<tr>
<td>Danish</td>
<td>da</td>
<td>2006</td>
</tr>
<tr>
<td>German</td>
<td>de</td>
<td>2006</td>
</tr>
<tr>
<td>Greek</td>
<td>el</td>
<td>2007</td>
</tr>
<tr>
<td>English</td>
<td>en</td>
<td>2007</td>
</tr>
<tr>
<td>Spanish</td>
<td>es</td>
<td>2006</td>
</tr>
<tr>
<td>Basque</td>
<td>eu</td>
<td>2007</td>
</tr>
<tr>
<td>Hungarian</td>
<td>hu</td>
<td>2007</td>
</tr>
<tr>
<td>Italian</td>
<td>it</td>
<td>2007</td>
</tr>
<tr>
<td>Japanese</td>
<td>ja</td>
<td>2006</td>
</tr>
<tr>
<td>Dutch</td>
<td>nl</td>
<td>2006</td>
</tr>
<tr>
<td>Portuguese</td>
<td>pt</td>
<td>2006</td>
</tr>
<tr>
<td>Slovenian</td>
<td>sl</td>
<td>2006</td>
</tr>
<tr>
<td>Swedish</td>
<td>sv</td>
<td>2006</td>
</tr>
<tr>
<td>Turkish</td>
<td>tr</td>
<td>2006</td>
</tr>
<tr>
<td>Chinese</td>
<td>zh</td>
<td>2007</td>
</tr>
</tbody>
</table>

| Average   | 7.2       | 26.4    | 29.8  | 35.1  | 36.7  | 39.3 |

Table 6.3: Directed unlabeled attachment scores for 19 different languages from CoNLL shared task. The “rand.”, “left”, and “right” columns reports Random, LeftChain, and RightChain baselines. The “Our-NR” and “Our” columns show results of our algorithm; “NR” means that Noun-Root dependency suppression was used. For comparison, “Spi5” and “Spi6” are the results reported in (Spitkovsky et al., 2011c) in Tables 5 and 6 respectively.

\(^5\text{train.conll and test.conll files for CoNLL2006 languages and dtrain.conll and dtest.conll for CoNLL2007 languages.}\)
The results are shown in Table 6.3. The *Random, Left Chain, and Right Chain* baselines are compared to our results and to the results that were reported by (Spitkovsky et al., 2011c). The scores are higher for 6 (7) languages compared to “Spi5” (“Spi6”), the averaged attachment score is lower.

Interestingly, Arabic, Danish, and Japanese have very high *LeftChain (RightChain)* baseline and no method was able to beat them so far.

### 6.4 Results for projective parsing

In the projective parsing algorithm, we employ all the three submodels *edge model and distance model, and subtree model*. The respective hyperparameters $\alpha, \beta,$ and $\gamma$, which determine the weights of the individual submodels, were set manually. After a couple of experiments, we end up with the following values, which give relatively good results for all three languages.

$$\alpha = 1, \quad \beta = 2, \quad \gamma = 3$$

The smoothing constant for reducibility scores from Equation (3.1) was set to 0.01. Changing this value in reasonable limits does not affect the results.

The evaluation is performed on the same data as the sampling. The attachment scores are computed on all sentences in the testing data. In Table 6.4, we show the results of our parser using the three different metrics:

- **Unlabeled attachment score (UAS)** – the standard metric for dependency parsing evaluation,
- **Undirected unlabeled attachment score (UUAS)** – edge direction is disregarded,
- **NED** – neutral edge direction, which was introduced by Schwartz (Schwartz et al., 2011). It treats not only a node’s gold parent and child as the correct answer, but also its gold grandparent, which neutralizes the effect of edge inversion.

In Table 6.5, the results of our parser are compared with the results previously reported by Spitkovsky (Spitkovsky et al., 2011c). In this papers, the attachment scores are reported excluding the punctuation. The comparison of the results is quite hard, since the scores across languages and settings of the parsers varies greatly. Moreover, the comparison is not fair,

---

6In some papers about unsupervised parsing, only short sentences are selected for evaluation and the scores are therefore much higher.

7All punctuation nodes are removed from the trees. If a removed punctuation node is not a leaf, its children are attached below the parent of the removed node.
Table 6.4: The quality of our parser measured by three different metrics: unlabeled attachment score (UAS), its undirected variant (UUAS), and neutral direction (NED). Punctuation marks were included in this evaluation.

<table>
<thead>
<tr>
<th>language</th>
<th>UAS [%]</th>
<th>UUAS [%]</th>
<th>NED [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Czech</td>
<td>42.6</td>
<td>50.0</td>
<td>62.6</td>
</tr>
<tr>
<td>English</td>
<td>39.5</td>
<td>47.2</td>
<td>62.7</td>
</tr>
<tr>
<td>German</td>
<td>28.7</td>
<td>41.5</td>
<td>51.7</td>
</tr>
</tbody>
</table>

Table 6.5: Unlabeled attachment scores (UAS) compared to the latest reported results on the same datasets. ‘Spitkovsky1’ results are copied from the work (Spitkovsky et al., 2011b), ‘Spitkovsky2’ results come from the later work (Spitkovsky et al., 2011c). Here, the punctuation is excluded form evaluation.

<table>
<thead>
<tr>
<th>parser</th>
<th>our [%]</th>
<th>Spitkovsky1 [%]</th>
<th>Spitkovsky2 [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Czech</td>
<td>47.6</td>
<td>37.8</td>
<td>31.4</td>
</tr>
<tr>
<td>English</td>
<td>41.5</td>
<td>50.3</td>
<td>34.9</td>
</tr>
<tr>
<td>German</td>
<td>31.8</td>
<td>28.6</td>
<td>33.5</td>
</tr>
</tbody>
</table>

since the different sources were used. We get the reducibility scores from the larger corpus. On the other hand, we do not use word forms in parsing and in (Spitkovsky et al., 2011b), there was used an information about punctuation marks. However, we can say that our parser outperforms the others for Czech. For English and German, it is in both cases once worse and once better than in the previously reported results.

6.4.1 Error Analysis

After inspecting the resulting dependency trees, we have found the following obvious errors:

- Noun phrases – The phrases that consists of more nouns were badly structured. This was caused probably by ignoring word forms. For example, the structure of the sequence ‘*NN NN NN*’ can be hardly recognized by our parser.

- Grammatical words – In some cases, there were mistakes in attachment of the grammatical (function) words. The most noticeable were the German articles whose positions in the tree were switched with the appropriate nouns. This caused the very poor score for German. The reason of these article-noun switches may come from the reducibility scores. The reducibility of the German bigram *NN ART* is unfortunately quite high and the reducibilities of *ART* and *NN* are too close.
Klein and Manning (Klein and Manning, 2004) observed the similar behavior in their experiments with DMV.

- Full stops – Full stops are often attached to the last noun in the sentence, which is often wrong. That is why the attachment scores are higher after removing punctuation.

6.4.2 Ablation Analysis

To investigate the impact of individual components of the model, we run the parser for all possible component combinations. The attachment scores are shown in Table 6.6. The subtree model, which utilizes the newly introduced reducibility scores of n-grams, has obviously the highest impact.

<table>
<thead>
<tr>
<th>lang.</th>
<th>-</th>
<th>e</th>
<th>d</th>
<th>s</th>
<th>ed</th>
<th>es</th>
<th>ds</th>
<th>eds</th>
</tr>
</thead>
<tbody>
<tr>
<td>Czech</td>
<td>23.0</td>
<td>27.4</td>
<td>25.3</td>
<td>35.6</td>
<td>24.4</td>
<td>43.8</td>
<td>39.5</td>
<td>47.6</td>
</tr>
<tr>
<td>German</td>
<td>18.7</td>
<td>22.1</td>
<td>21.7</td>
<td>25.4</td>
<td>22.1</td>
<td>30.7</td>
<td>27.8</td>
<td>31.8</td>
</tr>
<tr>
<td>English</td>
<td>20.4</td>
<td>15.5</td>
<td>25.3</td>
<td>29.3</td>
<td>28.4</td>
<td>27.3</td>
<td>33.0</td>
<td>41.5</td>
</tr>
</tbody>
</table>

Table 6.6: Ablation analysis. Unlabeled attachment scores for the different combinations of model components. The letters e, d, and s stay for the presence of edge, distance, and subtree model respectively. The hyphen shows the baseline scores, that is randomly generated dependency trees, when no model is used. Here, the punctuation was excluded from evaluation.
Chapter 7

Conclusions

This report described two different algorithms for unsupervised dependency parsing based on Gibbs sampling.

The projective algorithm utilizes the reducibility feature, which prove to be very useful in unsupervised dependency parsing task. We extract the n-gram reducibility scores from a large corpus, and then make the computationally demanding inference on smaller data using only these scores. The best results were obtained on Czech. We explain it by the fact that there are less grammatical (function) words in Czech, which are sometimes problematic for obtaining reducibility.

The non-projective algorithm does not utilize the reducibility feature, even though we believe it would help as well. We would like to adapt the reducibility feature also for non-projective “gappy” structures in future work. However, for several languages (e.g. Spanish, Italian, Portuguese) this algorithm appeared to have even better results than previously published best results.


ÚFAL
ÚFAL (Ústav formální a aplikované lingvistiky; http://ufal.mff.cuni.cz) is the Institute of Formal and Applied linguistics, at the Faculty of Mathematics and Physics of Charles University, Prague, Czech Republic. The Institute was established in 1990 after the political changes as a continuation of the research work and teaching carried out by the former Laboratory of Algebraic Linguistics since the early 60s at the Faculty of Philosophy and later the Faculty of Mathematics and Physics. Together with the “sister” Institute of Theoretical and Computational Linguistics (Faculty of Arts) we aim at the development of teaching programs and research in the domain of theoretical and computational linguistics at the respective Faculties, collaborating closely with other departments such as the Institute of the Czech National Corpus at the Faculty of Philosophy and the Department of Computer Science at the Faculty of Mathematics and Physics.

CKL
As of 1 June 2000 the Center for Computational Linguistics (Centrum komputační lingvistiky; http://ckl.mff.cuni.cz) was established as one of the centers of excellence within the governmental program for support of research in the Czech Republic. The center is attached to the Faculty of Mathematics and Physics of Charles University in Prague.

Technical Reports
The ÚFAL/CKL technical report series has been established with the aim of disseminate topical results of research currently pursued by members, cooperators, or visitors of the Institute. The technical reports published in this Series are results of the research carried out in the research projects supported by the Grant Agency of the Czech Republic, GAČR 405/96/K214 (“Komplexní program”), GAČR 405/96/0198 (Treebank project), grant of the Ministry of Education of the Czech Republic VS 96151, and project of the Ministry of Education of the Czech Republic LN00A063 (Center for Computational Linguistics). Since November 1996, the following reports have been published.

ÚFAL TR-1996-01 Eva Hajičová, The Past and Present of Computational Linguistics at Charles University
Jan Hajič and Barbora Hladká, Probabilistic and Rule-Based Tagging of an Inflective Language – A Comparison

ÚFAL TR-1997-02 Vladislav Kuboň, Tomáš Holan and Martin Plátek, A Grammar-Checker for Czech

ÚFAL TR-1997-03 Alla Bémová at al., Anotace na analytické rovině, Návod pro anotátory (in Czech)

ÚFAL TR-1997-04 Jan Hajič and Barbora Hladká, Tagging Inflective Languages: Prediction of Morphological Categories for a Rich, Structural Tagset

ÚFAL TR-1998-05 Geert-Jan M. Kruijff, Basic Dependency-Based Logical Grammar

ÚFAL TR-1999-06 Vladislav Kuboň, A Robust Parser for Czech

ÚFAL TR-1999-07 Eva Hajičová, Jarmila Panevová and Petr Sgall, Manuál pro tektogramatické značkování (in Czech)

ÚFAL TR-2000-08 Tomáš Holan, Vladislav Kuboň, Karel Oliva, Martin Plátek, On Complexity of Word Order

ÚFAL/CKL TR-2000-09 Eva Hajičová, Jarmila Panevová and Petr Sgall, A Manual for Tectogrammatical Tagging of the Prague Dependency Treebank

ÚFAL/CKL TR-2001-10 Zdeněk Žabokrtský, Automatic Functor Assignment in the Prague Dependency Treebank

ÚFAL/CKL TR-2001-11 Markéta Straňáková, Homonymie předložkových skupin v češtině a možnost jejich automatického zpracování

ÚFAL/CKL TR-2001-12 Eva Hajičová, Jarmila Panevová and Petr Sgall, Manuál pro tektogramatické značkování (III. verze)
ÚFAL/CKL TR-2008-38 Marie Mikulová, Rekonstrukce standardizovaného textu z mluvené řeči v Pražském závislostním korpusu mluvené češtiny. Manuál pro anotátory

ÚFAL/CKL TR-2008-39 Zdeněk Žabokrtský, Ondřej Bojar, TectoMT, Developer's Guide

ÚFAL/CKL TR-2008-40 Lucie Mladová, Diskurzní vztahy v češtině a jejich zachycení v Pražském závislostním korpusu 2.0

ÚFAL/CKL TR-2009-41 Marie Mikulová, Pokyny k překladu určené překladatelům, revizorům a korektorům textů z Wall Street Journal pro projekt PCEDT

ÚFAL/CKL TR-2011-42 Loganathan Ramasamy, Zdeněk Žabokrtský, Tamil Dependency Treebank (TamilTB) - 0.1 Annotation Manual

ÚFAL/CKL TR-2011-43 Nguy Giang Linh, Michal Novák, Anna Nedoluzhko, Coreference Resolution in the Prague Dependency Treebank

ÚFAL/CKL TR-2011-44 Anna Nedoluzhko, Jiří Mírovský, Annotating Extended Textual Coreference and Bridging Relations in the Prague Dependency Treebank

ÚFAL/CKL TR-2011-45 David Mareček, Zdeněk Žabokrtský, Unsupervised Dependency Parsing