Abstract

This paper presents a work in progress on the task of unsupervised parsing, following the main stream approach of optimizing the overall probability of the corpus. We evaluate a sequence of experiments for Czech with various modifications of corpus initiation, of dependency edge probability model and of sampling procedure, stressing especially the treeness constraint. The best configuration is then applied to 19 languages from CoNLL-2006 and CoNLL-2007 shared tasks. Our best achieved results are comparable to the state of the art in dependency parsing and outperform the previously published results for many languages.

1 Introduction

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

In recent years, quite a lot of works in unsupervised parsing (or grammar induction) was based on Dependency Model with Valence (DMV) introduced by (Klein and Manning, 2004); (Smith, 2007) and (Cohen et al., 2008) has focused on DMV variants, (Headden et al., 2009) introduced extended valency model (EVG) and added lexicalization and smoothing. (Spitkovsky et al., 2011b) used punctuation marks for splitting a sentence and impose parsing restrictions over its fragments. Gibbs sampling was used in (Naseem and Barzilay, 2011).

Some of the papers focused on English only, but some presented the results across wide rage of languages. The last such paper was (Spitkovsky et al., 2011a), where the evaluation was done on all 19 languages included in CoNLL shared tasks (Buchholz and Marsi, 2006) and (Nivre et al., 2007). The attachment scores are very high for English, for which the methods seems to be optimized, but the scores are quite low for some other languages.

In this paper, we describe our new approach to unsupervised dependency parsing. Unlike DMV, it is not based on constituency trees, which cannot handle non-projectivities. We have been inspired rather by the experiment described in (Brody, 2010), in which the dependency parsing task is formulated as a problem of word alignment; every sentence is aligned with itself with one constraint: no word can be attached to itself. However, unlike (Brody, 2010), where the output structures might not be trees and could contain cycles, we introduce a sampling method with the acyclicity constraint.

Our approach attempts at optimizing the overall probability of tree structures given the corpus. We perform the optimization using Gibbs sampling (Gilks et al., 1996).

We employ several ways of incorporating prior knowledge about dependency trees into the system:

- independence assumptions – we approximate probability of a tree by a product of probabilities of dependency edges,
- edge models and feature selection – we use words’ distance and their POS tags as the main indicators for predicting a dependency relation,
- hard constraints – some knowledge on dependency tree properties (such as acyclicity) is difficult to represent by local models, therefore we implement it as a hard constraint in the sampling procedure,
- corpus initialization – we study the effect of different initializations of trees in the corpus,
• basic linguistic assumptions – according to
the dependency syntax tradition, we expect
the trees to be verbocentric. This is done
without determining which part-of-speech
tag is what.

All experiments are evaluated in detail using Czech data. The configuration which per-
forms best for Czech is applied also on other lan-
guages available in the CoNLL shared task cor-
pora (Buchholz and Marsi, 2006) and (Nivre et al.,
2007). Our goal is to achieve good results across
various languages without tuning the parser indi-
vidually for each language, so we use the other
language data exclusively for evaluation purposes.

2 Data preparation

We used Czech training part (dtrain.conll) from
CoNLL 2007 collection, which corresponds to
approximately one third of Prague Dependency
Treebank 2.0 (Hajič and others, 2006), PDT in
the sequel. We selected all sentences containing at
most 15 words after removing punctuation.1 The
resulting data contains 123,804 words in 14,766
sentences (out of 368,640 words and 25,360 sen-
tences in the original dtrain.conll file).

We are aware of a strong bias caused by this
filtering. For instance, it leads to a consider-
ably higher proportion of sentences without a verb
(such as titles – recall that PDT contains mainly
newspaper articles). However, such filtering is a
usual practise in unsupervised parsing due to time
complexity issues.

Since Czech is a morphologically rich language,
there are around 3,000 morphological tags distin-
guished in PDT. They consist of 15 positions, each
of them corresponding to one morphological cate-
gory. In the CoNLL format, Czech positional tags
are distributed into three columns: CPOS (first po-

tition), POS (second position), and FEATURES
(third to fifteenth position). For the purpose of un-
supervised parsing experiments, we reduce the tag
set at two levels of granularity:

• CoarsePOS – only the first letter is consid-
ered for each tag (11 distinct values, such as
V for verbs and P for pronouns),

• FinePOS – only the first (coarse-grained
POS) and the fifth letter (morphological case)
is used if case is defined (such as N4 for
nouns in accusative), or the first and the sec-
ond letter otherwise (such as Vf for infinitive
verb forms); there are 58 distinct values.2

We use this data in all our tuning experiments
(Sections 6.2 – 6.5). The final evaluation on
CoNLL (Section 6.6) is different. It is made on
all the sentences (without length limit) and only
CoNLL POS tags are used there.

3 Models

Similarly to (Brody, 2010), we use two models
which are very close to IBM Model 1 and 2 for
word alignment (Brown et al., 1993). We do not
model fertility (IBM Model 3), but we plan to
involve it in future work. We introduce another
model (called NounRoot) that postulates verbo-
centricity of the dependency trees and tries to re-
press Noun-Root dependencies.

3.1 Standard Dependency Models

In our models, each possible dependency edge is
characterized by three attributes:

• $T^g$ – tag of the governing node,

• $T^d$ – tag of the dependent node,

• $P^d,g$ – signed distance between governing
and dependent word (it is negative, if the de-
pendent word precedes the governing one,
and is equal to 0 if the governing node is the
technical root).

The first model (called Dep) postulates that the
tag of the governing node depends only on the tag
of the dependent node. The probability that the
node $d$ is attached below the node $g$ is:

$$ P(d \rightarrow g) = P(T^g|T^d) = \frac{P(T^g, T^d)}{P(T^d)} \quad (1) $$

We assume that the dependencies follow a Chi-
inese Restaurant Process (Aldous, 1985), in which
the probability $P(T^g|T^d)$ is proportional to the
number of times $T^g$ have governed $T^d$ in the past,
as follows:

$$ P(T^g|T^d_i) = \frac{\text{count}(-1)(T^g_i, T^d_i) + \alpha_1}{\text{count}(-1)(T^d_i) + \alpha_1|T|}, \quad (2) $$

1If a removed punctuation node was not a leaf, its children
were attached below the removed node’s parent. This occurs
mainly with coordinations without conjunctions.

2This shape of tags has been previously shown to perform
well for supervised parsing.
where the index $i$ corresponds to the position of the dependent word in the corpus, \( \text{count}^{(-i)} \) represents number of occurrences in the history (from 1 to $i - 1$), \(|T|\) is the number of tags in the tag set and $\alpha_1$ is the Dirichlet hyper-parameter.

The second model (called Dist) assumes that the length of the dependency edge depends on the tag of the dependent node:

\[
P(d \rightarrow g) = P(D^{d,g} | T^{d}) = \frac{P(D^{d,g}, T^{d})}{P(T^{d})} \tag{3}
\]

\[
P(D_i | T_i^{d}) = \frac{\text{count}^{(-i)}(D_i, T_i^{d}) + \alpha_2}{\text{count}^{(-i)}(T_i^{d}) + \alpha_2 |D|} \tag{4}
\]

where \(|D|\) is the number of all possible distances in the corpus. This number was set to 30.

The probability of a particular analysis (i.e., the probability of all dependency trees $T$ built on a whole given corpus $\mathcal{C}$) can be computed as:

\[
P(\mathcal{C}, T) = \prod_{i=1}^{N} P(T_i^{d} ; D_i | T_i^{d}) \tag{5}
\]

We maximize this probability using Gibbs sampling (Gilks et al., 1996).

### 3.2 Noun-Root Dependency Repression

During our first experiments, we noticed that nouns (especially subjects) often substitute verbs in the governing positions. Since majority of grammars are verbocentric (verbs dominates their subjects and objects), we decided to penalize noun-root edges. Of course, we do not want to state explicitly which tag represents nouns in a particular tag set. Instead, nouns are recognized automatically as the most frequent coarse-grained tag category in the corpus (this simple rule holds for all languages in the CoNLL 2006 and 2007 sets).\(^3\) We add the following model called Noun-Root:

\[
P(d \rightarrow g) = \begin{cases} 
\beta & \text{if } d \text{ is noun and } g \text{ is root} \\
1 & \text{otherwise}
\end{cases} \tag{6}
\]

\(^3\)We are aware that introducing this rule is a kind of hack, which departs from the line of purely unsupervised parsing and which will become useless with automatically induced POS tags in future experiments. On the other hand, this simple trick has a substantial effect on parsing quality. Therefore we decided to present results both with and without using it.

This model is added into the product in Equation (5). The value of $\beta$ was experimentally set to 0.01.

### 4 Sampling

We sample from the posterior distribution of our model $P(T^g, D^{g,d}|T^{d})$ using Gibbs sampling (a standard Markov chain Monte Carlo technique). We sample each dependency edge independently. Computing the conditional probabilities is straightforward, because the numerators and denominators in the product in Equation (5) are exchangeable. If we substitute the parent of a word by a new parent, we can deal with the dependency as if it were the last one in the corpus. The history remains unchanged and updating the probability is thus very efficient.

#### 4.1 Basic sampling algorithm

The pseudocode of the basic sampling algorithm is shown in Figure 1. This algorithm chooses one parent for each word. It may create cycles and discontinuous directed graphs; such graphs are also accepted as the algorithm’s initial input.

```
iterate {
    foreach sentence {
        foreach node in rand_permutation_of_nodes {
            # estimate probability of node’s parents
            foreach parent in (0 .. |sentence|) {
                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 1: Pseudo-code of the basic sampling approach (cycles are allowed).

#### 4.2 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,\(^4\)

\(^4\)This constraint is not compliant with the RandInit initialization.
• SingleRoot – the technical root can have only one child.

Tree-sampling algorithm with pseudocode in Figure 2 ensures the treeness of the sampled structures. It is more complicated, because it checks acyclicity after each sampled edge. If there is a cycle, it chooses one edge 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.

```
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 {
          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 2: Pseudo-code of the tree-sampling approach (cycles are not allowed).

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 Experimental Setup

This section describes the ways of initialization, and how the final dependency trees are built from sampling.

5.1 Corpus Initialization

We implemented four different procedures for initiating dependency edges in the corpus:

• RandInit – each word is attached below a randomly chosen word from the same sentence (or the technical root); treeness is not ensured,

• RandTreeInit – like RandInit, but treeness is ensured (only edges not leading to a cycle are added),

• LeftChainInit – in each sentence, each word is attached below its left neighbor; the first word is attached below the technical root,

• RightChainInit – each word is attached below its right neighbor; the last word is attached below the technical root.

The last two are used only for computing the baseline scores.

5.2 Dirichlet hyper-parameters

Following (Brody, 2010), we set the Dirichlet hyper-parameters $\alpha_1$ and $\alpha_2$ to values 0.01 and 0.05 respectively. We did not optimize the values carefully because our preliminary experiments confirm the observation of (Brody, 2010): limited variations (up to an order of magnitude) in these parameters have only a negligible effect on the final results.

5.3 Number of iterations

Experiments showed that the sampling algorithm makes only little changes of probabilities after the 30th iteration (see the Figure 3). All the experiments were running with 30 “burn-in” iterations and then other 20 iterations from which the final dependency trees were computed.

5.4 Parsing

We can simply take the trees after the last iteration and declare them as a result. A better way is, however, take the last (in our case 20) iterations and create an average trees (or average directed graphs). We tested two procedures for creating an average tree (or graph) from $n$ different trees (graphs):

• Max – We attach each node to its most frequent parent. This method allows cycles.
**MST** – Each edge has a weight proportional to the number of times it appeared during the iterations. A maximum spanning tree algorithm (Chu and Liu, 1965) is then applied on each sentence. This method always creates trees.

### 6 Experiments and Evaluation

#### 6.1 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.

### 6.2 Baseline and upper bound estimates

We evaluate four baselines straightforwardly corresponding to four corpus initiation procedures described in Section 5.1: **RandBaseline**, **RandTreeBaseline**, **LeftChainBaseline**, and **RightChainBaseline**.

In order to have an upper bound limit, we use Ryan McDonald’s implementation of Maximum Spanning Tree parser (McDonald et al., 2005) (**SupervisedMST**). Only features based on reduced POS tags are accessible to the parser. We use the data described in Section 2 both for training and evaluation in the 10-fold cross-validation fashion and present the average result.

The results of the baseline and upper bound experiments are summarized in Table 1.

| Parser          | Tags    | UAS  | UUAS | NED  |
|-----------------|---------|------|------|------|
| RandBaseline    | –       | 12.0 | 19.9 | 27.5 |
| RandTreeB.      | –       | 11.9 | 21.0 | 31.0 |
| LeftChainB.     | –       | 30.2 | 53.6 | 67.2 |
| RightChainB.    | –       | 25.5 | 52.0 | 60.6 |
| SupervisedMST   | CoarsePOS | 73.9 | 78.6 | 86.6 |
| SupervisedMST   | FinePOS  | 82.5 | 84.9 | 90.3 |

Table 1: Lower and upper bounds for unsupervised parsing of Czech based on reduced POS tags.

### 6.3 Results for Czech

Selected experiments and results for Czech are summarized in Table 2. We started with a simple configuration without sampling constraints. Then we were gradually adding our improvements and constraints: **MST** parsing, **Tree** and **SingleRoot** constraint and **NounRoot** model. Everything was measured both for **CoarsePOS** and **FinePOS** tags and evaluated with all three measures.

We can see that CoarsePOS tags work better if we do not use **SingleRoot** constraint or **NounRoot** model. Adding **NounRoot** model improves the UAS by 8 percent. We choose the settings of the experiment number 10 (which uses all our improvements and constraints) as the best configuration for Czech. It has the highest **UUAS** score and the values of the other scores are very close to the maximum achieved values.

### 6.4 Learning curves

It is useful to draw learning curves in order to see how well the learning algorithm can exploit additional data. Figure 4 shows the speed of growth of UAS for our best unsupervised configuration in
### Table 2: Evaluation of different configurations of the unsupervised parser for Czech.

| n. | Initialization | Tags | Models | Constraints | Parsing | UAS | UUAS | NED |
|----|----------------|------|--------|-------------|---------|-----|------|------|
| 1  | Random         | CoarsePOS | Dep+Dist | Max         | 45.1    | 51.2 | 55.8 |
| 2  | Random         | FinePOS   | Dep+Dist | Max         | 41.3    | 47.6 | 51.0 |

#### Baseline configuration:
1. Random CoarsePOS Dep+Dist – Max 45.1 51.2 55.8
2. Random FinePOS Dep+Dist – Max 41.3 47.6 51.0

#### Parsing with Maximum spanning tree algorithm:
3. Random CoarsePOS Dep+Dist – MST 44.8 58.8 67.1
4. Random FinePOS Dep+Dist – MST 36.7 50.1 55.1

#### Using tree-sampling:
5. RandomTree CoarsePOS Dep+Dist Tree MST 45.5 55.1 59.5
6. RandomTree FinePOS Dep+Dist Tree MST 36.2 46.6 50.0

#### Single-root constraint added:
7. RandomTree CoarsePOS Dep+Dist+SingleRoot Tree+SingleRoot MST 41.8 58.9 72.2
8. RandomTree FinePOS Dep+Dist+SingleRoot Tree+SingleRoot MST 41.2 58.6 70.8

#### Noun-Root repression model added:
9. RandomTree CoarsePOS Dep+Dist+NounRoot Tree+SingleRoot MST 49.6 62.2 73.3
10. RandomTree FinePOS Dep+Dist+NounRoot Tree+SingleRoot MST 49.8 62.6 73.0

#### Experiments with constraints on the best configuration:
11. RandomTree FinePOS Dep+Dist+NounRoot – MST 42.0 56.3 62.8
12. RandomTree CoarsePOS Dep+Dist+NounRoot – MST 50.0 59.8 66.9
13. RandomTree FinePOS Dep+Dist+NounRoot Tree MST 46.8 55.9 61.1
14. RandomTree FinePOS Dep+Dist+NounRoot SingleRoot MST 40.8 58.0 66.6

#### Other selected experiments:
15. RandomTree FinePOS Dep+Dist+NounRoot – Max 45.1 51.2 55.8
16. RandomTree FinePOS Dep+Dist+NounRoot – Max 44.6 50.5 53.0
17. RandomTree FinePOS Dep+Dist+NounRoot Tree+SingleRoot Max 49.9 62.5 72.8

Table 2: Evaluation of different configurations of the unsupervised parser for Czech.

![Learning curves for Czech](image)

Figure 4: Learning curves for Czech: UAS of unsupervised (our best configuration) and supervised (unlexicalized McDonald’s MST) parsers as functions of data size. FinePOS tags were used.

One can see that from 10,000 tokens the UAS for our best configuration grows very little and we do not need more data if we are dealing with POS tags only. We suppose that more data would be needed when using lexicalization.

#### 6.5 Error analysis

Table 3 shows attachment scores for individual coarse-grained Czech POS tags. One can see very low UAS values with particles, interjections, and punctuation (special characters not filtered in the preprocessing step), however, these categories are not frequent in the corpus. Prepositions and conjunctions are more frequent, but their attachment score is still only 20.4% and 14.2% respectively. This fact is caused mainly by reversed dependencies; our parser attaches prepositions below nouns and conjunctions below verbs, while in the corpus, prepositions dominate nouns and conjunctions dominate verbs. These reversed dependencies are treated as correct with UUAS and NED measures.

| CPOS          | Occurrences | UAS [%] | Err. [%] |
|---------------|-------------|---------|----------|
| N (nouns)     | 21934       | 48.5    | 18.8     |
| A (adjectives)| 12890       | 80.1    | 2.6      |
| V (verbs)     | 10946       | 55.1    | 7.2      |
| P (pronouns)  | 6294        | 66.2    | 2.6      |
| D (adverbs)   | 4025        | 49.4    | 3.3      |
| R (prepositions)| 2596      | 20.4    | 8.2      |
| C (numerals)  | 1884        | 40.5    | 2.2      |
| J (conjunctions)| 957        | 14.2    | 4.7      |
| T (particles) | 198         | 23.6    | 0.5      |
| I (interjections)| 5        | 27.8    | 0.0      |
| Z (punctuation)| 3         | 18.8    | 0.0      |

Table 3: UAS for individual coarse-grained Czech POS tags. The “Err.” column shows the percentage of errors on the whole corpus.

Nouns make most errors in total, especially in the longer noun phrases, where the correct struc-
Table 4: 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., 2011a) in Tables 5 and 6 respectively.

| Language  | Baselines | Results |
|-----------|-----------|---------|
|           | CoNLL     | Our-NR  | Our    | Spi5 | Spi6 |
|           | rand. | left | right |       |       |       |
| Arabic    | ar       | 2007  | 3.9   | 59.0 | 6.0  | 24.8  | 25.0 | 22.0 | **42.5** |
| Bulgarian | bg       | 2006  | 8.0   | 38.8 | 17.9 | **51.4** | 25.4 | 44.3 | 43.9 |
| Catalan   | ca       | 2007  | 3.9   | 30.0 | 24.8 | 56.3  | 55.3 | **63.8** | 59.8 |
| Czech     | cs       | 2007  | 7.4   | 29.6 | 24.2 | **33.3** | 24.3 | 31.4 | 28.4 |
| Danish    | da       | 2006  | 6.7   | 47.8 | 13.1 | 38.6  | 30.2 | **44.0** | 38.3 |
| German    | de       | 2006  | 7.2   | 22.0 | 23.4 | 21.8  | 26.7 | **33.5** | 30.4 |
| Greek     | el       | 2007  | 4.9   | 19.7 | 31.4 | 33.4  | 39.0 | 21.4 | 13.2 |
| English   | en       | 2007  | 4.4   | 21.0 | 29.4 | 23.8  | 24.0 | 34.9 | **45.2** |
| Spanish   | es       | 2006  | 4.3   | 29.8 | 24.7 | **54.6** | 53.0 | 33.3 | 50.6 |
| Basque    | eu       | 2007  | 11.1  | 23.0 | 30.5 | 34.7  | 29.1 | **43.6** | 24.0 |
| Hungarian | hu       | 2007  | 6.5   | 5.5  | 41.4 | **48.1** | 48.0 | 23.0 | 34.7 |
| Italian   | it       | 2007  | 4.2   | 37.4 | 21.6 | **60.6** | 57.5 | 37.6 | 52.3 |
| Japanese  | ja       | 2006  | 14.2  | 13.8 | 67.2 | **53.5** | 52.2 | **53.5** | 50.2 |
| Dutch     | nl       | 2006  | 7.5   | 24.5 | 28.0 | **43.4** | 32.2 | 32.5 | 27.8 |
| Portuguese| pt       | 2006  | 5.8   | 31.2 | 25.8 | 41.8  | **43.2** | 34.4 | 36.7 |
| Slovenian | sl       | 2006  | 7.9   | 26.6 | 24.3 | **34.6** | 25.4 | 33.6 | 32.2 |
| Swedish   | sv       | 2006  | 7.8   | 27.8 | 25.9 | 26.9  | 23.3 | **42.5** | 50.0 |
| Turkish   | tr       | 2006  | 6.4   | 1.5  | 65.4 | 32.1  | 32.2 | 33.4 | **35.9** |
| Chinese   | zh       | 2007  | 15.3  | 13.4 | 41.3 | 34.6  | 21.0 | 34.5 | **43.2** |
| Average   |         | 7.2   | 26.4 | 29.8 | **39.4** | 35.1 | **36.7** | 39.3 |

6.6 Results for CoNLL languages

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 (experiment 10 in Table 2) and the same configuration without using Noun-Root dependency repression (experiment 8). 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. Unlike in 6.3, 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.

The results are shown in Table 4. The Random, LeftChain, and RightChain baselines are compared to our results and to the results reported by (Spitkovsky et al., 2011a). It is obvious, that using the Noun-Root suppression (“Our-NR” column) improves the parsing quality for the majority of languages and has higher scores for 12 (out of 19) languages than previous results (“Spi5” and “Spi6”). If we do not use the Noun-Root suppression (“Our” column), the scores are higher for 6 (7) languages compared to “Spi5” (“Spi6”), but the averaged attachment score is quite similar.

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

7 Conclusions

We described our novel work on unsupervised dependency parser based on Gibbs sampling. We showed that introducing treeness constraint in sampling improves attachment score for Czech from about 45% to 50%. The other improvement was caused by repressing Noun-Root dependencies. We reached 49.9% unlabeled attachment score for Czech. If we apply the same parser configuration to 19 languages available in the CoNLL 2006 and 2007 data, we outperform the previously published results for 12 languages.

Our method does not work well for English. It reached only 24% UAS, which is far below the RightChain baseline. This is the opposite of other approaches (based on DMV), which are very good for English and whose results for other languages
are presented rarely. In the future, we would like to add a fertility model and introduce lexicalization. We are also aware that the parsing quality strongly depends on the tag set, so we plan to incorporate some form of unsupervised tagging or word clustering.

Acknowledgement

This research was supported by the grants GA201/09/H057 (Res Informatica), MSM0021620838, GAUK 116310, and by the European Commission’s 7th Framework Program (FP7) under grant agreement no 247762 (FAUST). We thank anonymous reviewers for their valuable comments and suggestions.

References

D. Aldous. 1985. Exchangeability and related topics. In l’Ecole d’ete de probabilites de Saint-Flour, pages 1–198, Berlin, Germany. Springer.

Samuel Brody. 2010. 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. Association for Computational Linguistics.

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

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

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

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

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

Jan Hajic et al. 2006. Prague Dependency Treebank 2.0. CD-ROM, Linguistic Data Consortium, LDC Catalog No.: LDC2006T01, Philadelphia.

William P. Headlen, III, Mark Johnson, and David McClosky. 2009. 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. Association for Computational Linguistics.

Dan Klein and Christopher D. Manning. 2004. 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. Association for Computational Linguistics.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajic. 2005. Non-Projective Dependency Parsing using Spanning Tree Algorithms. In Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP), pages 523–530, Vancouver, BC, Canada.

Tahira Naseem and Regina Barzilay. 2011. Using Semantic Cues to Learn Syntax. In AAAI.

Joakim Nivre, Johan Hall, Sandra Kübler, Ryan McDonald, Jens Nilsson, Sebastian Riedel, and Deniz Yuret. 2007. The CoNLL 2007 Shared Task on Dependency Parsing. In Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007, pages 915–932, Prague, Czech Republic, June. Association for Computational Linguistics.

Roy Schwartz, Omri Abend, Roi Reichart, and Ari Rappoport. 2011. 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. Association for Computational Linguistics.

Noah Ashton Smith. 2007. Novel estimation methods for unsupervised discovery of latent structure in natural language text. Ph.D. thesis, Baltimore, MD, USA. AAI3240799.

Valentin I. Spitkovsky, Hiyan Alshawi, and Daniel Jurafsky. 2011a. Lateen EM: Unsupervised training with multiple objectives, applied to dependency grammar induction. In Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, pages 830–837, Portland, Oregon, USA, June. Association for Computational Linguistics.

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