Exploiting Reducibility in Unsupervised Dependency Parsing

David Mareček and **Zdeněk Žabokrtský**
Charles University in Prague, Faculty of Mathematics and Physics
Institute of Formal and Applied Linguistics
Malostranské náměstí 25, 11800 Prague, Czech Republic
{marecek,zabokrtsky}@ufal.mff.cuni.cz

Abstract

The possibility of deleting a word from a sentence without violating its syntactic correctness belongs to traditionally known manifestations of syntactic dependency. We introduce a novel unsupervised parsing approach that is based on a new n-gram reducibility measure. We perform experiments across 18 languages available in CoNLL data and we show that our approach achieves better accuracy for the majority of the languages then previously reported results.

1 Introduction

The true nature of the notion of dependency (after removing sedimentary deposits of rules imposed only by more or less arbitrary conventions) remains still somewhat vague and elusive. This holds in spite of a seemingly strong background intuition and even after a decade of formalized large-scale dependency-based resources being available to the research community. It is undeniable that a huge progress has been reached in the field of supervised dependency parsing, especially due to the CoNLL shared task series. However, when it comes to unsupervised parsing, there are surprisingly few clues we could rely on.

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 (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 by 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). 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 be dependent on some other (still present) word.

As it is usual with dichotomies in natural languages, it seems more adequate to use a continuous scale instead of the reducible-irreducible opposition. That is why we introduce a simple reducibility measure based on n-gram corpus statistics. We employ this reducibility measure as the main feature in our unsupervised parsing procedure.

The procedure is based on a commonly used Bayesian inference technique called Gibbs sampling (Gilks et al., 1996). In our sampler, the more reducible a given token is, the more likely it is to be sampled as a dependant and not as a head. After certain number of sampling iterations, for each sentence a final dependency tree is created (one token per node, including punctuation) that maximizes the product of edge probabilities gathered along the sampling history.

Our approach allows to utilize information from very large corpora. While the computationally demanding sampling procedure can be applied only on limited data, the unrepeated precomputation of statistics for reducibility estimates can easily exploit much larger data.

We are not aware of any other published work on unsupervised parsing employing 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. It seems that the two views of dependency (frequent co-occurrence of head-dependant pair, versus reducibility of the dependant) are rather complementary, so fruitful combinations can be hopefully expected in future.

The remainder of this paper is structured as follows. Section 2 briefly outlines the state of the art in unsupervised dependency parsing. Our measure of reducibility based on a large monolingual corpus is presented in Section 3. Section 4 shows our models which serve for generating probability estimates for edge sampling described in Section 5. Experimental parsing results for languages included in CoNLL shared task treebanks are summarized in Section 6. Section 7 concludes this article.

2 Related Work

The most 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 (2004). The inference algorithm was further improved by Smith (2007) and Cohen et al. (2008). Headden, Johnson and McClosky (2009) introduced the Extended Valence Grammar (EVG) and added lexicalization and smoothing. Blunsom and Cohn (2010) use tree substitution grammars, which allow learning larger dependency fragments.

Unfortunately, many of these works show results only for English.[1] However, the main feature of unsupervised methods should be their applicability across a wide range of languages. Such experiments were done by Spitkovsky (2011b; 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 linguistic analysis (Spitkovsky et al., 2011a) shows that the unsupervised part-of-speech tags may be more useful for this task than the supervised ones.

Another possibility for obtaining dependency structures for languages without any linguistically annotated resources can be the projection using a parallel treebank with a resource-rich language (typically English). McDonald et al. (2011) showed that such projection produce better structures than the current unsupervised parsers do. However, our task is different. We would like to produce structures that are not burdened by any linguistic conventions.

In this paper, we describe a novel approach to unsupervised dependency parsing. Our model differs from DMV, since we employ the *reducibility* feature and use *fertility* of nodes instead of generating *STOP* signs.

We use Gibbs sampling procedure for inference instead of Variational Bayes, which has been more common for induction of linguistic strucures. Gibbs sampling algorithm for grammar induction was used also by Mareček and Žabokrtský (2011). However, their sampling algorithm produces generally non-projective trees. Our sampler, which is described in Section 5, introduces a completely different small-change operator that guarantees projective edges.

3 Computing Reducibility scores

We call a word (or a sequence of words) in a sentence *reducible*, if the sentence after removing the word remains grammatically correct. Although we cannot automatically recognize grammaticality of such newly created sentence, we can search for it in a large corpus. If we find it, we assume the word was reducible in the original sentence.

Since the number of such reducible word sequences found in any corpus will be low, we determine the reducibility scores from their individual types (part-of-speech tags). This then implicitly allows some sharing of the scores between different word sequences.

The necessity to search for the whole sentences in the corpus and not only for some smaller context (considering, for example, just left and right neigh-

[1]The state-of-the-art unsupervised parsers achieve more than 50% of attachment score measured on the Penn Treebank.

bor), 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 large contexts. For instance, verbs in free word-order languages can be placed almost at any position in a sentence; therefore, without the full sentence context, they would have to be considered as reducible. To prevent this, we decided to work exclusively with the full sentence context instead of shorter contexts.

Other way that would lead to 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 two following 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 removed from the prepositional phrase. Using part-of-speech tags instead of word forms is thus not suitable for computing reducibility scores.

Although we search for reducible sequences of word forms in the corpus, we compute reducibility scores for sequences of part-of-speech tags. This requires to have the corpus morphologically disambiguated. A sequences of part-of-speech tags will be denoted as "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 such 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.[2] If so, then such occurrence of PoS n-gram is reducible, otherwise it is not. We denote the number of such reducible occurrences of PoS n-gram g by $r(g)$. The number of all its occurrences is $c(g)$.

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

unigrams	R	bigrams	R	trigrams	R
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 1: Reducibility scores of the most frequent English 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)

The relative reducibility $R(g)$ of a PoS n-gram g is then computed as

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

where the normalization constant N, which expresses relative reducibility over all the PoS n-grams (denoted by G), causes the scores are concentrated around the value 1.

$$N = \frac{\sum_{g \in G} (r(g) + \sigma_1)}{\sum_{g \in G} (c(g) + \sigma_2)} \tag{2}$$

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

$$\sigma_1 = \frac{\sum_{g \in G} r(g)}{\sum_{g \in G} c(g)}, \qquad \sigma_2 = 1 \tag{3}$$

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 1, 2, and 3 show reducibility scores of the most frequent PoS n-grams of three selected languages: English, German, and Czech. If we consider only unigrams, 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

unigrams	R	bigrams	R	trigrams	R
VVPP	0.00	NN APPR	0.00	NN APPR NN	0.01
APPR	0.27	APPR ART	0.00	ADJA NN APPR	0.01
VVFIN	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 2: Reducibility scores of the most frequent German 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	R	bigrams	R	trigrams	R
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 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)

and adverbs are very high for all three examined languages. That is desired, because the reducible unigrams will more likely become leaves in 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.[3]

4 Models

We introduce a new generative model that is different from the widely used Dependency Model with Valence (DMV). In DMV (Klein and Manning, 2004) and in the extended model EVG (Headden III et al., 2009), there is a *STOP* sign indicating that no more dependents in a given direction will be generated. Given a certain head, all its dependents in left direction are generated first, then the *STOP* sign in that direction, then all its right dependents and then *STOP* in the other direction. This process continues recursively for all generated dependents.

Our model introduces *fertility* of a node, which substitutes the *STOP* sign. For a given head, we first generate the number of its left and right children (*fertility model*) and then we fill these positions by generating its individual dependents (*edge model*). If a zero fertility is generated in both the directions, the head becomes a leaf.

Besides the fertility model and the edge model, we use two more models (*subtree model* and *distance model*), which force the generated trees to have more desired shape.[4]

4.1 Fertility Model

We express a fertility of a node by a pair of numbers: the number of its left dependents and the number of its right dependents. For example, fertility "1-3" means that the node has one left and three right dependents, fertility "0-0" indicates that it is a leaf. Fertility is conditioned by part-of-speech tag of the node and it is computed following the Chinese restaurant process. This means that if a specific fertility has been frequent for a given PoS tag in the past, it is more likely to be generated again. The formula for computing probability of fertility f_i of a word on the position i in the corpus is as follows:

$$P_f(f_i|t_i) = \frac{c^{-i}(``t_i, f_i") + \alpha P_0(f_i)}{c^{-i}(``t_i") + \alpha}, \qquad (4)$$

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

[4]In fact, the *subtree model* and the *distance model* disrupt a bit the generative story, because the probabilites do not sum up to one when they are used. However, they proved to help with inducing better linguistic structures.

where t_i is part-of-speech tag of the word on the position i, $c^{-i}(``t_i, f_i")$ stands for the count of words with PoS tag t_i and fertility f_i in the history, and P_0 is a prior probability for the given fertility which depends on the total number of node dependents denoted by $|f_i|$ (the sum of numbers of left and right dependents):

$$P_0(f_i) = \frac{1}{2^{|f_i|+1}} \tag{5}$$

This prior probability has a nice property: for a given number of nodes, the product of fertility probabilities over all the nodes is equal for all possible dependency trees. This ensures the stability of this model during the inference.

Besides the basic fertility model, we introduce also an extended fertility model, which uses frequency of a given word form for generating 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) attachment of auxiliary verbs depends on the annotation style, but number of their children is also not very variable. The higher the frequency of a word form, the higher probability mass is concentrated on one specific number of children and the lower Dirichlet hyperparameter α in Equation 4 is needed. The extended fertility is described by equation

$$P'_f(f_i|t_i, w_i) = \frac{c^{-i}(``t_i, f_i") + \frac{\alpha_e}{F(w_i)} P_0(f_i)}{c^{-i}(``t_i") + \frac{\alpha_e}{F(w_i)}}, \tag{6}$$

where $F(w_i)$ is a frequency of the word w_i, which is computed as a number of words w_i in our corpus divided by number of all words.

4.2 Edge Model

After the fertility (number of left and right dependents) is generated, the individual slots are filled using the *edge model*. A part-of-speech tag of each dependent is conditioned by part-of-speech tag of the head and the edge direction (position of the dependent related to the head).[5]

Similarly as for the fertility model, we employ Chinese restaurant process to assign probabilities of individual dependent.

$$P_e(t_j|t_i, d_j) = \frac{c^{-i}(``t_i, t_j, d_j") + \beta}{c^{-i}(``t_i, d_j") + \beta|T|}, \tag{7}$$

where t_i and t_j are the part-of-speech tags of the head and the generated dependent respectively; d_j is a direction of edge between the words i and j, which can have two values: *left* and *right*. $c^{-i}(``t_i, t_j, d_j")$ stands for the count of edges $t_i \leftarrow t_j$ with the direction d_j in the history, $|T|$ is a number of unique tags in the corpus and β is a Dirichlet hyperparameter.

4.3 Distance Model

Distance model is an auxiliary model that prevents the resulting trees from being too flat. Ideally, it would not be needed, but experiments showed that it helps to infer better trees. This simple model says that shorter edges are more probable than longer ones. We define probability of a distance between a word and its parent as its inverse value,[6] which is then normalized by the normalization constant ϵ_d.

$$P_d(i, j) = \frac{1}{\epsilon_d} \left(\frac{1}{|i - j|} \right)^{\gamma} \tag{8}$$

The hyperparameter γ determines the weight of this model.

4.4 Subtree Model

The subtree model uses the reducibility measure. It plays an important role since it forces the reducible words to be leaves and reducible n-grams to be subtrees. Words with low reducibility are forced towards the root of the tree. We define $desc(i)$ as a sequence of tags $[t_l, \ldots, t_r]$ that corresponds to all the descendants of the word w_i including w_i, i.e. the whole subtree of w_i. The probability of such subtree is proportional to its reducibility $R(desc(i))$. The hyperparameter δ determines the weight of the model; ϵ_s is a normalization constant.

$$P_s(i) = \frac{1}{\epsilon_s} R(desc(i))^{\delta} \tag{9}$$

[5]For the edge model purposes, the PoS tag of the technical root is set to '`<root>`' and it is in the zero-th position in the sentence, so the head word of the sentence is always its right dependent.

[6]Distance between any word and the technical root of the dependency tree was set to 10. Since each technical root has only one dependent, this value does not affect the model.

4.5 Probability of the Whole Treebank

We want to maximize the probability of the whole generated treebank, which is computed as follows:

$$P_{treebank} = \prod_{i=1}^{n} (P'_f(f_i|t_i, w_i) \tag{10}$$
$$P_e(t_i|t_{\pi(i)}, d_i) \tag{11}$$
$$P_d(i, \pi(i)) \tag{12}$$
$$P_s(i)), \tag{13}$$

where $\pi(i)$ denotes the parent of the word on the position i. We multiply the probabilities of fertility, edge, distance from parent, and subtree over all words (nodes) in the corpus. The extended fertility model P'_f can be substituted by its basic variant P_f.

5 Sampling Algorithm

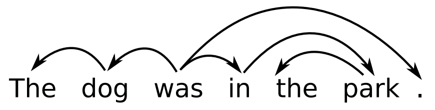
For stochastic searching for the most probable dependency trees, we employ Gibbs sampling, a standard Markov Chain Monte Carlo technique (Gilks et al., 1996). In each iteration, we loop over all words in the corpus in a random order and change the dependencies in their neighborhood (a small change described in Section 5.2). In the end, "average" trees based on the whole sampling are built.

5.1 Initialization

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

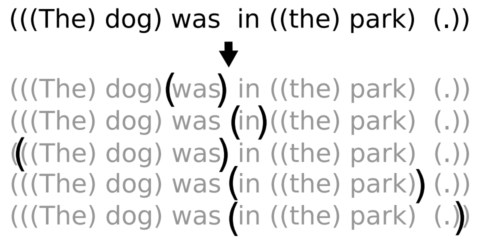
- For each sentence, we choose randomly one word as the head and attach all other words to it.
- We are picking one word after another in a random order and we attach it to the nearest left (or right) neighbor that has not been attached yet. The left-right choice is made by a coin flip. If it is not possible to attach a word to one side, we attach it to the other side. The last unattached word becomes 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 and we choose the first one due to its simplicity.



(((The) dog) was (in ((the) park)) (.))

Figure 1: Arrow and bracketing notation of a projective dependency tree.


(((The) dog) was in ((the) park) (.))

(((The) dog)(was) in ((the) park) (.))
(((The) dog) was (in)((the) park) (.))
((((The) dog) was) in ((the) park) (.))
(((The) dog) was (in ((the) park)) (.))
(((The) dog) was (in ((the) park) (.)))

Figure 2: An example of small change in a projective tree. The bracket (in the park) is removed and there are five possibilities how to replace it.

5.2 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 1.

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. An example of such change is in Figure 2.

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

1. Pick a random non-root word w (the word *in* in our example) and find its parent p (the word *was*).
2. Find all other children of w and p (the words *dog*, *park*, and .) and denote this set 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 adjacent to the word d as a continuous subset of C and attach all words from D to d. D may be also empty.

5. Attach the remaining words from C that were not in D to the new head g.

5.3 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. These counts are collected over the whole corpus with the collection-rate 0.01.[7]

When the samling is finished, we build final dependency trees based on the edge counts obtained during the sampling. We employ the maximum spanning tree (MST) algorithm (Chu and Liu, 1965) to find them; the weights of edges for computing MST correspond to the number of times they were present during the sampling. This averaging method was used also by Mareček and Žabokrtský (2011).

Other possibilities for obtaining final dependency trees would be using Eisner's projective algorithm (Eisner, 1996) or using annealing method (favoring more likely changes) at the end of the sampling. However, the general non-projective MST algorithm enable non-projective edges, which are by no means negligible in treebanks (Havelka, 2007).

6 Experiments and Evaluation

We evaluate our parser on 20 treebanks (18 languages) included in CoNLL shared tasks 2006 (Buchholz and Marsi, 2006) and 2007 (Nivre et al., 2007).

Similarly to some previous papers on unsupervised parsing (Gillenwater et al., 2011; Spitkovsky et al., 2011b), the tuning experiments were performed on English only. We used English for checking functionality of the individual models and for optimizing hyperparameter values. The best configuration of the parser achieved on English development data was 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.

[7]After each small change is made, the edges from the whole corpus are collected with a probability 0.01.

language	tokens (mil.)	language	tokens (mil.)
Arabic	19.7	Greek	20.9
Basque	14.1	Hungarian	26.3
Bulgarian	18.8	Italian	39.7
Catalan	27.0	Japanese	2.6
Czech	20.3	Portuguese	31.7
Danish	15.9	Slovenian	13.7
Dutch	27.1	Spanish	53.4
English	85.0	Swedish	19.2
German	56.9	Turkish	16.5

Table 4: Wikipedia texts statistics

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 we compute n-gram reducibility scores.

The treebanks are taken from the CoNLL shared task 2006 and 2007. The experiments are performed for all languages except for Chinese.[8] We use only the testing parts of the treebanks (the files `test.conll`) for the dependency tree induction. As a source of the part-of-speech tags, we use the fine-grained gold PoS tags, which are in the fifth column in the CoNLL format.

For obtaining reducibility scores, we used the W2C corpus[9] of Wikipedia articles, which was downloaded by Majliš and Žabokrtský (2012). Their statistics across languages are shown in Table 4. To make them useful, the necessary preprocessing steps must have been done. The texts were first automatically segmented and tokenized[10] and then they were part-of-speech tagged by TnT tagger (Brants, 2000), which was trained on the respective CoNLL training data (the files `train.conll`). The quality of such tagging is not very high, since we do not use any lexicons[11] or pretrained models. However, it is sufficient for obtaining good reducibility scores.

[8]We do not have appropriate Chinese segmenter that would segment Chinese texts in the same way as in CoNLL.

[9]`http://ufal.mff.cuni.cz/~majlis/w2c/`

[10]The segmentation to sentences and tokenization was performed using the TectoMT framework (Popel and Žabokrtský, 2010).

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

6.2 Setting the Hyperparameters

The applicability of individual models and their parameters were tested on development data set of English (the file `en/dtest.conll` in CoNLL shared task 2007).

After several experiments, we have observed that the extended fertility model provides better results than the basic fertility model; the parser using the basic fertility model achieved 44.1% attachment score for English, whereas the extended fertility model increased the score to 46.8%. The four hyperparameters α_e (extended fertility model), β (edge model), γ (distance model), and δ (subtree model), were set by a grid search algorithm,[12] which found the following optimal values:

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

In informal experiments, parameters were tuned also for other treebanks and we found out that they vary across languages. Therefore, adjusting the hyperparameters on another language would probably change the scores significantly.

6.3 Evaluation

The best setting from the experiments on English is now used for evaluating our parser on all CoNLL languages. To be able to compare our parser attachment score to previously published results, the following steps must be done:

- We take the testing part of each treebank (the file `test.conll`) and remove all the punctuation marks. If the punctuation node is not a leaf, its children are attached to the parent of the removed node.
- Some previous papers report results on up-to-10-words sentences only. Therefore we extract such sentences from the test data and evaluate on this subsets as well.

[12]Here we make use of manually annotated trees. However, we use only English treebank an we are setting only four numbers out of several previously given values (e.g α_e out of 0.01, 0.1, 1, 10). These numbers could be tuned also by inspecting the outputs. So we believe this method can be treated as unsupervised.

CoNLL		≤ 10 tokens		all sentences	
language	year	gil11	our	spi11	our
Arabic	06	–	40.5	16.6	**26.5**
Arabic	07	–	48.0	**49.5**	27.9
Basque	07	–	30.8	24.0	**26.8**
Bulgarian	06	**58.3**	53.2	43.9	**46.0**
Catalan	07	–	63.5	**59.8**	47.0
Czech	06	53.2	**58.9**	27.7	**49.5**
Czech	07	–	63.7	28.4	**48.0**
Danish	06	45.9	**49.5**	38.3	**38.6**
Dutch	06	33.5	**48.8**	27.8	**44.2**
English	07	–	64.1	45.2	**49.2**
German	06	46.7	**60.8**	30.4	**44.8**
Greek	07	–	30.2	13.2	**20.2**
Hungarian	07	–	61.8	34.7	**51.8**
Italian	07	–	50.5	**52.3**	43.3
Japanese	06	57.7	**65.4**	50.2	**50.8**
Portuguese	06	54.0	**62.3**	36.7	**50.6**
Slovenian	06	**50.9**	21.0	**32.2**	18.1
Spanish	06	57.9	**67.3**	50.6	**51.9**
Swedish	06	45.0	**60.5**	**50.0**	48.2
Turkish	07	–	13.0	**35.9**	15.7
Average:		50.3*	**54.7***	37.4	**40.0**

Table 5: Comparison of directed attachment scores with previously reported results on CoNLL treebanks. The column "gil11" contains results reported by Gillenwater et al (2011) (see the best configuration in Table 7 in their paper). They provided only results on sentences of up to 10 tokens from CoNLL 2006 treebanks. Results in the column "spi11" are taken from Spitkovsky et al (2011b), best configuration in Table 6 in their paper. The average score in the last line is computed across all comparable results, i.e. for comparison with "gil11" only the CoNLL'06 results are averaged (*). Our parser was not evaluated on Turkish CoNLL'06 data and Chinese data, because we have not them available.

The resulting scores are given in Table 5. We compare our results with results previously reported by Gillenwater (2011) and Spitkovsky (2011b), who used the CoNLL data for evaluation too. Since they provide results for several configurations of their parsers, we choose only the best one from each the paper. We define the best configuration as the one with the highest average attachment score across all the tested languages.

We can see that our parser outperforms the previously published ones. In one case, it is better for 8 out of 10 data sets, in the other case, it is better for 14 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 unannotated corpora for computing reducibility scores, while the others used only the CoNLL data.

6.4 Error Analysis

Our main motivation for developing an unsupervised dependency parser was that we wanted to be able to parse any language. However, the experiments show that our parser fails for some languages. In this section, we try to analyze and explain some of the most substantial types of errors.

Auxiliary verbs in Slovenian – In the Slovenian treebank, many verbs are composed of two words: main verb (marked as `Verb-main`) and auxiliary verb (`Verb-copula`). Our parser choose the auxiliary verb as the head and the main verb and all its dependants become its children. That is why the attachment score is so poor (only 18.1%). In fact, the induced structure is not so bad. The main verb is switched with the auxiliary one which causes also the wrong attachment of all its dependants.

Articles in German – Attachment of about one half of German articles is wrong. Instead of the article being attached below the appropriate noun, the noun is attached below the article. It is a similar problem as the aforementioned Slovenian auxiliary verbs. The dependency between content and function word is switched and the dependants of the content word are attached to the function word. Klein and Manning (2004) observed a similar behavior in their experiments with DMV.

Noun phrases in English – The structure of phrases that consist of more nouns are often induced badly. This is caused probably by ignoring word forms. For example, the structure of the sequence '*NN NN NN*' can be hardly recognized by our parser.

fert.	edge	dist.	subtr.	en	de	cs
			(random baseline)	19.8	18.4	26.7
✓				8.71	13.7	14.9
	✓			18.9	20.2	26.5
		✓		23.6	19.5	25.3
			✓	28.2	23.7	33.5
✓	✓			21.2	22.9	23.5
✓		✓		19.9	19.7	25.5
✓			✓	7.8	17.5	22.7
	✓	✓		24.1	19.5	27.1
	✓		✓	25.5	27.5	40.7
		✓	✓	31.2	25.2	33.1
✓	✓	✓		30.7	26.2	22.0
✓	✓		✓	14.1	18.1	34.6
✓		✓	✓	36.1	32.2	38.9
	✓	✓	✓	34.8	26.7	42.4
✓	✓	✓	✓	**46.8**	**36.5**	**47.2**

Table 6: Ablation analysis. Unlabeled attachment scores for different combinations of model components (fertility model, edge model, distance model and subtree model). The scores are computed on all sentences of the development data. Punctuation is included into the evaluation.

6.5 Ablation Analysis

To investigate the impact of individual components of the model, we run the parser for all possible component combinations. We choose three languages along the scale of word order freedom: English (very rigid word order), Czech (relatively free word order), and German (somewhere in the middle). The attachment scores are shown in Table 6. If no model is used for the inference and the sampling algorithm samples completely random trees, we get the *random baseline* score, which is 19.8% for English[13]. From the perspective of the *subtree model*, which implements the reducibility feature, we can see that it is the most useful model here. Alone, it improves the score for English to 28.2%. If we do not use it, the score decreases from 46.8% (when all models are used) to 30.7%. Very important is also the distance model which eliminates the possibility of attaching all words to one head word. If we omit

[13]This relatively high baseline scores are caused by the MST algorithm, which chooses the most frequent edges from random trees i.e. the shortest ones.

it, the score for English falls drastically to 14.1%. Some combinations of models have their scores far below the baseline. This is caused by the fact that some regularities have been found but the structures are induced differently and thus all attachments are wrong.

6.6 Induction without Wikipedia Corpus

We have performed also experiments using exclusively the CoNLL data. However, the numbers of reducible words in CoNLL training set were very low (50 words at maximum in CoNLL 2006 training data and 10 words at maximum in CoNLL 2007 training data). This led to completely unreliable reducibility scores and the consequent poor results.

7 Conclusions and Future Work

We have shown that employing the reducibility feature is useful in unsupervised dependency parsing task. We extracted the n-gram reducibility scores from a large corpus, and then made the computationally demanding inference on smaller data using only these scores. We evaluated our parser on 18 languages included in CoNLL and for 14 of them, we achieved higher attachment scores than previously published results.

The most errors were caused by function words, which sometimes take over the dependents of adjacent content words. This can be caused by the fact that the reducibility cannot handle function words correctly, because they must be reduced together with a content word, not one after another.

In future work, we would like to estimate the hyperparameters automatically. Furthermore, we would like to get rid of manually designed PoS tags and use some kind of unsupervised clusters in order to have all the annotation process completely unsupervised. We would also like to employ lexicalized models that should help in situations in which the PoS tags are too coarse.

Finally, we would like to move towards deeper syntactic structures, where the tree would be formed only by content words and the function words would be treated in a different way.

Software

The source code of our unsupervised dependency parser including the script for computing reducibility scores from large corpora is available at `http://ufal.mff.cuni.cz/~marecek/udp`.

Acknowledgement

This research was supported by the grants GAUK 116310, GA201/09/H057 (Res Informatica), LM2010013, MSM0021620838, and by the European Commission's 7th Framework Program (FP7) under grant agreement n° 247762 (FAUST). We thank anonymous reviewers for their valuable comments and suggestions.

References

Phil Blunsom and Trevor Cohn. 2010. 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. Association for Computational Linguistics.

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

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.

Jason Eisner. 1996. Three New Probabilistic Models for Dependency Parsing: An Exploration. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING-96)*, pages 340–345, Copenhagen, August.

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

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

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

Jiří Havelka. 2007. 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.

William P. Headden 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.

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

Markéta Lopatková, Martin Plátek, and Vladislav Kuboň. 2005. 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. Springer.

Martin Majliš and Zdeněk Žabokrtský. 2012. Language richness of the web. In *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC 2012)*, Istanbul, Turkey, May. European Language Resources Association (ELRA).

David Mareček and Zdeněk Žabokrtský. 2011. 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.

Ryan McDonald, Slav Petrov, and Keith Hall. 2011. 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. Association for Computational Linguistics.

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.

Martin Popel and Zdeněk Žabokrtský. 2010. TectoMT: modular NLP framework. In *Proceedings of the 7th international conference on Advances in natural language processing*, IceTAL'10, pages 293–304, Berlin, Heidelberg. Springer-Verlag.

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

Valentin I. Spitkovsky, Hiyan Alshawi, and Daniel Jurafsky. 2011b. 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)*.

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