Stop-probability estimates computed on a large corpus improve Unsupervised Dependency Parsing

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

Abstract

Even though the quality of unsupervised dependency parsers grows, they often fail in recognition of very basic dependencies. In this paper, we exploit a prior knowledge of STOP-probabilities (whether a given word has any children in a given direction), which is obtained from a large raw corpus using the reducibility principle. By incorporating this knowledge into Dependency Model with Valence, we managed to considerably outperform the state-of-the-art results in terms of average attachment score over 20 treebanks from CoNLL 2006 and 2007 shared tasks.

1 Introduction

The task of unsupervised dependency parsing (which strongly relates to the grammar induction task) has become popular in the last decade, and its quality has been greatly increasing during this period.

The first implementation of Dependency Model with Valence (DMV) (Klein and Manning, 2004) with a simple inside-outside inference algorithm (Baker, 1979) achieved 36% attachment score on English and was the first system outperforming the adjacent-word baseline.[1]

Current attachment scores of state-of-the-art unsupervised parsers are higher than 50% for many languages (Spitkovsky et al., 2012; Blunsom and Cohn, 2010). This is still far below the supervised approaches, but their indisputable advantage is the fact that no annotated treebanks are needed and the induced structures are not burdened by any linguistic conventions. Moreover, supervised parsers always only simulate the treebanks they were trained on, whereas unsupervised parsers have an ability to be fitted to different particular applications.

Some of the current approaches are based on the DMV, a generative model where the grammar is expressed by two probability distributions: $P_{choose}(c_d|c_h, dir)$, which generates a new child c_d attached to the head c_h in the direction dir (left or right), and $P_{stop}(STOP|c_h, dir, \cdots)$, which makes a decision whether to generate another child of c_h in the direction dir or not.[2] Such a grammar is then inferred using sampling or variational methods.

Unfortunately, there are still cases where the inferred grammar is very different from the grammar we would expect, e.g. verbs become leaves instead of governing the sentences. Rasooli and Faili (2012) and Bisk and Hockenmaier (2012) made some efforts to boost the verbocentricity of the inferred structures; however, both of the approaches require manual identification of the POS tags marking the verbs, which renders them useless when unsupervised POS tags are employed.

The main contribution of this paper is a considerable improvement of unsupervised parsing quality by estimating the P_{stop} probabilities externally using a very large corpus, and employing this prior knowledge in the standard inference of DMV. The estimation is done using the reducibility principle introduced in (Mareček and Žabokrtský, 2012). The reducibility principle postulates that if a word (or a sequence of words) can be removed from a sentence without violating its grammatical correctness, it is a leaf (or a subtree) in its dependency structure. For the purposes of this paper, we assume the following hypothesis:

If a sequence of words can be removed from

[1] The adjacent-word baseline is a dependency tree in which each word is attached to the previous (or the following) word. The attachment score of 35.9% on all the WSJ test sentences was taken from (Blunsom and Cohn, 2010).

[2] The P_{stop} probability may be conditioned by additional parameters, such as adjacency adj or fringe word c_f, which will be described in Section 4.

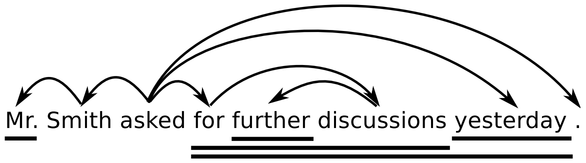


Figure 1: Example of a dependency tree. Sequences of words that can be reduced are underlined.

a sentence without violating its grammatical correctness, no word outside the sequence depends on any word in the sequence.

Our hypothesis is a generalization of the original hypothesis since it allows a reducible sequence to form several adjacent subtrees.

Let's outline the connection between the P_{stop} probabilities and the property of reducibility. Figure 1 shows an example of a dependency tree. Sequences of reducible words are marked by thick lines below the sentence. Consider for example the word *"further"*. It can be removed and thus, according to our hypothesis, no other word depends on it. Therefore, we can deduce that the P_{stop} probability for such word is high both for the left and for the right direction. The phrase *"for further discussions"* is reducible as well and we can deduce that the P_{stop} of its first word (*"for"*) in the left direction is high since it cannot have any left children. We do not know anything about its right children, because they can be located within the sequence (and there is really one in Figure 1). Similarly, the word *"discussions"*, which is the last word in this sequence, cannot have any right children and we can estimate that its right P_{stop} probability is high. On the other hand, non-reducible words such, as the verb *"asked"* in our example, can have children, and therefore their P_{stop} can be estimated as low for both directions.

The most difficult task in this approach is to automatically recognize reducible sequences. This problem, together with the estimation of the stop-probabilities, is described in Section 3. Our model, not much different from the classic DMV, is introduced in Section 4. Section 5 describes the inference algorithm based on Gibbs sampling. Experiments and results are discussed in Section 6. Section 7 concludes the paper.

2 Related Work

Reducibility: The notion of reducibility belongs to the traditional linguistic criteria for recognizing dependency relations. As mentioned e.g. by Kübler et al. (2009), the head h of a construction c determines the syntactic category of c and can often replace c. In other words, the descendants of h can be often removed without making the sentence incorrect. Similarly, in the Dependency Analysis by Reduction (Lopatková et al., 2005), the authors assume that stepwise deletions of dependent elements within a sentence preserve 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).

We have directly utilized the aforementioned criteria for dependency relations in unsupervised dependency parsing in our previous paper (Mareček and Žabokrtský, 2012). Our dependency model contained a submodel which directly prioritized subtrees that form reducible sequences of POS tags. Reducibility scores of given POS tag sequences were estimated using a large corpus of Wikipedia articles. The weakness of this approach was the fact that longer sequences of POS tags are very sparse and no reducibility scores could be estimated for them. In this paper, we avoid this shortcoming by estimating the STOP probabilities for individual POS tags only.

Another task related to reducibility is *sentence compression* (Knight and Marcu, 2002; Cohn and Lapata, 2008), which was used for text summarization. The task is to shorten the sentences while retaining the most important pieces of information, using the knowledge of the grammar. Conversely, our task is to induce the grammar using the sentences and their shortened versions.

Dependency Model with Valence (DMV) has been the most popular approach to unsupervised dependency parsing in the recent years. It was introduced by Klein and Manning (2004) and further improved by Smith (2007) and Cohen et al. (2008). Headden III et al. (2009) introduce the Extended Valence Grammar and add lexicalization and smoothing. Blunsom and Cohn (2010) use tree substitution grammars, which allow learning of larger dependency fragments by employing the Pitman-Yor process. Spitkovsky et al. (2010) improve the inference using iterated learning of increasingly longer sentences. Further improvements were achieved by better dealing with punctuation (Spitkovsky et al., 2011b) and new "boundary" models (Spitkovsky et al., 2012).

Other approaches to unsupervised dependency parsing were described e.g. in (Søgaard, 2011), (Cohen et al., 2011), and (Bisk and Hockenmaier, 2012). There also exist "less unsupervised" approaches that utilize an external knowledge of the POS tagset. For example, Rasooli and Faili (2012) identify the last verb in the sentence, minimize its probability of reduction and thus push it to the root position. Naseem et al. (2010) make use of manually-specified universal dependency rules such as *Verb→Noun*, *Noun→Adjective*. McDonald et al. (2011) identify the POS tags by a cross-lingual transfer. Such approaches achieve better results; however, they are useless for grammar induction from plain text.

3 STOP-probability estimation

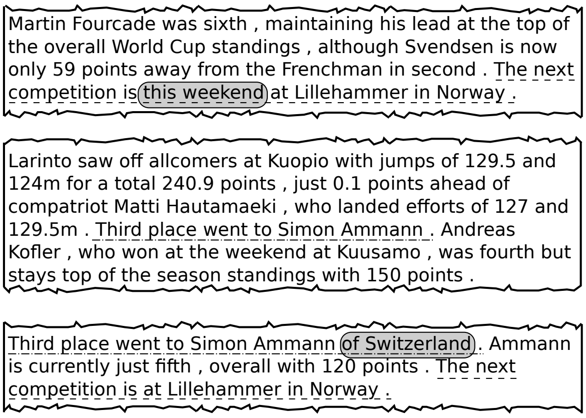
3.1 Recognition of reducible sequences

We introduced a simple procedure for recognition of reducible sequences in (Mareček and Žabokrtský, 2012): The particular sequence of words is removed from the sentence and if the remainder of the sentence exists elsewhere in the corpus, the sequence is considered reducible. We provide an example in Figure 2. The bigram *"this weekend"* in the sentence *"The next competition is this weekend at Lillehammer in Norway."* is reducible since the same sentence without this bigram, i.e., *"The next competition is at Lillehammer in Norway."*, is in the corpus as well. Similarly, the prepositional phrase *"of Switzerland"* is also reducible.

It is apparent that only very few reducible sequences can be found by this procedure. If we use a corpus containing about 10,000 sentences, it is possible that we found no reducible sequences at all. However, we managed to find a sufficient amount of reducible sequences in corpora containing millions of sentences, see Section 6.1 and Table 1.

3.2 Computing the STOP-probability estimations

Recall our hypothesis from Section 1: If a sequence of words is reducible, no word outside the sequence can depend on any word in the sequence. Or, in terms of dependency structure: A reducible sequence consists of one or more adjacent subtrees. This means that the first word of a reducible sequence does not have any left children and, similarly, the last word in a reducible sequence does not have any right children. We make use of this property directly for estimating P_{stop} probabilities.



Martin Fourcade was sixth , maintaining his lead at the top of the overall World Cup standings , although Svendsen is now only 59 points away from the Frenchman in second . The next competition is this weekend at Lillehammer in Norway .

Larinto saw off allcomers at Kuopio with jumps of 129.5 and 124m for a total 240.9 points , just 0.1 points ahead of compatriot Matti Hautamaeki , who landed efforts of 127 and 129.5m . Third place went to Simon Ammann . Andreas Kofler , who won at the weekend at Kuusamo , was fourth but stays top of the season standings with 150 points .

Third place went to Simon Ammann of Switzerland . Ammann is currently just fifth , overall with 120 points . The next competition is at Lillehammer in Norway .

Figure 2: Example of reducible sequences of words found in a large corpus.

Hereinafter, $P^{est}_{stop}(c_h, dir)$ denotes the STOP-probability we want to estimate from a large corpus; c_h is the head's POS tag and dir is the direction in which the STOP probability is estimated. If c_h is very often in the first position of reducible sequences, $P^{est}_{stop}(c_h, \mathit{left})$ will be high. Similarly, if c_h is often in the last position of reducible sequences, $P^{est}_{stop}(c_h, \mathit{right})$ will be high.

For each POS tag c_h in the given corpus, we first compute its left and right "raw" score $S_{stop}(c_h, \mathit{left})$ and $S_{stop}(c_h, \mathit{right})$ as the relative number of times a word with POS tag c_h was in the first (or last) position in a reducible sequence found in the corpus. We do not deal with sequences longer than a trigram since they are highly biased.

$$S_{stop}(c_h, \mathit{left}) = \frac{\text{\# red.seq. } [c_h, \ldots] + \lambda}{\text{\# } c_h \text{ in the corpus}}$$

$$S_{stop}(c_h, \mathit{right}) = \frac{\text{\# red.seq. } [\ldots, c_h] + \lambda}{\text{\# } c_h \text{ in the corpus}}$$

Note that the S_{stop} scores are not probabilities. Their main purpose is to sort the POS tags according to their "reducibility".

It may happen that for many POS tags there are no reducible sequences found. To avoid zero scores, we use a simple smoothing by adding λ to each count:

$$\lambda = \frac{\text{\# all reducible sequences}}{W},$$

where W denotes the number of words in the given corpus. Such smoothing ensures that more frequent irreducible POS tags get a lower S_{stop} score than the less frequent ones.

Since reducible sequences found are very sparse, the values of $S_{stop}(c_h, dir)$ scores are very small. To convert them to estimated probabilities $P^{est}_{stop}(c_h, dir)$, we need a smoothing that fulfills the following properties:

(1) P^{est}_{stop} is a probability and therefore its value must be between 0 and 1.

(2) The number of *no-stop* decisions (no matter in which direction) equals to W (number of words) since such decision is made before each word is generated. The number of *stop* decisions is $2W$ since they come after generating the last children in both the directions. Therefore, the average $P^{est}_{stop}(h, dir)$ over all words in the treebank should be $2/3$.

After some experimenting, we chose the following normalization formula

$$P^{est}_{stop}(c_h, dir) = \frac{S_{stop}(c_h, dir)}{S_{stop}(c_h, dir) + \nu}$$

with a normalization constant ν. The condition (1) is fulfilled for any positive value of ν. Its exact value is set in accordance with the requirement (2) so that the average value of P^{est}_{stop} is $2/3$.

$$\sum_{dir \in \{l,r\}} \sum_{c \in C} count(c) P^{est}_{stop}(c, dir) = \frac{2}{3} \cdot 2W,$$

where $count(c)$ is the number of words with POS tag c in the corpus. We find the unique value of ν that fulfills the previous equation numerically using a binary search algorithm.

4 Model

We use the standard generative Dependency Model with Valence (Klein and Manning, 2004). The generative story is the following: First, the head of the sentence is generated. Then, for each head, all its left children are generated, then the left STOP, then all its right children, and then the right STOP. When a child is generated, the algorithm immediately recurses to generate its subtree. When deciding whether to generate another child in the direction dir or the STOP symbol, we use the $P^{dmv}_{stop}(STOP|c_h, dir, adj, c_f)$ model. The new child c_d in the direction dir is generated according to the $P_{choose}(c_d|c_h, dir)$ model. The probability of the whole dependency tree T is the following:

$$\begin{aligned} P_{tree}(T) = {} & P_{choose}(head(T)|ROOT, right) \\ & \cdot P_{tree}(D(head(T))) \end{aligned}$$

$$\begin{aligned} & P_{tree}(D(c_h)) = \\ & \prod_{dir \in \{l,r\}} \prod_{\substack{c_d \in \\ deps(dir,h)}} P^{dmv}_{stop}(\neg STOP|c_h, dir, adj, c_f) \\ & \qquad P_{choose}(c_d|c_h, dir) P_{tree}(D(c_d)) \\ & \qquad P^{dmv}_{stop}(STOP|c_h, dir, adj, c_f), \end{aligned}$$

where $P_{tree}(D(c_h))$ is probability of the subtree governed by h in the tree T.

The set of features on which the P^{dmv}_{stop} and P_{choose} probabilities are conditioned varies among the previous works. Our P^{dmv}_{stop} depends on the head POS tag c_h, direction dir, adjacency adj, and fringe POS tag c_f (described below). The use of adjacency is standard in DMV and enables us to have different P^{dmv}_{stop} for situations when no child was generated so far ($adj = 1$). That is, $P^{dmv}_{stop}(c_h, dir, adj = 1, c_f)$ decides whether the word c_h has any children in the direction dir at all, whereas $P^{dmv}_{stop}(h, dir, adj = 0, c_f)$ decides whether another child will be generated next to the already generated one. This distinction is of crucial importance for us: although we know how to estimate the STOP probabilities for $adj = 1$ from large data, we do not know anything about the STOP probabilities for $adj = 0$.

The last factor c_f, called fringe, is the POS tag of the previously generated sibling in the current direction dir. If there is no such sibling (in case $adj = 1$), the head c_h is used as the fringe c_f. This is a relatively novel idea in DMV, introduced by Spitkovsky et al. (2012). We decided to use the fringe word in our model since it gives slightly better results.

We assume that the distributions of P_{choose} and P^{dmv}_{stop} are good if the majority of the probability mass is concentrated on few factors; therefore, we apply a Chinese Restaurant process (CRP) on them.

The probability of generating a new child node c_d attached to c_h in the direction dir given the history (all the nodes we have generated so far) is computed using the following formula:

$$P_{choose}(c_d|c_h, dir) = \\ = \frac{\alpha_c \frac{1}{|C|} + count^{-}(c_d, c_h, dir)}{\alpha_c + count^{-}(c_h, dir)},$$

where $count^{-}(c_d, c_h, dir)$ denotes the number of times a child node c_d has been attached to c_h in the direction dir in the history. Similarly, $count^{-}(c_h, dir)$ is the number of times something has been attached to c_h in the direction dir. The α_c is a hyperparameter and $|C|$ is the number of distinct POS tags in the corpus.[3]

The STOP probability is computed in a similar way:

$$P^{dmv}_{stop}(STOP|c_h, dir, adj, c_f) = \\ = \frac{\alpha_s \frac{2}{3} + count^{-}(STOP, c_h, dir, adj, c_f)}{\alpha_s + count^{-}(c_h, dir, adj, c_f)}$$

where $count^{-}(STOP, c_h, dir, adj, c_f)$ is the number of times a head c_h had the last child c_f in the direction dir in the history.

The contribution of this paper is the inclusion of the stop-probability estimates into the DMV. Therefore, we introduce a new model $P^{dmv+est}_{stop}$, in which the probability based on the previously generated data is linearly combined with the probability estimates based on large corpora (Section 3).

$$P^{dmv+est}_{stop}(STOP|c_h, dir, 1, c_f) = \\ = (1-\beta) \cdot \frac{\alpha_s \frac{2}{3} + count^{-}(STOP, c_h, dir, 1, c_f)}{\alpha_s + count^{-}(c_h, dir, 1, c_f)} \\ + \beta \cdot P^{est}_{stop}(c_h, dir)$$

$$P^{dmv+est}_{stop}(STOP|c_h, dir, 0, c_f) = \\ = P^{dmv}_{stop}(STOP|c_h, dir, 0, c_f)$$

The hyperparameter β defines the ratio between the CRP-based and estimation-based probability. The definition of the $P^{dmv+est}_{stop}$ for $adj = 0$ equals the basic P^{dmv}_{stop} since we are able to estimate only the probability whether a particular head POS tag c_h can or cannot have children in a particular direction, i.e if $adj = 1$.

Finally, we obtain the probability of the whole generated treebank as a product over the trees:

$$P_{treebank} = \prod_{T \in treebank} P_{tree}(T).$$

An important property of the CRP is the fact that the factors are exchangeable – i.e. no matter how the trees are ordered in the treebank, the $P_{treebank}$ is always the same.

5 Inference

We employ the Gibbs sampling algorithm (Gilks et al., 1996). Unlike in (Mareček and Žabokrtský, 2012), where edges were sampled individually, we sample whole trees from all possibilities on a given sentence using dynamic programming. The algorithm works as follows:

1. A random projective dependency tree is assigned to each sentence in the corpus.
2. *Sampling:* We go through the sentences in a random order. For each sentence, we sample a new dependency tree based on all other trees that are currently in the corpus.
3. Step 2 is repeated in many iterations. In this work, the number of iterations was set to 1000.
4. After the burn-in period (which was set to the first 500 iterations), we start collecting counts of edges between particular words that appeared during the sampling.
5. *Parsing:* Based on the collected counts, we compute the final dependency trees using the Chu-Liu/Edmonds' algorithm (1965) for finding maximum directed spanning trees.

5.1 Sampling

Our goal is to sample a new projective dependency tree T with probability proportional to $P_{tree}(T)$. Since the factors are exchangeable, we can deal with any tree as if it was the last one in the corpus.

We use dynamic programming to sample a tree with N nodes in $\mathcal{O}(N^4)$ time. Nevertheless, we sample trees using a modified probability $P'_{tree}(T)$. In $P_{tree}(T)$, the probability of an edge depends on counts of all other edges, including the edges in the same tree. We instead use $P'_{tree}(T)$, where the counts are computed using only the other trees in the corpus, i.e., probabilities

[3] The number of classes $|C|$ is often used in the denominator. We decided to put its reverse value into the numerator since we observed such model to perform better for a constant value of α_c over different languages and tagsets.

of edges of T are independent. There is a standard way to sample using the real $P_{tree}(T)$ – we can use $P'_{tree}(T)$ as a proposal distribution in the Metropolis-Hastings algorithm (Hastings, 1970), which then produces trees with probabilities proportional to $P_{tree}(T)$ using acceptance-rejection scheme. We do not take this approach and we sample proportionally to $P'_{tree}(T)$ only, because we believe that for large enough corpora, the two distributions are nearly identical.

To sample a tree containing words $w_1, \ldots, w_N$ with probability proportional to $P'_{tree}(T)$, we first compute three tables:

- $t_i(g,i,j)$ for $g < i$ or $g > j$ is the sum of probabilities of any tree on words $w_i, \ldots, w_j$ whose root is a child of w_g, but not an outermost child in its direction;
- $t_o(g,i,j)$ is the same, but the tree is the outermost child of w_g;
- $f_o(g,i,j)$ for $g < i$ or $g > j$ is the sum of probabilities of any forest on words $w_i, \ldots, w_j$, such that all the trees are children of w_g and are the outermost children of w_g in their direction.

All the probabilities are computed using the P'_{tree}. If we compute the tables inductively from the smallest trees to the largest trees, we can precompute all the $\mathcal{O}(N^3)$ values in $\mathcal{O}(N^4)$ time.

Using these tables, we sample the tree recursively, starting from the root. At first, we sample the root r proportionally to the probability of a tree with the root r, which is a product of the probability of left children of r and right children of r. The probability of left children of r is either $P'_{stop}(STOP|r, \mathit{left})$ if r has no children, or $P'_{stop}(\neg STOP|r, \mathit{left}) f_o(r, 1, r-1)$ otherwise; the probability of right children is analogous.

After sampling the root, we sample the ranges of its left children, if any. We sample the first left child range l_1 proportionally either to $t_o(r, 1, r-1)$ if $l_1 = 1$, or to $t_i(r, l_1, r-1) f_o(r, 1, l_1 - 1)$ if $l_1 > 1$. Then we sample the second left child range l_2 proportionally either to $t_o(r, 1, l_1 - 1)$ if $l_2 = 1$, or to $t_i(r, l_2, l_1 - 1) f_o(r, 1, l_2 - 1)$ if $l_2 > 1$, and so on, while there are any left children. The right children ranges are sampled similarly. Finally, we recursively sample the children, i.e., their roots, their children and so on. It is simple to verify using the definition of P_{tree} that the described method indeed samples trees proportionally to P'_{tree}.

5.2 Parsing

Beginning the 500th iteration, we start collecting counts of individual dependency edges during the remaining iterations. After each iteration is finished (all the trees in the corpus are re-sampled), we increment the counter of all directed pairs of nodes which are connected by a dependency edge in the current trees.

After the last iteration, we use these collected counts as weights and compute maximum directed spanning trees using the Chu-Liu/Edmonds' algorithm (Chu and Liu, 1965). Therefore, the resulting trees consist of edges maximizing the sum of individual counts:

$$T_{MST} = \arg\max_T \sum_{e \in T} count(e)$$

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

6 Experiments

6.1 Data

We use two types of resources in our experiments. The first type are CoNLL treebanks from the year 2006 (Buchholz and Marsi, 2006) and 2007 (Nivre et al., 2007), which we use for inference and for evaluation. As is the standard practice in unsupervised parsing evaluation, we removed all punctuation marks from the trees. In case a punctuation node was not a leaf, its children are attached to the parent of the removed node.

For estimating the STOP probabilities (Section 3), we use the Wikipedia articles from W2C corpus (Majliš and Žabokrtský, 2012), which provide sufficient amount of data for our purposes. Statistics across languages are shown in Table 1.

The Wikipedia texts were automatically tokenized and segmented to sentences so that their tokenization was similar to the one in the CoNLL evaluation treebanks. Unfortunately, we were not able to find any segmenter for Chinese that would produce a desired segmentation; therefore, we removed Chinese from evaluation.

The next step was to provide the Wikipedia texts with POS tags. We employed the TnT tagger (Brants, 2000) which was trained on the re-

language	tokens (mil.)	red. seq.	language	tokens (mil.)	red. seq.
Arabic	19.7	546	Greek	20.9	1037
Basque	14.1	645	Hungarian	26.3	2237
Bulgarian	18.8	1808	Italian	39.7	723
Catalan	27.0	712	Japanese	2.6	31
Czech	20.3	930	Portuguese	31.7	4765
Danish	15.9	576	Slovenian	13.7	513
Dutch	27.1	880	Spanish	53.4	1156
English	85.0	7603	Swedish	19.2	481
German	56.9	1488	Turkish	16.5	5706

Table 1: Wikipedia texts statistics: total number of tokens and number of reducible sequences found in them.

spective CoNLL training data. The quality of such tagging is not very high since we do not use any lexicons or pretrained models. However, it is sufficient for obtaining usable stop probability estimates.

6.2 Estimated STOP probabilities

We applied the algorithm described in Section 3 on the prepared Wikipedia corpora and obtained the stop-probabilities P^{est}_{stop} in both directions for all the languages and their POS tags. To evaluate the quality of our estimations, we compare them with P^{tb}_{stop}, the stop probabilities computed directly on the evaluation treebanks. The comparisons on five selected languages are shown in Figure 3. The individual points represent the individual POS tags, their size (area) shows their frequency in the particular treebank. The y-axis shows the stop probabilities estimated on Wikipedia by our algorithm, while the x-axis shows the stop probabilities computed on the evaluation CoNLL data. Ideally, the computed and estimated stop probabilities should be the same, i.e. all the points should be on the diagonal.

Let's focus on the graphs for English. Our method correctly recognizes that adverbs RB and adjectives JJ are often leaves (their stop probabilities in both directions are very high). Moreover, the estimates for RB are even higher than JJ, which will contribute to attaching adverbs to adjectives and not reversely. Nouns (NN, NNS) are somewhere in the middle, the stop probabilities for proper nouns (NNP) are estimated higher, which is correct since they have much less modifiers then the common nouns NN. The determiners are more problematic. Their estimated stop probability is not very high (about 0.65), while in the real treebank they are almost always leaves.

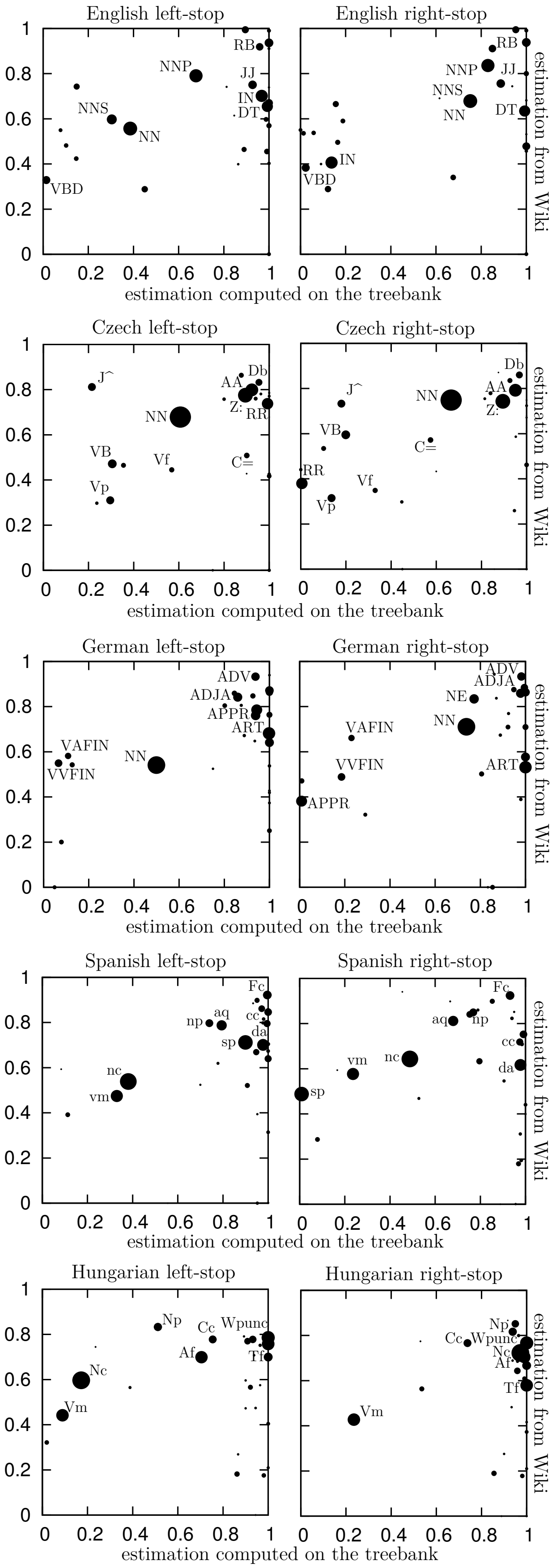


Figure 3: Comparison of P^{est}_{stop} probabilities estimated from raw Wikipedia corpora (y-axis) and of P^{tb}_{stop} probabilities computed from CoNLL treebanks (x-axis). The area of each point shows the relative frequency of an individual tag.

This is caused by the fact that determiners are often obligatory in English and cannot be simply removed as, e.g., adjectives. The stop probabilities of prepositions (IN) are also very well recognized. While their left-stop is very probable (prepositions always start prepositional phrases), their right-stop probability is very low. The verbs (the most frequent verbal tag is VBD) have very low both right and left-stop probabilities. Our estimation assigns them the stop probability about 0.3 in both directions. This is quite high, but still, it is one of the lowest among other more frequent tags, and thus verbs tend to be the roots of the dependency trees. We could make similar analyses for other languages, but due to space reasons we only provide graphs for Czech, German, Spanish, and Hungarian in Figure 3.

6.3 Settings

After a manual tuning, we have set our hyperparameters to the following values:

$$\alpha_c = 50, \quad \alpha_s = 1, \quad \beta = 1/3$$

We have also found that the Gibbs sampler does not always converge to a similar grammar. For a couple of languages, the individual runs end up with very different trees. To prevent such differences, we run each inference 50 times and take the run with the highest final $P_{treebank}$ (see Section 4) for the evaluation.

6.4 Results

Table 2 shows the results of our unsupervised parser and compares them with results previously reported in other works. In order to see the impact of using the estimated stop probabilities (using model $P_{stop}^{dmv+est}$), we provide results for classical DMV (using model P_{stop}^{dmv}) as well. We do not provide results for Chinese since we do not have any appropriate tokenizer at our disposal (see Section 3), and also for Turkish from CoNLL 2006 since the data is not available to us.

We now focus on the third and fourth column of Table 2. The addition of estimated stop probabilities based on large corpora improves the parsing accuracy on 15 out of 20 treebanks. In many cases, the improvement is substantial, which means that the estimated stop probabilities forced the model to completely rebuild the structures. For example, in Bulgarian, if the P_{stop}^{dmv} model is used, all the prepositions are leaves and the verbs seldom govern sentences. If the $P_{stop}^{dmv+est}$ model is used, prepositions correctly govern nouns and verbs move to roots. We observe similar changes on Swedish as well. Unfortunately, there are also negative examples, such as Hungarian, where the addition of the estimated stop probabilities decreases the attachment score from 60.1% to 34%. This is probably caused by not very good estimates of the right-stop probability (see the last graph in Figure 3). Nevertheless, the estimated stop probabilities increase the average score over all the treebanks by more than 12% and therefore prove its usefulness.

In the last two columns of Table 2, we provide results of two other works reported in the last year. The first one (*spi12*) is the DMV-based grammar inducer by Spitkovsky et al. (2012),[4] the second one (*mar12*) is our previous work (Mareček and Žabokrtský, 2012). Comparing with (Spitkovsky et al., 2012), our parser reached better accuracy on 12 out of 20 treebanks. Although this might not seem as a big improvement, if we compare the average scores over the treebanks, our system significantly wins by more than 6%. The second system (*mar12*) outperforms our parser only on one treebank (on Italian by less than 3%) and its average score over all the treebanks is only 40%, i.e., more than 8% lower than the average score of our parser.

To see the theoretical upper bound of our model performance, we replaced the P_{stop}^{est} estimates by the P_{stop}^{tb} estimates computed from the evaluation treebanks and run the same inference algorithm with the same setting. The average attachment score of such reference DMV is almost 65%. This shows a huge space in which the estimation of STOP probabilities could be further improved.

7 Conclusions and Future Work

In this work, we studied the possibility of estimating the DMV stop-probabilities from a large raw corpus. We proved that such prior knowledge about stop-probabilities incorporated into the standard DMV model significantly improves the unsupervised dependency parsing and, since we are not aware of any other fully unsupervised dependency parser with higher average attachment score over CoNLL data, we state that we reached a new state-of-the-art result.[5]

[4] Possibly the current state-of-the-art results. They were compared with many previous works.

[5] A possible competitive work may be the work by Blunsom and Cohn (2010), who reached 55% accuracy on English as well. However, they do not provide scores measured on other CoNLL treebanks.

CoNLL		this work			other systems	
language	year	P_{stop}^{dmv}	$P_{stop}^{dmv+est}$	reference P_{stop}^{dmv+tb}	spi12	mar12
Arabic	06	10.6 (±8.7)	**38.2 (±0.5)**	*61.2*	10.9	26.5
Arabic	07	22.0 (±0.1)	35.3 (±0.2)	*65.3*	**44.9**	27.9
Basque	07	**41.1 (±0.2)**	35.5 (±0.2)	*52.3*	33.3	26.8
Bulgarian	06	25.9 (±1.4)	54.9 (±0.2)	*73.2*	**65.2**	46.0
Catalan	07	34.9 (±3.4)	**67.0 (±1.7)**	*72.0*	62.1	47.0
Czech	06	32.3 (±3.8)	52.4 (±5.2)	*64.0*	**55.1**	49.5
Czech	07	32.9 (±0.8)	51.9 (±5.2)	*62.1*	**54.2**	48.0
Danish	06	30.8 (±4.3)	**41.6 (±1.1)**	*60.0*	22.2	38.6
Dutch	06	25.7 (±5.7)	**47.5 (±0.4)**	*58.9*	46.6	44.2
English	07	36.5 (±5.9)	**55.4 (±0.2)**	*63.7*	29.6	49.2
German	06	29.9 (±4.6)	**52.4 (±0.7)**	*65.5*	39.1	44.8
Greek	07	**42.5 (±6.0)**	26.3 (±0.1)	*64.7*	26.9	20.2
Hungarian	07	**60.8 (±0.2)**	34.0 (±0.3)	*68.3*	58.2	51.8
Italian	07	34.5 (±0.3)	39.4 (±0.5)	*64.5*	40.7	**43.3**
Japanese	06	**64.8 (±3.4)**	61.2 (±1.7)	*76.4*	22.7	50.8
Portuguese	06	35.7 (±4.3)	69.6 (±0.1)	*77.3*	**72.4**	50.6
Slovenian	06	**50.1 (±0.2)**	35.7 (±0.2)	*50.2*	35.2	18.1
Spanish	06	38.1 (±5.9)	**61.1 (±0.1)**	*65.6*	28.2	51.9
Swedish	06	28.0 (±2.3)	**54.5 (±0.4)**	*61.6*	50.7	48.2
Turkish	07	51.6 (±5.5)	**56.9 (±0.2)**	*67.0*	44.8	15.7
Average:		36.4	**48.7**	*64.7*	42.2	40.0

Table 2: Attachment scores on CoNLL 2006 and 2007 data. Standard deviations are provided in brackets. DMV model using standard P_{stop}^{dmv} probability is compared with DMV with $P_{stop}^{dmv+est}$, which incorporates STOP estimations based on reducibility principle. The reference DMV uses P_{stop}^{tb}, which are computed directly on the treebanks. The results reported in previous works by Spitkovsky et al. (2012), and Mareček and Žabokrtský (2012) follows.

In future work, we would like to focus on unsupervised parsing without gold POS tags (see e.g. Spitkovsky et al. (2011a) and Christodoulopoulos et al. (2012)). We suppose that many of the current works on unsupervised dependency parsers use gold POS tags only as a simplification of this task, and that the ultimate purpose of this effort is to develop a fully unsupervised induction of linguistic structure from raw texts that would be useful across many languages, domains, and applications.

The software which implements the algorithms described in this paper, together with P_{stop}^{est} estimations computed on Wikipedia texts, can be downloaded at http://ufal.mff.cuni.cz/~marecek/udp/.

Acknowledgments

This work has been supported by the AMALACH grant (DF12P01OVV02) of the Ministry of Culture of the Czech Republic.

Data and some tools used as a prerequisite for the research described herein have been provided by the LINDAT/CLARIN Large Infrastructural project, No. LM2010013 of the Ministry of Education, Youth and Sports of the Czech Republic.

We would like to thank Martin Popel, Zdeněk Žabokrtský, Rudolf Rosa, and three anonymous reviewers for many useful comments on the manuscript of this paper.

References

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

Yonatan Bisk and Julia Hockenmaier. 2012. Induction of linguistic structure with combinatory categorial grammars. *The NAACL-HLT Workshop on the Induction of Linguistic Structure*, page 90.

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.

Christos Christodoulopoulos, Sharon Goldwater, and Mark Steedman. 2012. Turning the pipeline into a loop: Iterated unsupervised dependency parsing and PoS induction. In *Proceedings of the NAACL-HLT Workshop on the Induction of Linguistic Structure*, pages 96–99, June.

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.

Shay B. Cohen, Dipanjan Das, and Noah A. Smith. 2011. Unsupervised structure prediction with non-parallel multilingual guidance. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 50–61, Stroudsburg, PA, USA. Association for Computational Linguistics.

Trevor Cohn and Mirella Lapata. 2008. Sentence compression beyond word deletion. In *Proceedings of the 22nd International Conference on Computational Linguistics - Volume 1*, COLING '08, pages 137–144, Stroudsburg, PA, USA. Association for Computational Linguistics.

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.

W. Keith Hastings. 1970. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):pp. 97–109.

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.

Kevin Knight and Daniel Marcu. 2002. Summarization beyond sentence extraction: a probabilistic approach to sentence compression. *Artif. Intell.*, 139(1):91–107, July.

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. 2012. Exploiting reducibility in unsupervised dependency parsing. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 297–307, Stroudsburg, PA, USA. Association for Computational Linguistics.

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.

Tahira Naseem, Harr Chen, Regina Barzilay, and Mark Johnson. 2010. Using universal linguistic knowledge to guide grammar induction. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP '10, pages 1234–1244, Stroudsburg, PA, USA. 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.

Mohammad Sadegh Rasooli and Heshaam Faili. 2012. Fast unsupervised dependency parsing with arc-standard transitions. In *Proceedings of ROBUS-UNSUP*, pages 1–9.

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

Anders Søgaard. 2011. From ranked words to dependency trees: two-stage unsupervised non-projective dependency parsing. In *Proceedings of TextGraphs-6: Graph-based Methods for Natural Language Processing*, TextGraphs-6, pages 60–68, Stroudsburg, PA, USA. Association for Computational Linguistics.

Valentin I. Spitkovsky, Hiyan Alshawi, and Daniel Jurafsky. 2010. From baby steps to leapfrog: how "less is more" in unsupervised dependency parsing. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, pages 751–759, Stroudsburg, PA, USA. Association for Computational Linguistics.

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. Punctuation: Making a point in unsupervised dependency parsing. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning (CoNLL-2011)*.

Valentin I. Spitkovsky, Hiyan Alshawi, and Daniel Jurafsky. 2012. Three Dependency-and-Boundary Models for Grammar Induction. In *Proceedings of the 2012 Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL 2012)*.