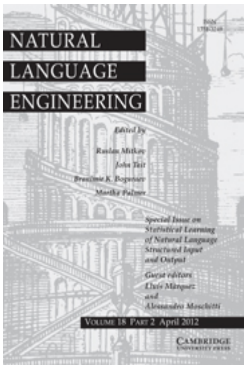

Unsupervised dependency parsing without training

ANDERS SØGAARD

Natural Language Engineering / Volume 18 / Special Issue 02 / April 2012, pp 187 - 203
DOI: 10.1017/S1351324912000022, Published online: 14 March 2012

Natural Language Engineering **18** (2): 187–203. © Cambridge University Press 2012
doi:10.1017/S1351324912000022

Unsupervised dependency parsing without training[†]

ANDERS SØGAARD
Center for Language Technology University of Copenhagen Njalsgade 142
DK-2300 Copenhagen S, Denmark
e-mail: soegaard@hum.ku.dk

(*Received 28 August 2011; accepted 22 December 2011*)

Abstract

Usually unsupervised dependency parsers try to optimize the probability of a corpus by revising the dependency model that is assumed to have generated the corpus. In this paper we explore a different view in which a dependency structure is, among other things, a partial order on the nodes in terms of centrality or saliency. Under this assumption we directly model centrality and derive dependency trees from the ordering of words. The result is an approach to unsupervised dependency parsing that is very different from standard ones in that it requires no training data. The input words are ordered by centrality, and a parse is derived from the ranking using a simple deterministic parsing algorithm, relying on the universal dependency rules defined by Naseem *et al.* (Naseem, T., Chen, H., Barzilay, R., Johnson, M. 2010. Using universal linguistic knowledge to guide grammar induction. In *Proceedings of Empirical Methods in Natural Language Processing*, Boston, MA, USA, pp. 1234–44.). Our approach is evaluated on data from twelve different languages and is remarkably competitive.

1 Introduction

Dependency parsing is the problem of assigning dependency structures to sentences that represent their syntactic properties. Dependency structures are typically trees such that all words but the root of the sentence, typically the main verb, have unique syntactic heads. The dependencies between heads and dependents are typically labeled by grammatical functions, such as subject, object or modifier. Consider, for example, the dependency structures in Figure 1.

In the relative clause in the English sentence, for example, the head of the word *advertising* is the verb *works*. This can be seen from following the in-coming edge from *advertising*. The dependency label indicates that *advertising* is the subject of the verb. Note that the main verbs also have heads. The main verbs are roots of the

[†] This paper continues the work presented at TextGraphs-6 at ACL-HLT 2011 in Portland, OR, USA. The unsupervised dependency parser presented there, however, did not make use of universal dependency rules in parsing (Section 3.2) and used more dense word graphs. The new parser performs better, and the code, which is made public, is a lot simpler. I would like to thank Anders Johannsen, Alessandro Moschitti, Joakim Nivre and the three anonymous reviewers for their valuable suggestions.

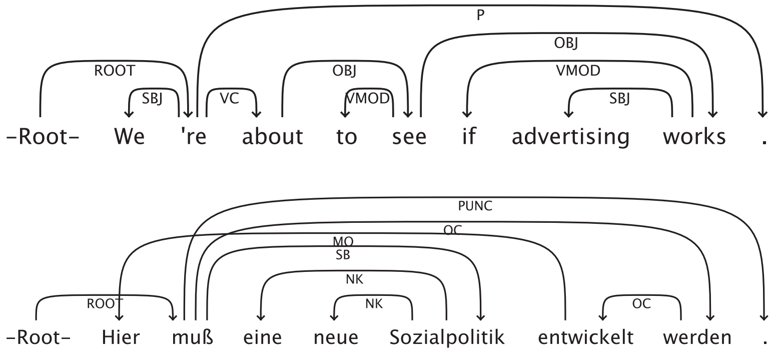
P
OBJ
ROOT
OBJ
VMOD
SBJ
VC
VMOD
SBJ
-Root- We 're about to see if advertising works .
PUNC
OC
MO
SB
NK
ROOT
NK
OC
-Root- Hier muß eine neue Sozialpolitik entwickelt werden .

Fig. 1. Derived dependency structures from the Penn-III treebank of English and the TIGER treebank of German.

sentences, but they are attached to an artificial root node so that we can think of parsing as the problem of assigning a syntactic head to every word (in a way that avoids producing cycles). Note also the difference between the English and German structures. The German structure contains crossing edges; this property is referred to and formally defined below as *non-projectivity*.

In supervised dependency parsing, parsing models are learned from large collections of sentences annotated with dependency structures (treebanks), but in this paper we consider the more difficult problem of unsupervised dependency parsing, which is the problem facing us when we want to parse languages for which no treebanks are available.

Unsupervised dependency parsers do not achieve the same quality as supervised or semi-supervised parsers, but in some situations precision may be less important compared to the cost of producing manually annotated data. Moreover, unsupervised dependency parsing is attractive from a theoretical point of view as it does not rely on a particular style of annotation and may potentially provide insights about the difficulties of human language learning.

Unsupervised dependency parsing has seen rapid progress recently, with error reductions of almost 30% on English in six years (Schwartz *et al.* 2011) and better results for other languages (Gillenwater *et al.* 2010; Naseem *et al.* 2010), but results are still far from what can be achieved with small seeds, language-specific rules (Druck, Mann and McCallum 2009) or using cross-language adaptation (Smith and Eisner 2009; Spreyer and Kuhn 2009; McDonald, Petrov and Hall 2011; Søgaard 2011).

The standard method in unsupervised dependency parsing is to optimize the overall probability of the corpus by assigning trees to its sentences that capture general patterns in the distribution of part-of-speech (POS). This happens in several iterations over the corpus. This method requires clever initialization, which can be seen as a kind of minimal supervision. State-of-the-art unsupervised dependency parsers, except Seginer (2007) and Spitkovsky *et al.* (2011a), also rely on manually annotated text or text processed by supervised POS taggers. Since there is an intimate relationship between POS tagging and dependency parsing, the POS tags can also be seen as a seed or a partial annotation. Inducing a model from the corpus is typically a very slow process.

This paper presents a new and very different approach to unsupervised dependency parsing. The parser does not induce a model from a big corpus, but only considers the sentence in question.[1] The obvious advantage of not relying on training data is that we do not have to worry about whether the test data reflect the same distribution as the target data (sample bias), and since our models are much smaller, parsing will be very fast.

The parser assigns a dependency structure to a sequence of words in two stages. It first decorates the n nodes of what will become our dependency structure with word forms, constructs a directed acyclic graph from the nodes and ranks them using iterative graph-based ranking (Brin and Page 1998). Subsequently, it constructs a tree from the ranked list of words using a simple $\mathcal{O}(n^2)$ parsing algorithm.

Our parser is evaluated on the selection of twelve dependency treebanks also used by Gillenwater *et al.* (2010), and we compare our results to state-of-the-art in multilingual unsupervised dependency parsing (Gillenwater *et al.* 2010; Naseem *et al.* 2010). Evaluation and error analysis are presented in Sections 4 and 5. In Section 6, we also evaluate a version of our parser that is not at all informed by POS.

1.1 Preliminaries

The observed variables in unsupervised dependency parsing are a corpus of sentences $\mathbf{s} = s_1, \ldots, s_n$ where each word w_j in s_i is associated with a POS tag p_j. The hidden variables are dependency structures $\mathbf{t} = t_1, \ldots, t_n$ where s_i labels the vertices of t_i. We assume that dependency structures are trees. Each vertex has a single incoming edge, possibly except one called the root of the tree. In this work and in most of the other works in dependency parsing, we introduce an artificial root node so that all vertices decorated by word forms have an incoming edge.

The dependency structures in Figure 1 are trees decorated with labels and augmented with a linear order on the nodes. Each edge (i, j) is referred to as a dependency between a head word w_i and a dependent word w_j and sometimes written as $w_i \rightarrow w_j$. Let w_0 be the artificial root of the dependency structure. We use $\rightarrow^+$ to denote the transitive closure on the set of edges. Both nodes and edges are typically labeled, but in unsupervised dependency parsing, edge labels are typically ignored. Since a dependency structure is a tree, it satisfies the following three constraints: A dependency structure over a sentence $s : w_1, \ldots, w_n$ is *connected*, i.e.

$$\forall w_i \in s.w_0 \rightarrow^+ w_i$$

In other words, there is a path from the root to all nodes. A dependency structure is also *acyclic*, i.e.

$$\neg\exists w_i \in s.w_i \rightarrow^+ w_i$$

[1] The only exception is a function word extraction step that makes use of whatever textual context is available. In our experiments we use the other test sentences for this step.

This formula says, you can never reach the same node following a non-empty path, which also implies that all paths are finite. Finally, a dependency structure is *single-rooted*, i.e.

$$\forall w_i.\forall w_j.(w_0 \rightarrow w_i \wedge w_0 \rightarrow w_j) \Rightarrow w_i = w_j$$

The last formula says that the artificial root node only has a single dependent. If we also require that each vertex other than the artificial root node has an incoming edge, we have a complete characterization of dependency structures. In sum, a dependency structure is a tree with a linear order on the leaves where the root of the tree for practical reasons is attached to an artificial root node. The artificial root node makes it easier to implement parsing algorithms.

Finally, we define *projectivity*, i.e. whether the linear order is projective with respect to dependency tree, as the property of dependency trees that if $w_i \rightarrow w_j$, it also holds that all words in-between w_i and w_j are dominated by w_i, i.e. $w_i \rightarrow^+ w_k$. Intuitively, a projective dependency structure contains no crossing edges. Projectivity is not a necessary property of dependency structures. Some dependency structures are projective, others are not. Most if not all previous works in unsupervised dependency parsing have focused on projective dependency parsing, building on work in context-free parsing, but our parser is guaranteed to produce well-formed non-projective dependency trees. Our parser will only be able to produce a subclass of all possible non-projective dependency trees; we return to this issue later on. Non-projective parsing algorithms for supervised dependency parsing have, for example, been presented in McDonald *et al.* (2005) and Nivre (2009). Unsupervised learning algorithms for non-projective dependency parsing have been introduced in McDonald and Satta (2007) and Cohen, Rodriguez and Satta (2011b).

1.2 Related work

Dependency Model with Valence (DMV) by Klein and Manning (2004) was the first unsupervised dependency parser to achieve an accuracy for manually POS-tagged English above a right-branching baseline.

Dependency model with valence is a generative model in which the sentence root is generated and then each head recursively generates its left and right dependents. For each $s_i \in \mathbf{s}$, t_i is assumed to have been built the following way: The arguments of a head h in direction d are generated one after another with the probability that no more arguments of h should be generated in direction d conditioned on h, d and whether this would be the first argument of h in direction d. The POS tag of the argument of h is generated for given h and d. Klein and Manning (2004) use expectation maximization (EM) to estimate probabilities with linguistically biasing initialization.

Smith and Eisner (2005) use contrastive estimation instead of EM, while Smith and Eisner (2006) use structural annealing to penalize long-distance dependencies initially, gradually weakening the penalty during training. Cohen, Gimpel and Smith (2008) use Bayesian priors (Dirichlet and Logistic Normal) with DMV. All of the above approaches to unsupervised dependency parsing build on the generative model introduced by Klein and Manning (2004) with different additional assumptions based on linguistic intuition.

In a similar way Gillenwater *et al.* (2010) try to penalize models with a large number of distinct dependency types by using sparse posteriors. They evaluate their system on eleven treebanks from the CoNLL 2006 Shared Task and the Penn-III treebank and achieve state-of-the-art performance.

Headden, Johnson and McClosky (2009) extended the DMV model with additional valence and lexical information.

An exception to using linguistically biased priors is Spitkovsky, Alshawi and Jurafsky (2009), who use predictions on sentences of length n to initialize search on sentences of length $n + 1$. In other words, their method requires no manual tuning and bootstraps itself on increasingly longer sentences.

A very different, but interesting, approach is taken in Brody (2010), who use methods from unsupervised word alignment for unsupervised dependency parsing. In particular, he sees dependency parsing as directional alignment from a sentence (possible dependents) to itself (possible heads) with the modification that words cannot align to themselves; following Klein and Manning (2004) and the subsequent papers mentioned above, Brody (2010) considers sequences of POS tags rather than raw text. Results given below are state-of-the-art, but in some cases better than the DMV model.

2 Ranking dependency tree nodes

The main intuition behind our approach to unsupervised dependency parsing is that the nodes near the root in a dependency structure are in some sense the most important or the most central ones. Semantically, the nodes near the root typically express the main predicate and its arguments. Iterative graph-based ranking (Brin and Page 1998) was first used to rank webpages according to their centrality, and now the technique has found wide applications in natural language processing. Variations of the PageRank algorithm presented by Page and Brin (1998) have been used in keyword extraction and extractive summarization (Mihalcea and Tarau 2004), word sense disambiguation (Agirre and Soroa 2009) and abstractive summarization (Ganesan, Zhai and Han 2010). In this paper we use it as the first step in a two-step unsupervised dependency-parsing procedure to rank words by centrality.

The parser assigns a dependency structure to a sequence of words in two stages. In the first step, it ranks the words in the input sentence using iterative graph-based ranking. In the second step, it constructs a tree from the ranked list of words using a simple $\mathcal{O}(n^2)$ deterministic parsing algorithm. This section describes the graph construction step in some detail and briefly describes the used iterative graph-based ranking algorithm.

2.1 Edges

The graph over the words in the input sentence is constructed by adding directed edges between the word nodes. The edges are not weighted, but multiple edges between nodes will more likely make transitions between these nodes in iterative graph-based ranking.

Some of the edge assignments discussed below may seem rather heuristic. The edge template was developed on development data from the English Penn-III treebank (Marcus, Marcinkiewicz and Santorini 1993). Our edge selection was incremental considering first an extended set of candidate edges with arbitrary parameters and then considering each edge type at a time. If the edge type was helpful, we optimized any possible parameters (say context windows) and went on to the next edge type, otherwise we disregarded it. Following Gillenwater *et al.* (2010), we apply the best setting of English to all other languages.

Short edges. Eisner and Smith (2005) used soft constraints to favor short dependencies. Guided by a similar intuition, we add links between all words and their neighbors. This makes probability mass flow from central words to their neighboring words. It also guarantees that the final graph is connected (which will be important, since we use iterative graph-based ranking without random restarts).

Function words. We use a keyword extraction algorithm without stop word lists to extract function or non-content words. The algorithm is a crude simplification of TextRank (Mihalcea and Tarau 2004) that does not rely on linguistic resources so that we can easily apply it to low-resource languages. Since we do not use stop word lists, highly ranked words will typically be function words.[2] For the fifty most highly ranked words, we add additional links from their neighboring words. This will add additional probability mass to the function words. This is relevant to capture structures, such as prepositional phrases, where the function words take content words as complements.

Morphological inequality. If two words, w_i, w_j, have different prefixes or suffixes, i.e. the first two or last three letters, we add an edge between them.

Verb edges. All words are attached to all words with a POS tag beginning with 'V. . . '. This captures the fact that verbs are syntactically and semantically important in all languages that we know. We then add links from all verbs to all nouns. While this obviously gives semantics to the POS of verbs and nouns, we note that this is very similar in spirit to the approach to unsupervised dependency parsing taken in Naseem *et al.* (2010), to which we return later on, except much simpler. In a way, the verb edges encode the universal dependency rules $\text{ROOT} \longrightarrow \text{VERB}$ and $\text{VERB} \longrightarrow \text{NOUN}$ (see Figure 5).

The verb edges are the only edges that rely on POS. So these edges are left out when we evaluate in Section 7 an unsupervised dependency parser that is not at all informed by POS.

[2] Function words are highly ranked by keyword extraction algorithms that do not use stop word lists because of their frequency.

2.2 Ranking

Given the constructed graph, we rank the nodes using the algorithm of Page and Brin (1998), also known as PageRank. The PageRank algorithm proceeds as a random walk on the graph. The analogy of the determined web surfer tirelessly clicking hyperlinks on web pages comes from the original paper introducing this algorithm (Brin and Page 1998). A different analogy would be that of a no less determined drunkard touring the bars and pubs of his home town. The drunkard dutifully writes down the number of times he has visited different places. His route is not planned ahead. Each bar keeps a list of suggested places to visit, and whenever he decides to leave a bar, he picks a destination off of that bar's list at random. The PageRank value of a bar is the number of visits the bar gets out of the total number of bar visits. In most applications, the web surfer restarts with some probability; in our analogy, the drunkard experiences a blackout. The damping factor is the probability of not randomly restarting. However, in our experiments the damping factor is set to 1.0, meaning there are no random restarts. Since our graphs are connected, we will never get trapped anywhere in our word graphs.

More formally, the input to the PageRank algorithm is any directed graph $G = \langle E, V \rangle$ and the output is an assignment $PR : V \rightarrow \mathbb{R}$ of a score, also referred to as PageRank, to each vertex in the graph such that all scores sum to 1. A simplified version of PageRank can be defined recursively as follows:

$$PR(v) = \Sigma_{w \in B_v} \frac{PR(w)}{L(w)}$$

where B_v is the set of vertices such that $(w, v) \in E$, and $L(w)$ is the number of outgoing links from w, i.e. $|\{(u, u')|(u, u') \in E, u = w\}|$. Page and Brin (1998), as already mentioned, introduces a damping factor (typically 0.85) to reflect that fact that Internet users do not continue crawling web sites forever, but restart, returning to random web sites. This influences centrality judgments and therefore should be reflected in the probability assignment. Since there is no obvious analogue of this in our case, we simplify the PageRank algorithm and do not incorporate random restarts (or equivalently, set the damping factor to 1.0).

Note that although our graphs are non-weighted and directed, like a graph of web pages and hyperlinks (and unlike, for example, the text graphs in Mihalcea and Tarau 2004), several pairs of nodes may be connected by multiple edges, making a transition between them more probable. Multiple edges provide a coarse weighting of the underlying minimal graph.

2.3 Example

In Figure 2, we see an example graph of word nodes, represented as a matrix, and a derived dependency structure.[3] We see that there are three edges from *The* to

[3] The dependency structure in Figure 1 contains dependency labels such as 'SBJ' and 'ROOT'. These are just included for readability. We follow the literature on unsupervised dependency parsing and focus only on unlabeled dependency parsing. The red arc indicates that our

from/to	The	finger-pointing	has	already	begun	.
The	0	3	2	2	3	2
finger-pointing	3	0	5	2	3	2
has	2	4	0	3	3	2
already	2	2	5	0	3	2
begun	2	3	3	3	0	3
.	2	2	3	2	4	0
PR(%)	13.4	17.4	21.2	15.1	19.3	13.6

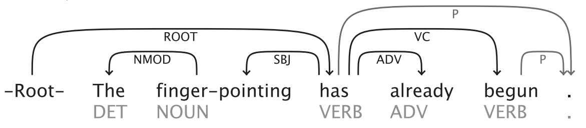

Fig. 2. Graph, pagerank (PR) and predicted dependency structure for sentence 7 in PTB-III section 23.

finger-pointing and two from *The* to *has*, for example. We then compute the PageRank of each node using the algorithm described in Page and Brin (1998); see also Figure 2. The PageRank values rank the nodes or the words. In Section 3, we describe a method for building a dependency tree from the ranking of nodes. This method will produce the correct analysis of this sentence except for punctuation (see Figure 2). This is because the PageRank scores reflect syntactic superiority; the root of the sentence typically has the highest rank, and the least important nodes are ranked lowly.

3 From ranking of nodes to dependency trees

Our non-projective parsing algorithm makes reference to the universal dependency rules introduced in Naseem *et al.* (2010). The rules are in fact what make our parsing algorithm non-projective. In order to apply universal dependency rules to all languages, we replace POS tags with the corresponding tags in the twelve-tag 'universal' POS tagset presented in Petrov, Das and McDonald (2011); see the tags that decorate the nodes in Figure 2. However, we first introduce our parsing algorithm without universal rules. The algorithm described in the first subsection thus only produces projective dependency trees.

3.1 Without universal rules

Consider again the example given in Figure 2. Once we have ranked the nodes in our dependency structure, we build a dependency structure from the ranking using the parsing algorithm in Figure 3. The input of the graph is a list of ranked words $\pi = \langle n_1, \ldots, n_m \rangle$, where each node n_i corresponds to a sentence position $n_{pr2ind(i)}$ decorated by a word form $w_{pr2ind(i)}$, where $pr2ind : \{1, \ldots, m\} \rightarrow \{1, \ldots, m\}$ is a

parser will end up predicting the wrong syntactic head for the punctuation sign, namely the embedded verb rather than the main verb.

```
1: $\pi = \langle n_1, \dots, n_m \rangle$ # the ranking of nodes
2: $H = \{n_0\}$ # possible heads
3: $D = \emptyset$ # dependency structure
4: $pr2ind : \{1, \dots, m\} \rightarrow \{1, \dots, m\}$ # a mapping from rank to sentence position
5: for $1 \leq i \leq m$ do
6:     if $|H|$=1 then
7:         $c = 0$ # used to ensure single-rootedness
8:     else
9:         $c = 1$
10:    end if
11:    $n_{j'} = \arg\min_{n_j \in H[c:]} |pr2ind(i) - pr2ind(j)|$ # select head of $w_{pr2ind(i)}$
12:    $H = n_i \cup H$ # make $n_i$ a possible head
13:    $D = \{(w_{pr2ind(i)} \leftarrow w_{pr2ind(j')})\} \cup D$ # add new edge to $D$
14: end for
15: return $D$
```

Fig. 3. Parsing algorithm without universal dependency rules.

mapping from rank to sentence position. Note that $H[c :]$ is list slice notation used to indicate that when $c = 1$, the first element on the list H is ignored.

The interesting step in the algorithm is the head selection step. Each word is assigned a syntactic head taken from all the words that were already assigned heads. Of these words, we simply select the closest possible head. While this is not stated explicitly in the pseudocode, if two possible heads are equally close, we select the one with the highest PageRank.

Our parsing algorithm runs in $\mathcal{O}(n^2)$, since it runs over the ranked words in a single pass considering only previously stored words as possible heads, and guarantees connectivity, acyclicity and single-rootedness, and thus produces well-formed non-projective dependency trees. To see this, remember that well-formed dependency trees are such that all nodes but the artificial root nodes have a single incoming edge. This follows immediately from the fact that each node is assigned a head (line 11). Furthermore, the dependency tree must be acyclic. This follows immediately from the fact that a word can only attach to a word with higher rank than itself. Connectivity follows from the fact that there is an artificial root node and all words are attached to this node or to nodes dominated by the root node. Finally, we ensure single-rootedness by explicitly disregarding the root node once we have attached the node with the highest rank to it (lines 6–7). Our parsing algorithm also guarantees projectivity, however, since we always select the closest possible head. Say we have a dependency from w_2 to its dependent w_5, where w_i is the word in the ith position. If we were to select a head for w_3 now, it would not be possible to select w_1 or w_6, since w_2 is already in the set of possible heads H that we have to consider.

The derivation of a simple Penn-III sentence ‘The market crumbled .’ is given in Figure 4.

3.2 With universal rules

Naseem *et al.* (2010) introduced universal dependency rules as linguistic priors or soft constraints in unsupervised dependency parsing. The intuition is that we

π	H	D
⟨ crumbled, market, the, . ⟩	⟨ ROOT ⟩	$\emptyset$
⟨ market, the, . ⟩	⟨ crumbled ⟩	{ ROOT → crumbled }
⟨ the, . ⟩	⟨ crumbled, market ⟩	{ ROOT → crumbled, crumbled → market }
⟨ . ⟩	⟨ crumbled, market, the ⟩	{ ROOT → crumbled, crumbled → market, market → the }
⟨ ⟩	⟨ crumbled, market, the, . ⟩	{ ROOT → crumbled, crumbled → market, market → the, crumbled → . }

Fig. 4. Example derivation.

ROOT⟶VERB	
VERB⟶VERB VERB⟶NOUN VERB⟶ADV VERB⟶ADP [new] VERB⟶CONJ [new] VERB⟶DET [new] VERB⟶NUM [new] VERB⟶ADJ [new]	NOUN⟶ADJ NOUN⟶DET NOUN⟶NOUN NOUN⟶NUM
ADP⟶NOUN	ADP⟶ADV

Fig. 5. Universal dependency rules (Naseem *et al.*, 2010) with our revisions.

should be able to make use of what we know languages have in common. The universal dependency rules proposed in their paper are presented in Figure 5 with few revisions.

Converting the universal dependency rules to the tagset proposed by Petrov *et al.* (2011) caused one rule to fall out, since the tagset does not have a category for AUX; see the discussion in Petrov *et al.* (2011). In addition, the rule VERB⟶PRON, encoding a preference for pronouns to be headed by verbs, was removed during development on English data. Finally, we added rules for the remaining POS (CONJ (conjunctions), ADP (adpositions) and NUM (numerals)), and we also added VERB⟶DET, since some treebanks, such as the Danish treebank, have determiners as heads of nouns phrases.

We will now modify the above parsing algorithm in the following simple way: When selecting a head for any word on the ranked list, we first see if we can find a head such that the head-dependent pair is an instantiation of one of our universal dependency rules. More precisely, we select the closest head that enables us to instantiate a rule. Only if that is not possible, say if the current word belongs to the category NUM and H = {VERB, ADP}, we will select the nearest possible head in H, as we did before. For all the above reasons, our new parsing algorithm produces well-formed non-projective dependency trees, but now we no longer have a guarantee that structures are also projective. In particular, we can have crossing edges that are instantiations of universal dependency rules.

Note on expressivity: The introduction of preference for universal dependency rules allows our parser to produce non-projective dependency structures with arbitrary gap-degree (Maier and Søgaard 2008), meaning that any number of words occurring between a head h and one of its dependents can be dominated by a node dominating the head h. It is nevertheless not possible to generate all possible non-projective dependency trees. First of all, given a sentence with POS information, any crossing edge has to be licensed by one of our universal dependency rules. So the number of crossing edges can be ruled out in advance. Moreover, we cannot generate all tree structures either, since we have a fixed bound on the number of POS tags (i.e. twelve). Consequently, we cannot have a dependency with gap-degree k with $k > 12$ without some of the gap words sharing syntactic heads.

4 Experiments

We use exactly the same experimental set-up as Gillenwater *et al.* (2010). The edge model was developed on development data from the English Penn-III treebank (Marcus *et al.* 1993), and we evaluate on Section 23 of the English treebank and the test sections of the remaining eleven treebanks, which were used in the CoNLL-X Shared Task (Buchholz and Marsi 2006). Gillenwater *et al.* (2010) for some reason did not evaluate on the Arabic and Chinese treebanks also used in the shared task. We also follow Gillenwater *et al.* (2010) in only evaluating our parser on sentences of at most ten non-punctuation words and in reporting unlabeled attachment scores excluding punctuation. This is standard in unsupervised dependency parsing, but for completeness, we also report scores on full-length sentences.

We first consider baselines and upper bounds on performance. The baseline that is relevant to evaluate the algorithm that constructs the ranking of words is to use a random ordering of the input words instead, or simply use the linear order they originally appeared in. The lines with bullet points and circles (rand, randUR, lin and linUR) in Figure 6 present these baselines. The upper bound on performance is using an ordering of input words chosen by an oracle. Searching all possible orderings is of course not feasible in general, but instead we use orders that are consistent with the partial orders defined by the gold dependency trees. This does not give us the actual upper bound, but a very good approximation. The lines with crosses (tree and treeUR) in Figure 6 are the upper bounds on performance.

It is interesting to note that the our upper bound (treeUR), which averages around 75%, is only a couple of percentage points lower than the upper bound on DMV-based approaches (Cohen, Das and Smith 2011a). In that sense, our two-stage approach is as promising as the original model proposed by Klein and Manning (2004).

We present results on sentences of at most ten non-punctuation words in Figure 7 with usUR referring to our results. DMV PR-AS 140 is the best performing model

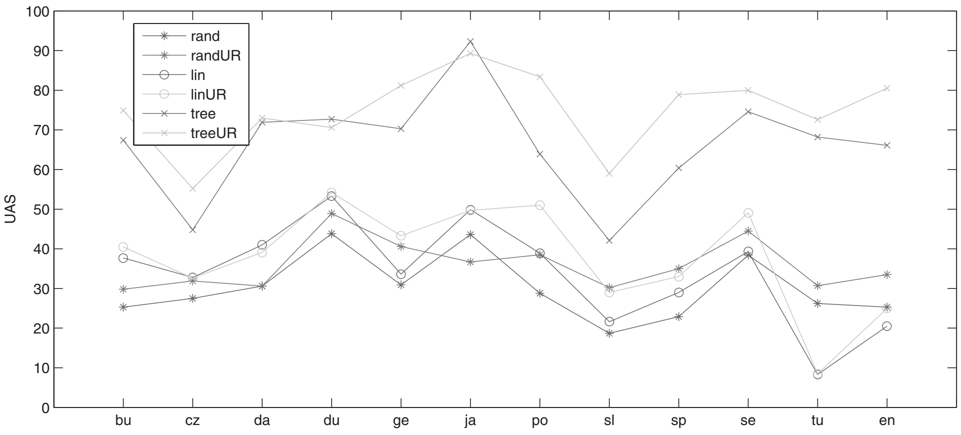

Fig. 6. Upper and lower bounds: rand(UR) is parsing with randomly ranked words, with or without universal rules; lin(UR) is parsing with linear order and tree(UR) is parsing with order consistent with partial orders defined by the gold dependency trees.

	DMV PR-AS 140	E-DMV PR-AS 140	us	usUR	Brod10	Nase10
Bulgarian	54.0	**59.8**	46.4	51.4	-	-
Czech	32.0	**54.6**	44.0	45.7	-	-
Danish	42.4	47.2	50.8	51.4	41.9	**51.9**
Dutch	37.9	**46.6**	39.7	38.3	35.3	-
English	61.9	64.4	52.6	59.9	39.3	**71.9**
German	39.6	35.7	48.7	**57.6**	-	-
Japanese	60.2	59.4	**62.3**	60.9	-	-
Portuguese	47.8	49.5	47.0	54.6	-	**71.5**
Slovene	50.3	**51.2**	38.3	39.7	-	50.9
Spanish	62.4	57.9	47.0	52.6	-	**67.2**
Swedish	38.7	41.4	52.3	60.5	-	**62.1**
Turkish	53.4	**56.9**	50.3	54.0	-	-
AV	48.4	**52.2**	48.3	**52.2**	-	-

Fig. 7. Unlabeled attachment scores (in %) on sentences of length of at most 10 ignoring punctuation.

in Gillenwater *et al.* (2010) using posterior regularization and DMV, and E-DMV PR-AS 140 is the best performing model using posterior regularization and the extended model from Headden *et al.* (2009).

Our model is on average considerably better than DMV PR-AS 140 and as good as E-DMV PR-AS 140. It is better than E-DMV PR-AS 140 on five languages with absolute improvements of as much as 15%. However, our model is consistently worse than Naseem *et al.* (2010). On the other hand, we believe that the results are very promising for a radically new model, and they are considerably better than the results presented in Brody (2010), who also presents a model that is very different from the DMV-inspired models. Note also that universal dependency rules, in general, help considerably, e.g. for English and German, but in a few cases

	rand	randR	us	usUR	Spit11
Bulgarian	21.3	27.6	37.1	39.3	**40.5**
Czech	20.9	24.0	35.8	36.9	**37.8**
Danish	22.2	25.1	36.6	**37.8**	37.1
Dutch	32.5	31.7	37.6	**37.9**	14.0
English	19.3	26.8	40.3	43.0	-
German	21.8	30.9	34.6	**41.2**	28.6
Japanese	36.2	39.9	**41.4**	39.5	27.5
Portuguese	21.2	30.1	45.0	**46.7**	33.5
Slovene	16.9	23.4	33.0	**33.6**	31.2
Spanish	18.6	26.3	38.0	**39.0**	32.3
Swedish	25.9	34.1	43.8	**48.5**	46.4
Turkish	24.4	27.5	41.9	**42.6**	40.9
AV	23.8	29.1	38.6	**40.2**	33.6

Fig. 8. Unlabeled attachment scores (in %) on full-length sentences. Averages disregard English, since Spitkovsky *et al.* (2011) do not report results for Section 23 of Penn-III.

hurt (Dutch and Japanese). Interestingly, the average improvement obtained by using universal dependency rules is about the same as the improvement obtained in Gillenwater *et al.* (2010) moving to the extended model.

We present results on full-length sentences in Figure 8. None of the above-mentioned authors present results for full-length sentences, so here we compare ourselves to the best model in Spitkovsky, Alshawi and Jurafsky (2011b) instead, which also uses the extended model from Headden *et al.* (2009). Note that our parser performs better than Spitkovsky *et al.*'s (2011b) parser on average, even without universal dependency rules.

5 Error analysis

In our error analysis, we focus on the results for German and Turkish. We first compare error distributions relative to POS of our German parsers with and without universal dependency rules. The numbers below are accuracies attaching dependents with a particular POS:

acc	us	usUR
NOUN	28%	34%
VERB	49%	49%
DET	44%	48%
ADP	17%	45%
ADJ	41%	48%

The universal dependency rules do not seem to help us attaching verbs, but note also that the parser without rules is already very good at attaching verbs. The rules considerably improve attachment of adpositions, but improvements are also observed with nouns, determiners and adjectives.

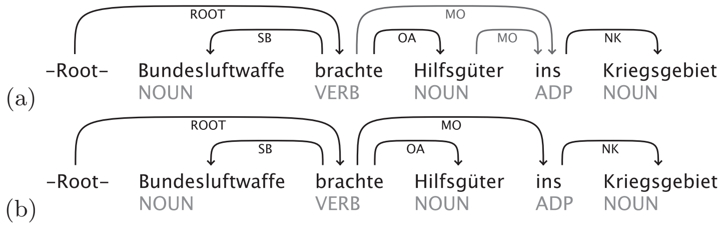
ROOT
SB
MO
OA
MO
NK
-Root- Bundesluftwaffe brachte Hilfsgüter ins Kriegsgebiet
NOUN VERB NOUN ADP NOUN
(a)
ROOT
SB
MO
OA
NK
-Root- Bundesluftwaffe brachte Hilfsgüter ins Kriegsgebiet
NOUN VERB NOUN ADP NOUN
(b)

Fig. 9. Predicted dependency structures for a sentence in the German test section with and without dependency rules. Red arcs indicate wrong predictions.

See Figure 9 for an example of where universal dependency rules help us attach correctly an adposition. The verb and direct object words have higher rank than the adposition, and the adposition is forced to attach to the direct object unless there is a dependency rule licensing an attachment further away. However, we do have a universal rule that licenses dependencies from verbs to adpositions (VERB⟶ADP), which is why we predict correct analysis in Figure 9(b). Figure 9(a) is the prediction made without universal dependency rules.

The rules allow us to attach dependents to candidate heads further away, so we predict more long dependencies. This also leads to a considerable absolute improvement of 15% in *f*-score for long dependencies (to words at least seven positions away).

f-score	us	usUR
to_root	57.3%	57.3%
1	65.7%	69.4%
2	30.3%	45.6%
3–6	24.0%	45.9%
7–	25.3%	40.6%

Improvements with long dependencies are primarily due to better recall.

For German, improvements were observed with nouns and adjectives, while the accuracy of attaching verbs did not improve by using universal dependency rules. A similar pattern is observed in the Turkish data, but perhaps less dramatically so that

acc	us	usUR
NOUN	39%	40%
VERB	55%	55%
ADJ	36%	38%
ADV	31%	36%
PRON	43%	43%

Increase in f-score is again higher with longer dependencies:

f-score	us	usUR
to_root	67.2%	67.2%
1	67.9%	68.1%
2	23.2%	32.4%
3–6	24.0%	29.3%
7–	10.5%	14.2%

The parsers predict fewer long dependencies for Turkish than for German ($< 3\%$ of the dependencies cover more than seven words); precision is reasonable (25% without universal dependency rules; 28.5% with rules), but recall is very low.

6 Unsupervised dependency parsing without POS

In the Introduction, we mentioned that POS, which can be seen as partial annotation, makes unsupervised dependency parsing considerably easier. A simple way for us to build an unsupervised dependency parser that does not rely on information about POS at all is by removing verb edges in the graph construction step and not use the universal dependency rules. We call this parser ‘usnP’ and compare it with ‘us’ and ‘usUR’ and our baselines with random ranking (on full-length sentences), leaving out English for comparability with Spitkovsky *et al.* (2011b):

	rand	randUR	us	usUR	usnP	Spit11
Bulgarian	21.3	27.6	37.1	39.3	33.3	**40.5**
Czech	20.9	24.0	35.8	36.9	32.2	**37.8**
Danish	22.2	25.1	36.6	37.8	**41.7**	37.1
Dutch	32.5	31.7	37.6	**37.9**	29.9	14.0
German	21.8	30.9	34.6	**41.2**	31.2	28.6
Japanese	36.2	39.9	41.4	39.5	**43.3**	27.5
Portuguese	21.2	30.1	45.0	**46.7**	33.5	33.5
Slovene	16.9	23.4	33.0	**33.6**	31.3	31.2
Spanish	18.6	26.3	38.0	**39.0**	29.4	32.3
Swedish	25.9	34.1	43.8	**48.5**	38.7	46.4
Turkish	24.4	27.5	41.9	**42.6**	38.5	40.9
AV	23.8	29.1	38.6	**40.2**	34.8	33.6

Note that our results with POS are on average better than those reported by Spitkovsky *et al.* (2011b) and considerably better than our baselines with random order. Somewhat surprisingly, results are better without POS for Danish and Japanese.

7 Conclusion

We have presented a new approach to unsupervised dependency parsing. The key idea is that a dependency structure also expresses centrality or saliency, so by direct modeling of centrality we obtain information that can be used to build dependency structures. Our unsupervised dependency parser thus works in two stages: it first uses iterative graph-based ranking to rank words in terms of centrality, and then constructs a dependency tree from the ranking. Our parser was shown to be competitive to state-of-the-art unsupervised dependency parsers. Finally, we showed that our parser also obtains promising results in the absence of information about POS.

References

Agirre, E., and Soroa, A. 2009. Personalizing pagerank for word sense disambiguation. In *Proceedings of the European Chapter of the Association for Computational Linguistics*, Athens, Greece, pp. 33–41.

Brin, S., and Page, L. 1998. The anatomy of a large-scale hypertextual web search engine. In *Proceedings of the International Web Conference*, Brisbane, Australia, pp. 107–17.

Brody, S. 2010. It depends on the translation: unsupervised dependency parsing via word alignment. In *Proceedings of the Empirical Methods in Natural Language Processing*, Boston, MA, pp. 1214–22.

Buchholz, S., and Marsi, E. 2006. CoNLL-X shared task on multilingual dependency parsing. In *Proceedings of the Computational Natural Language Learning*, New York City, NY, USA, pp. 149–64.

Cohen, S., Das, D., and Smith, N. 2011a. Unsupervised structure prediction with non-parallel multilingual guidance. In *Proceedings of the Empirical Methods in Natural Language Processing*, Edinburgh, Scotland, pp. 50–61.

Cohen, S., Gimpel, K., and Smith, N. 2008. Logistic normal priors for unsupervised probabilistic grammar induction. In *Proceedings of the Twenty-Second Annual Conference on Neural Information Processing Systems (NIPS)*, Vancouver, BC, Canada, December 8–11, pp. 321–8.

Cohen, S., Rodriguez, C., and Satta, G. 2011b. Exact inference for generative probabilistic non-projective dependency parsing. In *Proceedings of Empirical Methods in Natural Language Processing*, Edinburgh, Scotland, UK, pp. 1234–45.

Druck, G., Mann, G., and McCallum, A. 2009. Semi-supervised learning of dependency parsers using generalized expectation criteria. In *Proceedings of Association for Computational Linguistics*, Singapore, pp. 360–8.

Eisner, J., and Smith, N. A. 2005. Parsing with soft and hard constraints on dependency length. In *Proceedings of International Conference on Parsing Technologies*, Vancouver, BC, Canada, pp. 30–41.

Ganesan, K., Zhai, C., and Han, J. 2010. Opinosis: a graph-based approach to abstractive summarization of highly redudant opinions. In *Proceedings of International Conference of Computational Linguistics*, Beijing, China, pp. 340–8.

Gillenwater, J., Ganchev, K., Graca, J., Pereira, F., and Taskar, B. 2010. Sparsity in dependency grammar induction. In *Proceedings of Association for Computational Linguistics*, Uppsala, Sweden, pp. 194–9.

Headden, W., Johnson, M., and McClosky, D. 2009. Improving unsupervised dependency parsing with richer contexts and smoothing. In *Proceedings of North American Chapter of the Association for Computational Linguistics*, Boulder, CO, USA, pp. 101–9.

Klein, D., and Manning, C. 2004. Corpus-based induction of syntactic structure: models of dependency and constituency. In: *Proceedings of Association for Computational Linguistics*, Barcelona, Spain, pp. 478–85.

Maier, W., and Søgaard, A. 2008. Treebanks and mild context-sensitivity. In *Formal Grammar*, Hamburg, Germany, pp. 61–76.

Marcus, M., Marcinkiewicz, M., and Santorini, B. 1993. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics* **19**(2): 313–30.

McDonald, R., Pereira, F., Ribarov, and K., Hajič, J. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of Empirical Methods in Natural Language Processing*, Vancouver, BC, Canada, pp. 523–30.

McDonald, R., Petrov, S., and Hall, K. 2011. Multi-source transfer of delexicalized dependency parsers. In *Proceedings of Empirical Methods in Natural Language Processing*, Edinburgh, Scotland, UK, pp. 62–72.

McDonald, R., and Satta, G. 2007. On the complexity of non-projective data-driven dependency parsing. In *Proceedings of International Conference on Parsing Technologies*, Prague, Czech Republic, pp. 121–32.

Mihalcea, R., and Tarau, P. 2004. Textrank: bringing order into texts. In *Proceedings of Empirical Methods in Natural Language Processing*, Barcelona, Spain, pp. 404–11.

Naseem, T., Chen, H., Barzilay, R., and Johnson, M. 2010. Using universal linguistic knowledge to guide grammar induction. In *Proceedings of Empirical Methods in Natural Language Processing*, Boston, MA, USA, pp. 1234–44.

Nivre, J. 2009. Non-projective dependency parsing in expected linear time. In *Proceedings of Association for Computational Linguistics*, Singapore, pp. 351–9.

Petrov, S., Das, D., and McDonald, R. 2011. A universal part-of-speech tagset. *CoRR* abs/1104.2086.

Schwartz, R., Abend, O., Reichart, R., and Rappoport, A. 2011. Neutralizing linguistically problematic annotations in unsupervised dependency parsing evaluation. In *Proceedings of Association for Computational Linguistics*, Portland, OR, USA, pp. 663–72.

Seginer, Y. 2007. Fast unsupervised incremental parsing. In *Proceedings of Association for Computational Linguistics*, Prague, Czech Republic, pp. 384–91.

Smith, N., and Eisner, J. 2005. Contrastive estimation: training log-linear models on unlabeled data. In *Proceedings of Association for Computational Linguistics*, Ann Arbor, MI, pp. 354–62.

Smith, N., and Eisner, J. 2006. Annealing structural bias in multilingual weighted grammar induction. In *Proceedings of Association for Computational Linguistics*, Sydney, Australia, pp. 569–76.

Smith, D., and Eisner, J. 2009. Parser adaptation and projection with quasi-synchronous grammar features. In *Proceedings of Empirical Methods in Natural Language Processing*, Singapore, pp. 822–31.

Søgaard, A. 2011. Data point selection for cross-language adaptation of dependency parsers. In *Proceedings of Association for Computational Linguistics*, Portland, OR, USA, pp. 682–6.

Spitkovsky, V., Alshawi, H., Chang, A., and Jurafsky, D. 2011a. Unsupervised dependency parsing without gold part-of-speech tags. In *Proceedings of Empirical Methods in Natural Language Processing*, Edinburgh, Scotland, UK, pp. 1281–90.

Spitkovsky, V., Alshawi, H., and Jurafsky, D. 2009. Baby steps: how "less is more" in unsupervised dependency parsing. In *Proceedings of NIPS Workshop on Grammar Induction, Representation of Language and Language Learning*, Whistler, BC, Canada, pp. 1–9.

Spitkovsky, V., Alshawi, H., and Jurafsky, D. 2011b. Lateen EM: unsupervised training with multiple objectives applied to dependency grammar induction. In *Proceedings of Empirical Methods in Natural Language Processing*, Edinburgh, Scotland, UK, pp. 1269–80.

Spreyer, K., and Kuhn, J. 2009. Data-driven dependency parsing of new languages using incomplete and noisy training data. In *Proceedings of Computational Natural Language Learning*, Boulder, CO, USA, pp. 12–20.