From ranked words to dependency trees: two-stage unsupervised non-projective dependency parsing

Anders Søgaard
Center for Language Technology
University of Copenhagen
soegaard@hum.ku.dk

Abstract

Usually unsupervised dependency parsing tries to optimize the probability of a corpus by modifying the dependency model that was presumably used to generate the corpus. In this article 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 model the partial order directly and derive dependency trees from this order. The result is an approach to unsupervised dependency parsing that is very different from standard ones in that it requires no training data. Each sentence induces a model from which the parse is read off. Our approach is evaluated on data from 12 different languages. Two scenarios are considered: a scenario in which information about part-of-speech is available, and a scenario in which parsing relies only on word forms and distributional clusters. Our approach is competitive to state-of-the-art in both scenarios.

1 Introduction

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 on English (Marcus et al., 1993) of about 15% in six years (Klein and Manning, 2004; Spitkovsky et al., 2010), and better 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 et al., 2009) or using cross-language adaptation (Smith and Eisner, 2009; Spreyer et al., 2010).

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), 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 as 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 with a few exceptions only considers the sentence in question. It *does* use a larger corpus to induce distributional clusters and a ranking of key words in terms of frequency and centrality, but this is computationally efficient and is only indirectly related to the subsequent assignment of dependency structures to sentences. The obvious advantage of not relying on training data is that we do not have to

Proceedings of the TextGraphs-6 Workshop, pages 60–68,
Portland, Oregon, USA, 19-24 June 2011.

worry about whether the test data reflects the same distribution as the target data (domain adaptation), 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 and distributional clusters, constructs a directed acyclic graph from the nodes in $\mathcal{O}(n^2)$, and ranks the nodes using iterative graph-based ranking (Page and Brin, 1998). Subsequently, it constructs a tree from the ranked list of words using a simple $\mathcal{O}(n \log n)$ parsing algorithm.

Our parser is evaluated on the selection of 12 dependency treebanks also used in Gillenwater et al. (2010). We consider two cases: parsing raw text and parsing text with information about POS.

Strictly unsupervised dependency parsing is of course a more difficult problem than unsupervised dependency parsing of manually annotated POS sequences. Nevertheless our *strictly* unsupervised parser, which only sees word forms, performs significantly better than structural baselines, and it outperforms the standard POS-informed DMV-EM model (Klein and Manning, 2004) on 3/12 languages. The full parser, which sees manually annotated text, is competitive to state-of-the-art models such as E-DMV PR AS 140 (Gillenwater et al., 2010).[1]

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. Each vertex has a single incoming edge, possibly except one called the root of the tree. In this work and in most other work in dependency parsing, we introduce an artificial root node so that all vertices decorated by word forms have an incoming edge.

A dependency structure such as the one in Figure 1 is thus a tree 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 $w_i \rightarrow w_j$. 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. 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.:

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

A dependency structure is also *acyclic*, i.e.:

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

Finally, a dependency structure is *single-headed*, i.e.:

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

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 wrt. the 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 work in unsupervised dependency parsing has 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. Non-projective parsing algorithms for supervised dependency parsing have, for example, been presented in McDonald et al. (2005) and Nivre (2009).

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

[1] Naseem et al. (2010) obtain slightly better results, but only evaluate on six languages. They made their code public, though: http://groups.csail.mit.edu/rbg/code/dependency/

POS-tagged English above a right-branching baseline.

DMV is a generative model in which the sentence root is generated and then each head recursively generates its left and right dependents. For each $s_i \in \mathrm{s}$, 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 given h and d. Klein and Manning (2004) use expectation maximization (EM) to estimate probabilities with manually tuned linguistically-biased priors.

Smith and Eisner (2005) use contrastive estimation instead of EM, while Smith and Eisner (2006) use structural annealing which penalizes long-distance dependencies initially, gradually weakening the penalty during training. Cohen et al. (2008) use Bayesian priors (Dirichlet and Logistic Normal) with DMV. All of the above approaches to unsupervised dependency parsing build on the linguistically-biased priors introduced by Klein and Manning (2004).

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 11 treebanks from the CoNLL 2006 Shared Task and the Penn-III treebank and achieve state-of-the-art performance.

An exception to using linguistically-biased priors is Spitkovsky et al. (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 are below 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 ones. Semantically, the nodes near the root typically express the main predicate and its arguments. Iterative graph-based ranking (Page and Brin, 1998) was first used to rank webpages according to their centrality, but the technique has found wide application in natural language processing. Variations of the algorithm presented in 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 et al., 2010). In this paper, we use it as the first step in a two-step unsupervised dependency parsing procedure.

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 and distributional clusters, constructs a directed acyclic graph from the nodes in $\mathcal{O}(n^2)$, and ranks the nodes using iterative graph-based ranking. Subsequently, it constructs a tree from the ranked list of words using a simple $\mathcal{O}(n \log n)$ parsing algorithm. This section describes the graph construction step in some detail and briefly describes the iterative graph-based ranking algorithm used.

The first step, however, is assigning distributional clusters to the words in the sentence. We use a hierarchical clustering algorithm to induce 500 clusters from the treebanks using publicly available software.[2] This procedure is quadratic in the number of clusters, but linear in the size of the corpus. The cluster names are bitvectors (see Figure 1).

2.1 Edges

The text graph is now constructed by adding different kinds of directed edges between nodes. The edges are not weighted, but multiple edges between nodes will make transitions between these nodes in

[2] http://www.cs.berkeley.edu/∼pliang/software/brown-cluster-1.2.zip

iterative graph-based ranking more likely. The different kinds of edges play the same role in our model as the rule templates in the DMV model, and they are motivated below.

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 et al., 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.[3] Following data set et al. (2010), we apply the best setting for English to all other languages.

Vine edges. Eisner and Smith (2005) motivate a vine parsing approach to supervised dependency parsing arguing that language users have a strong preference for short dependencies. Reflecting preference for short dependencies, we first add links between all words and their neighbors and neighbors' neighbors. This also guarantees that the final graph is connected.

Keywords and closed class words. We use a keyword extraction algorithm without stop word lists to extract non-content words and the most important content words, typically nouns. 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 non-content words, followed by what is more commonly thought of as keywords. Immediate neighbors to top-100 words are linked to these words. The idea is that non-content words may take neighboring words as arguments, but dependencies are typically very local. The genuine keywords, ranked 100–1000, may be heads of dependents further away, and we therefore add edges between these words w_i and their neighboring words w_j if $|i - j| \leq 4$.

Head-initial/head-final. It is standard in unsupervised dependency parsing to compare against a structural baseline; either left-attach, i.e. all words attach to their left neighbor, or right-attach. Which structural baseline is used depends on the language in question. It is thus assumed that we know enough about the language to know what structural baseline performs best. It is therefore safe to incorporate this knowledge in our unsupervised parsers; our parsers are still as "unsupervised" as our baselines. If a language has a strong left-attach baseline, like Bulgarian, the first word in the sentence is likely to be very central for reasons of economy of processing. The language is likely to be head-initial. On the other hand, if a language has a strong right-attach baseline, like Turkish, the last word is likely to be central. The language is likely to be head-final. Some languages like Slovene have strong ($< 20\%$) left-attach *and* right-attach baselines, however. We incorporate the knowledge that a language has a strong left-attach or right-attach baseline if more than one third of the dependencies are attachments to a immediate left, resp. right, neighbor. Specifically, we add edges from all nodes to the first element in the sentence if a language has a strong left-attach baseline; and from all edges to the last (non-punctuation) element in the sentence if a language has a strong right-attach baseline.

Word inequality. An edge is added between two words if they have different word forms. It is not very likely that a dependent and a head have the same word form.

Cluster equality. An edge is added between two words if they are neighbors or neighbors' neighbors and belong to the same clusters. If so, the two words may be conjoined.

Morphological inequality. If two words w_i, w_j in the same context ($|i - j| \leq 4$) share prefix or suffix, i.e. the first or last three letters, we add an edge between them.

2.2 Edges using POS

Verb edges. All words are attached to all words with a POS tag beginning with "V...".

Finally, when we have access to POS information, we do not rely on vine edges besides left-attach, and we do not rely on keyword edges or suffix edges either.

[3] The search was simplified considerably. For example, we only considered symmetric context windows, where left context length equals length of right context, and we binned this length considering only values 1, 2, 4, 8 and all.

2.3 Ranking

Given the constructed graph we rank the nodes using the algorithm in Page and Brin (1998), also known as PageRank. The input to this 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:

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

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. $|\{(u, u')|(u, u') \in E, u = w\}|$. In addition to this, Page and Brin (1998) introduces a so-called damping factor 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 damping (or equivalent, set the damping factor to 1.0).

Note, by the way, that although our graphs are non-weighted and directed, like a graph of web pages and hyperlinks (and unlike the text graphs in Mihalcea and Tarau (2004), for example), 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.

2.4 Example

In Figure 1, we see an example graph of word nodes, represented as a matrix, and a derived dependency structure.[4] We see that there are four edges from *The* to *market* and six from *The* to *crumbles*, for example. We then compute the PageRank of each node using the algorithm described in Page and Brin (1998); see also Figure 1. The PageRank values rank the nodes or the words. In Sect. 3, we describe a method for building a dependency tree from

[4]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.

from/to	The	market	crumbled	.
The	0	4	6	3
market	4	0	5	3
crumbled	4	4	0	4
.	3	4	6	0
PR(%)	22.8	24.1	30.3	22.7

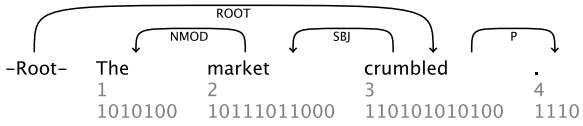

Figure 1: Graph, pagerank (PR) and predicted dependency structure for sentence 5, PTB-III Sect. 23.

a ranking of the nodes. This method will produce the correct analysis of this sentence; see Figure 1. 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

Consider the example in Figure 1 again. Once we have ranked the nodes in our dependency structure, we build a dependency structure from it using the parsing algorithm in Figure 2. 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 mapping from rank to sentence position.

The interesting step in the algorithm is the head selection step. Each word is assigned a head taken from all the previously used heads and the word to which a head was just assigned. Of these words, we simply select the closest head. If two possible heads are equally close, we select the one with highest PageRank.

Our parsing algorithm runs in $\mathcal{O}(n \log n)$, 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-headedness, and thus produces well-formed non-projective dependency trees. To see this, remember that wellformed dependency trees are such that all nodes but the artificial root nodes have a single incoming edge. This follows immediately from

1: $\pi = \langle n_1, \ldots, n_m \rangle$ # the ranking of nodes
2: $H = \langle n_0 \rangle$ # possible heads
3: $D = \emptyset$ # dependency structure
4: $pr2ind : \{1, \ldots, m\} \rightarrow \{1, \ldots, m\}$ # a mapping from rank to sentence position
5: **for** $n_i \in \pi$ **do**
6: **if** $|H|$=1 **then**
7: $c = 0$ # used to ensure single-headedness
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_j
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

Figure 2: Parsing algorithm.

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 that all words attach to this node or to nodes dominated by the root node. Finally, we ensure single-headedness by explicitly disregarding the root node once we have attached the node with highest rank to it (line 6–7). Our parsing algorithm does not guarantee projectivity, since the iterative graph-based ranking of nodes can permute the nodes in any order.

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 Sect. 23 of the English treebanks and the test sections of the remaining 11 treebanks, which were all 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 10 non-punctuation words and in reporting unlabeled attachment scores excluding punctuation.

4.1 Strictly unsupervised dependency parsing

We first evaluate the strictly unsupervised parsing model that has no access to POS information. Since we are not aware of other work in strictly unsupervised multi-lingual dependency parsing, so we compare against the best structural baseline (left-attach or right-attach) and the standard DMV-EM model of Klein and Manning (2004). The latter, however, *has* access to POS information and should not be thought of as a baseline. Results are presented in Figure 3.

It is interesting that we actually outperform DMV-EM on some languages. On average our scores are significantly better ($p < 0.01$) than the best structural baselines (3.8%), but DMV-EM with POS tags is still 3.0% better than our strictly unsupervised model. For English, our system performs a lot worse than Seginer (2007).

4.2 Unsupervised dependency parsing (standard)

We then evaluate our unsupervised dependency parser in the more standard scenario of parsing sentences annotated with POS. We now compare ourselves to two state-of-the-art models, namely DMV PR-AS 140 and E-DMV PR-AS 140 (Gillenwater et al., 2010). Finally, we also report results of the IBM model 3 proposed by Brody (2010) for unsupervised dependency parsing, since this is the only recent pro-

	baseline	EM	ours
Bulgarian	37.7	37.8	**41.9**
Czech	**32.5**	29.6	28.7
Danish	43.7	**47.2**	43.7
Dutch	**38.7**	37.1	33.1
English	33.9	**45.8**	36.1
German	27.2	35.7	**36.9**
Japanese	44.7	52.8	**56.5**
Portuguese	35.5	**35.7**	35.2
Slovene	25.5	**42.3**	30.0
Spanish	27.0	**45.8**	38.4
Swedish	30.6	**39.4**	34.5
Turkish	36.6	**46.8**	45.9
AV	34.5	41.3	38.3

Figure 3: Unlabeled attachment scores (in %) on raw text. (EM baseline has access to POS.)

posal we are aware of that departs significantly from the DMV model. The results are presented in Figure 4.

Our results are on average significantly better than DMV PR-AS 140 (2.5%), and better than DMV PR-AS 140 on 8/12 languages. E-DMV PR-AS 140 is slightly better than our model on average (1.3%), but we still obtain better results on 6/12 languages. Our results are a lot better than IBM-M3. Naseem et al. (2010) report better results than ours on Portuguese, Slovene, Spanish and Swedish, but worse on Danish.

5 Error analysis

In our error analysis, we focus on the results for German and Turkish. We first compare the results of the strictly unsupervised model on German with the results on German text annotated with POS. The main difference between the two models is that more links to verbs are added to the sentence graph prior to ranking nodes when parsing text annotated with POS. For this reason, the latter model improves considerably in attaching verbs compared to the strictly unsupervised model:

acc	strict-unsup	unsup
NN	43%	48%
NE	41%	39%
VVFIN	31%	100%
VAFIN	9%	86%
VVPP	13%	53%

While the strictly unsupervised model is about as good at attaching nouns as the model with POS, it is much worse attaching verbs. Since more links to verbs are added, verbs receive higher rank, and this improves f-scores for attachments to the artificial root node:

f-score	strict-unsup	unsup
to_root	39.5%	74.0%
1	62.3%	69.6%
2	7.4%	24.4%
3–6	0	22.4%
7	0	0

This is also what helps the model with POS when parsing the example sentence in Figure 5. The POS-informed parser also predicts longer dependencies.

The same pattern is observed in the Turkish data, but perhaps less dramatically so:

acc	strict-unsup	unsup
Noun	43%	42%
Verb	41%	51%

The increase in accuracy is again higher with verbs than with nouns, but the error reduction was higher for German.

f-score	strict-unsup	unsup
to_root	57.4%	90.4%
1	65.7%	69.6%
2	32.1%	26.5%
3–6	11.6%	24.7%
7	0	12.5%

The parsers predict more long dependencies for Turkish than for German; precision is generally good, but recall is very low.

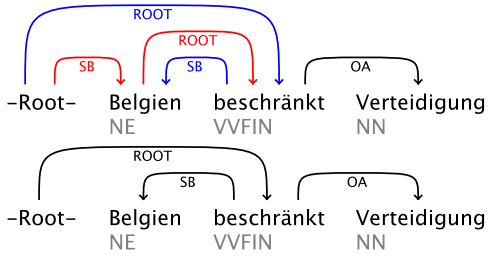

Figure 5: Predicted dependency structures for sentence 4 in the German test section; strictly unsupervised (above) and standard (below) approach. Red arcs show wrong decisions.

6 Conclusion

We have presented a new approach to unsupervised dependency parsing. The key idea is that a depen-

	DMV PR-AS 140	E-DMV PR-AS 140	ours	IBM-M3
Bulgarian	54.0	**59.8**	52.5	
Czech	32.0	**54.6**	42.8	
Danish	42.4	47.2	**55.2**	41.9
Dutch	37.9	46.6	**49.4**	35.3
English	61.9	**64.4**	50.2	39.3
German	39.6	35.7	**50.4**	
Japanese	**60.2**	59.4	58.3	
Portuguese	47.8	49.5	**52.8**	
Slovene	50.3	**51.2**	44.1	
Spanish	**62.4**	57.9	52.1	
Swedish	38.7	41.4	**45.5**	
Turkish	53.4	56.9	**57.9**	
AV	48.4	**52.2**	50.9	

Figure 4: Unlabeled attachment scores (in %) on text annotated with POS.

dency structure also expresses centrality or saliency, so by modeling centrality directly, we obtain information that we can use 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.

References

Eneko Agirre and Aitor Soroa. 2009. Personalizing pagerank for word sense disambiguation. In *EACL*.

Samuel Brody. 2010. It depends on the translation: unsupervised dependency parsing via word alignment. In *EMNLP*.

Sabine Buchholz and Erwin Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In *CoNLL*.

Shay Cohen, Kevin Gimpel, and Noah Smith. 2008. Unsupervised bayesian parameter estimation for dependency parsing. In *NIPS*.

Gregory Druck, Gideon Mann, and Andrew McCallum. 2009. Semi-supervised learning of dependency parsers using generalized expectation criteria. In *ACL-IJCNLP*.

Jason Eisner and Noah A. Smith. 2005. Parsing with soft and hard constraints on dependency length. In *IWPT*.

K Ganesan, C Zhai, and J Han. 2010. Opinosis: a graph-based approach to abstractive summarization of highly redudant opinions. In *COLING*.

Jennifer Gillenwater, Kuzman Ganchev, Joao Graca, Fernando Pereira, and Ben Taskar. 2010. Sparsity in dependency grammar induction. In *ACL*.

Dan Klein and Christopher Manning. 2004. Corpus-based induction of syntactic structure: models of dependency and constituency. In *ACL*.

Mitchell Marcus, Mary Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. 2005. Non-projective dependency parsing using spanning tree algorithms. In *HLT-EMNLP*.

Rada Mihalcea and Paul Tarau. 2004. Textrank: bringing order into texts. In *EMNLP*.

Tahira Naseem, Harr Chen, Regina Barzilay, and Mark Johnson. 2010. Using universal linguistic knowledge to guide grammar induction. In *EMNLP*.

Joakim Nivre. 2009. Non-projective dependency parsing in expected linear time. In *ACL-IJCNLP*.

Larry Page and Sergey Brin. 1998. The anatomy of a large-scale hypertextual web search engine. In *International Web Conference*.

Yoav Seginer. 2007. Fast unsupervised incremental parsing. In *ACL*.

Noah Smith and Jason Eisner. 2005. Contrastive estimation: training log-linear models on unlabeled data. In *ACL*.

Noah Smith and Jason Eisner. 2006. Annealing structural bias in multilingual weighted grammar induction. In *COLING-ACL*.

David Smith and Jason Eisner. 2009. Parser adaptation and projection with quasi-synchronous grammar features. In *EMNLP*.

Valentin Spitkovsky, Hiyan Alshawi, and Daniel Jurafsky. 2009. Baby steps: how "less is more" in unsupervised dependency parsing. In *NIPS Workshop on Grammar Induction, Representation of Language and Language Learning*.

Valentin Spitkovsky, Hiyan Alshawi, Daniel Jurafsky, and Christopher Manning. 2010. Viterbi training improves unsupervised dependency parsing. In *CoNNL*.

Kathrin Spreyer, Lilja Øvrelid, and Jonas Kuhn. 2010. Training parsers on partial trees: a cross-language comparison. In *LREC*.