A Transition-Based System for Joint Part-of-Speech Tagging and Labeled Non-Projective Dependency Parsing

Bernd Bohnet
Institute for Natural Language Processing
University Stuttgart
bohnet@ims.uni-stuttgart.de

Joakim Nivre
Department of Linguistics and Philology
Uppsala University
joakim.nivre@lingfil.uu.se

Abstract

Most current dependency parsers presuppose that input words have been morphologically disambiguated using a part-of-speech tagger before parsing begins. We present a transition-based system for joint part-of-speech tagging and labeled dependency parsing with non-projective trees. Experimental evaluation on Chinese, Czech, English and German shows consistent improvements in both tagging and parsing accuracy when compared to a pipeline system, which lead to improved state-of-the-art results for all languages.

1 Introduction

Dependency-based syntactic parsing has been the focus of intense research efforts during the last decade, and the state of the art today is represented by globally normalized discriminative models that are induced using structured learning. Graph-based models parameterize the parsing problem by the structure of the dependency graph and normally use dynamic programming for inference (McDonald et al., 2005; McDonald and Pereira, 2006; Carreras, 2007; Koo and Collins, 2010; Bohnet, 2010), but other inference methods have been explored especially for non-projective parsing (Riedel and Clarke, 2006; Smith and Eisner, 2008; Martins et al., 2009; Martins et al., 2010; Koo et al., 2010). Transition-based models parameterize the problem by elementary parsing actions and typically use incremental beam search (Titov and Henderson, 2007; Zhang and Clark, 2008; Zhang and Clark, 2011). Despite notable differences in model structure, graph-based and transition-based parsers both give state-of-the-art accuracy with proper feature selection and optimization (Koo and Collins, 2010; Zhang and Nivre, 2011; Bohnet, 2011).

It is noteworthy, however, that almost all dependency parsers presuppose that the words of an input sentence have been morphologically disambiguated using (at least) a part-of-speech tagger. This is in stark contrast to the best parsers based on PCFG models, such as the Brown parser (Charniak and Johnson, 2005) and the Berkeley parser (Petrov et al., 2006; Petrov and Klein, 2007), which not only can perform their own part-of-speech tagging but normally give better parsing accuracy when they are allowed to do so. This suggests that joint models for tagging and parsing might improve accuracy also in the case of dependency parsing.

It has been argued that joint morphological and syntactic disambiguation is especially important for richly inflected languages, where there is considerable interaction between morphology and syntax such that neither can be fully disambiguated without considering the other. Thus, Lee et al. (2011) show that a discriminative model for joint morphological disambiguation and dependency parsing outperforms a pipeline model in experiments on Latin, Ancient Greek, Czech and Hungarian. However, Li et al. (2011) and Hatori et al. (2011) report improvements with a joint model also for Chinese, which is not a richly inflected language but is nevertheless rich in part-of-speech ambiguities.

In this paper, we present a transition-based model for joint part-of-speech tagging and labeled dependency parsing with non-projective trees. Exper-

Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pages 1455–1465, Jeju Island, Korea, 12–14 July 2012. ©2012 Association for Computational Linguistics

iments show that joint modeling improves both tagging and parsing accuracy, leading to state-of-the-art accuracy for richly inflected languages like Czech and German as well as more configurational languages like Chinese and English. To our knowledge, this is the first joint system that performs labeled dependency parsing. It is also the first joint system that achieves state-of-the-art accuracy for non-projective dependency parsing.

2 Transition-Based Tagging and Parsing

Transition-based dependency parsing was pioneered by Yamada and Matsumoto (2003) and Nivre et al. (2004), who used classifiers trained to predict individual actions of a deterministic shift-reduce parser. Recent research has shown that better accuracy can be achieved by using beam search and optimizing models on the entire sequence of decisions needed to parse a sentence instead of single actions (Zhang and Clark, 2008; Huang and Sagae, 2010; Zhang and Clark, 2011; Zhang and Nivre, 2011; Bohnet, 2011). In addition, a number of different transition systems have been proposed, in particular for dealing with non-projective dependencies, which were beyond the scope of early systems (Attardi, 2006; Nivre, 2007; Nivre, 2009; Titov et al., 2009).

In this section, we start by defining a transition system for joint tagging and parsing based on the non-projective transition system proposed in Nivre (2009). We then show how to perform beam search and structured online learning with this model, and conclude by discussing feature representations.

2.1 Transition System

Given a set P of part-of-speech tags and a set D of dependency labels, a *tagged dependency tree* for a sentence $x = w_1, \ldots, w_n$ is a directed tree $T = (V_x, A)$ with labeling functions π and δ such that:

1. $V_x = \{0, 1, \ldots, n\}$ is a set of nodes,
2. $A \subseteq V_x \times V_x$ is a set of arcs,
3. $\pi : V_x \rightarrow P$ is a labeling function for nodes,
4. $\delta : A \rightarrow D$ is a labeling function for arcs,
5. 0 is the root of the tree.

The set V_x of *nodes* is the set of positive integers up to and including n, each corresponding to the linear position of a word in the sentence, plus an extra artificial root node 0. The set A of *arcs* is a set of pairs (i, j), where i is the *head* node and j is the *dependent* node. The functions π and δ assign a unique part-of-speech label to each node/word and a unique dependency label to each arc, respectively. This notion of dependency tree differs from the standard definition only by including part-of-speech labels as well as dependency labels (Kübler et al., 2009).

Following Nivre (2008), we define a *transition system* for dependency parsing as a quadruple $S = (C, T, c_s, C_t)$, where

1. C is a set of *configurations*,
2. T is a set of *transitions*, each of which is a (partial) function $t : C \rightarrow C$,
3. c_s is an *initialization function*, mapping a sentence x to a configuration $c \in C$,
4. $C_t \subseteq C$ is a set of *terminal configurations*.

A transition sequence for a sentence x in S is a sequence of configuration-transition pairs $C_{0,m} = [(c_0, t_0), (c_1, t_1), \ldots, (c_m, t_m)]$ where $c_0 = c_s(x)$, $t_m(c_m) \in C_t$ and $t_i(c_i) = c_{i+1}$ $(0 \leq i < m)$.[1]

In this paper, we take the set C of configurations to be the set of all 5-tuples $c = (\Sigma, B, A, \pi, \delta)$ such that Σ (the stack) and B (the buffer) are disjoint sublists of the nodes V_x of some sentence x, A is a set of dependency arcs over V_x, and π and δ are labeling functions as defined above. We take the initial configuration for a sentence $x = w_1, \ldots, w_n$ to be $c_s(x) = ([0], [1, \ldots, n], \{\,\}, \bot, \bot)$, where $\bot$ is the function that is undefined for all arguments, and we take the set C_t of terminal configurations to be the set of all configurations of the form $c = ([0], [\,], A, \pi, \delta)$ (for any A, π and δ). The tagged dependency tree defined for x by $c = (\Sigma, B, A, \pi, \delta)$ is the tree (V_x, A) with labeling functions π and δ, which we write $\textsc{Tree}(x, c)$.

The set T of transitions is shown in Figure 1. The $\textsc{Left-Arc}_d$ and $\textsc{Right-Arc}_d$ transitions both add an arc (with dependency label d) between the two nodes on top of the stack and replaces these nodes by the head node of the new arc (which is the rightmost node for $\textsc{Left-Arc}_d$ and the leftmost node for $\textsc{Right-Arc}_d$). The $\textsc{Shift}_p$ transition extracts the

[1] This definition of transition sequence differs from that of Nivre (2008) but is equivalent and suits our presentation better.

Transition		**Condition**
LEFT-ARC$_d$	$([\sigma\|i,j],B,A,\pi,\delta) \Rightarrow ([\sigma\|j],B,A\cup\{(j,i)\},\pi,\delta[(j,i)\rightarrow d])$	$i \neq 0$
RIGHT-ARC$_d$	$([\sigma\|i,j],B,A,\pi,\delta) \Rightarrow ([\sigma\|i],B,A\cup\{(i,j)\},\pi,\delta[(i,j)\rightarrow d])$	
SHIFT$_p$	$(\sigma,[i\|\beta],A,\pi,\delta) \Rightarrow ([\sigma\|i],\beta,A,\pi[i\rightarrow p],\delta)$	
SWAP	$([\sigma\|i,j],\beta,A,\pi,\delta) \Rightarrow ([\sigma\|j],[i\|\beta],A,\pi,\delta)$	$0 < i < j$

Figure 1: Transitions for joint tagging and dependency parsing extending the system of Nivre (2009). The stack Σ is represented as a list with its head to the right (and tail σ) and the buffer B as a list with its head to the left (and tail β). The notation $f[a \rightarrow b]$ is used to denote the function that is exactly like f except that it maps a to b.

first node in the buffer, pushes it onto the stack and labels it with the part-of-speech tag p. The SWAP transition extracts the second topmost node from the stack and moves it back to the buffer, subject to the condition that the two top nodes on the stack are still in the order given by the sentence.

Except for the addition of a tag parameter p to the SHIFT transition, this is equivalent to the system described in Nivre (2009), which thanks to the SWAP transition can handle arbitrary non-projective trees. The soundness and completeness results given in that paper trivially carry over to the new system. The only thing to note is that, before a terminal configuration can be reached, every word has to be pushed onto the stack in a SHIFT$_p$ transition, which ensures that every node/word in the output tree will be tagged.

2.2 Inference and Learning

While early transition-based parsers generally used greedy best-first inference and locally trained classifiers, recent work has shown that higher accuracy can be obtained using beam search and global structure learning to mitigate error propagation. In particular, it seems that the globally learned models can exploit a much richer feature space than locally trained classifiers, as shown by Zhang and Nivre (2011). Since joint tagging and parsing increases the size of the search space and is likely to require novel features, we use beam search in combination with structured perceptron learning.

```
PARSE($x$, $\mathbf{w}$, $b_1$, $b_2$)
 1  $h_0.c \leftarrow c_s(x)$
 2  $h_0.s \leftarrow 0.0$
 3  $h_0.\mathbf{f} \leftarrow \{0.0\}^{dim(\mathbf{w})}$
 4  BEAM $\leftarrow [h_0]$
 5  while $\exists h \in$ BEAM : $h.c \notin C_t$
 6      TMP $\leftarrow [\,]$
 7      foreach $h \in$ BEAM
 8          foreach $t \in T$ : PERMISSIBLE($h.c, t$)
 9              $h.\mathbf{f} \leftarrow h.\mathbf{f} + \mathbf{f}(h.c, t)$
10              $h.s \leftarrow h.s + \mathbf{f}(h.c, t) \cdot \mathbf{w}$
11              $h.c \leftarrow t(h.c)$
12              TMP $\leftarrow$ INSERT($h$, TMP)
13      BEAM $\leftarrow$ PRUNE(TMP, $b_1$, $b_2$)
14  $h \leftarrow$ TOP(BEAM)
15  $y \leftarrow$ TREE($x, h.c$)
16  return $y$
```

Figure 2: Beam search algorithm for joint tagging and dependency parsing of input sentence x with weight vector $\mathbf{w}$ and beam parameters b_1 and b_2. The symbols $h.c$, $h.s$ and $h.\mathbf{f}$ denote, respectively, the configuration, score and feature representation of a hypothesis h; $h.c.A$ denotes the arc set of $h.c$.

The beam search algorithm used to derive the best parse y for a sentence x is outlined in Figure 2. In addition to the sentence x, it takes as input a weight vector $\mathbf{w}$ corresponding to a linear model for scoring transitions out of configurations and two pruning parameters b_1 and b_2. A parse hypothesis h is represented by a configuration $h.c$, a score $h.s$ and a feature vector $h.\mathbf{f}$ for the transition sequence up to $h.c$. Hypotheses are stored in the list BEAM, which is sorted by descending scores and initialized to hold the hypothesis h_0 corresponding to the initial configuration $c_s(x)$ with score 0.0 and all features set to 0.0 (lines 1–4). In the main loop (lines 5–13), a set of new hypotheses is derived and stored in the list TMP, which is finally pruned and assigned as the new value of BEAM. The main loop terminates when all hypotheses in BEAM contain terminal configurations, and the dependency tree extracted from the top scoring hypothesis is returned (lines 14–16).

The set of new hypotheses is created in two nested loops (lines 7–12), where every hypothesis h in BEAM is updated using every permissible transition t for the configuration $h.c$. The feature representation of the new hypothesis is obtained by adding the feature vector $\mathbf{f}(t, h.c)$ for the current configuration-transition pair to the feature vector of the old hypothesis (line 9). Similarly, the score of the new hypothesis is the sum of the score $\mathbf{f}(t, h.c) \cdot \mathbf{w}$ of the current configuration-transition pair and the score of the old hypothesis (line 10). The feature representation/score of a complete parse y for x with transition sequence $C_{0,m}$ is thus the sum of the feature representations/scores of the configuration-transition pairs in $C_{0,m}$:

$$\begin{aligned} \mathbf{f}(x,y) &= \sum_{(c,t) \in C_{0,m}} \mathbf{f}(c,t) \\ s(x,y) &= \sum_{(c,t) \in C_{0,m}} \mathbf{f}(c,t) \cdot \mathbf{w} \end{aligned}$$

Finally, the configuration of the new hypothesis is obtained by evaluating $t(h.c)$ (line 11). The new hypothesis is then inserted into TMP in score-sorted order (line 12).

The pruning parameters b_1 and b_2 determine the number of hypotheses allowed in the beam and at the same time control the tradeoff between syntactic and morphological ambiguity. First, we extract the b_1 highest scoring hypotheses with distinct dependency trees. Then we extract the b_2 highest scoring remaining hypotheses, which will typically be tagging variants of dependency trees that are already in the beam. In this way, we prevent the beam from getting filled up with too many tagging variants of the same dependency tree, which was found to be harmful in preliminary experiments.

One final thing to note about the inference algorithm is that the notion of permissibility for a transition t out of a configuration c can be used to capture not only formal constraints on transitions – such as the fact that it is impossible to perform a SHIFT$_p$ transition with an empty buffer or illegal to perform a LEFT-ARC$_d$ transition with the special root node on top of the stack – but also to filter out unlikely dependency labels or tags. Thus, in the experiments later on, we will typically constrain the parser so that SHIFT$_p$ is permissible only if p is one of the k best part-of-speech tags with a score no more than α below the score of the 1-best tag, as determined by a preprocessing tagger. We also filter out instances of LEFT-ARC$_d$ and RIGHT-ARC$_d$, where d does not occur in the training data for the predicted part-of-speech tag combination of the head and dependent. This procedure leads to a significant speed up.

In order to learn a weight vector $\mathbf{w}$ from a training set $\{(x_j, y_j)\}_{j=1}^{\mathcal{T}}$ of sentences with their tagged dependency trees, we use a variant of the structured perceptron, introduced by Collins (2002), which makes N iterations over the training data and updates the weight vector for every sentence x_j where the highest scoring parse y^* is different from y_j. More precisely, we use the passive-aggressive update of Crammer et al. (2006):

$$\mathbf{w}^{i+1} = \mathbf{w}^i + \tau(\mathbf{f}(x_j, y_j) - \mathbf{f}(x_j, y^*))$$

where

$$\tau = \frac{\mathbf{f}(x_j, y_j) - \mathbf{f}(x_j, y^*)}{||\mathbf{f}(x_j, y_j) - \mathbf{f}(x_j, y^*)||^2}$$

We also use the early update strategy found beneficial for parsing in several previous studies (Collins and Roark, 2004; Zhang and Clark, 2008; Huang and Sagae, 2010), which means that, during learning, we terminate the beam search as soon as the hypothesis corresponding to the gold parse y_j falls out of the beam and update with respect to the partial transition sequence constructed up to that point. Finally, we use the standard technique of averaging over all weight vectors, as originally proposed by Collins (2002).

2.3 Feature Representations

As already noted, the feature representation $\mathbf{f}(x, y)$ of an input sentence x with parse y decomposes into feature representations $\mathbf{f}(c, t)$ for the transitions $t(c)$ needed to derive y from $c_s(x)$. Features may refer to any aspect of a configuration, as encoded in the stack Σ, the buffer B, the arc set A and the labelings π and δ. In addition, we assume that each word w in the input is assigned up to k candidate part-of-speech tags $\pi_i(w)$ with corresponding scores $s(\pi_i(w))$.

Features involving word prefixes and suffixes
$\pi_i(B_0)p_2(B_0)$, $\pi_i(B_0)s_2(B_0)$, $\pi_i(B_0)p_1(B_0)p_1(\Sigma_0)$
$\pi_i(\Sigma_0)p_1(\Sigma_0)p_1(\Sigma_1)$, $\pi_i(\Sigma_0)s_1(\Sigma_0)s_1(\Sigma_0)$
$\pi_i(\Sigma_0)p_2(\Sigma_0)s_3(\Sigma_1)$,$\pi_i(\Sigma_0)s_3(\Sigma_0)p_2(\Sigma_1)$
$\pi_i(\Sigma_0)w(B_0)s_1(\Sigma_0)$, $\pi_i(\Sigma_0)w(B_0)s_2(\Sigma_0)$
Features involving tag score differences and ranks
$\pi_i(B_0)[s(\pi_1(B_0)) - s(\pi_i(B_0))]$
$\pi_i(B_0)\pi_i(\Sigma_0)[s(\pi_1(B_0)) - s(\pi_i(B_0))]\ i$
$\pi_i(B_0)[s(\pi_1(B_0)) - s(\pi_i(B_0))]\pi(\Sigma_0)$
$w(B_0)[s(\pi_1(B_0)) - s(\pi_i(B_0))]\pi(\Sigma_0)$

Figure 3: Specialized feature templates for tagging. We use Σ_i and B_i to denote the ith token in the stack Σ and buffer B, respectively, with indexing starting at 0, and we use the following functors to extract properties of a token: $\pi_i()$ = ith best tag; $s(\pi_i())$ = score of ith best tag; $\pi()$ = finally predicted tag; $w()$ = word form; $p_i()$ = word prefix of i characters; $s_i()$ = word suffix of i characters. Score differences are binned in discrete steps of 0.05.

The bulk of features used in our system are taken from Zhang and Nivre (2011), although with two important differences. First of all, like Hatori et al. (2011), we have omitted all features that presuppose an arc-eager parsing order, since our transition system defines an arc-standard order. Secondly, any feature that refers to the part-of-speech tag of a word w in the buffer B will in our system refer to the top-scoring tag $\pi_1(w)$, rather than the finally predicted tag. By contrast, for a word in the stack Σ, part-of-speech features refer to the tag $\pi(w)$ chosen when shifting w onto the stack (which may or may not be the same as $\pi_1(w)$).

In addition to the standard features for transition-based dependency parsing, we have added features specifically to improve the tagging step in the joint model. The templates for these features, which are specified in Figure 3, all involve the ith best tag assigned to the first word of the buffer B (the next word to be shifted in a SHIFT$_p$ transition) in combination with neighboring words, word prefixes, word suffixes, score differences and tag rank.

Finally, in some experiments, we make use of two additional feature sets, which we call graph features (G) and cluster features (C), respectively. Graph features are defined over the factors of a graph-based dependency parser, which was shown to improve the accuracy of a transition-based parser by Zhang and Clark (2008). However, while their features were limited to certain first- and second-order factors, we use features over second- and third-order factors as found in the parsers of Bohnet and Kuhn (2012). These features are scored as soon as the factors are completed, using a technique that is similar to what Hatori et al. (2011) call delayed features, although they use it for part-of-speech tags in the lookahead while we use it for subgraphs of the dependency tree. Cluster features, finally, are features over word clusters, as first used by Koo et al. (2008), which replace part-of-speech tag features.[2]

We use a hash kernel to map features to weights. It has been observed that most of the computing time in feature-rich parsers is spent retrieving the index of each feature in the weight vector (Bohnet, 2010). This is usually done via a hash table, but significant speedups can be achieved by using a hash kernel, which simply replaces table lookup by a hash function (Bloom, 1970; Shi et al., 2009; Bohnet, 2010). The price to pay for these speedups is that there may be collisions, so that different features are mapped to the same index, but this is often compensated by the fact that the lower time and memory requirements of the hash kernel enables the use of negative features, that is, features that are never seen in the training set but occur in erroneous hypotheses at training time and can therefore be helpful also at inference time. As a result, the hash kernel often improves accuracy as well as efficiency compared to traditional techniques that only make use of features that occur in gold standard parses (Bohnet, 2010).

3 Experiments

We have evaluated the model for joint tagging and dependency parsing on four typologically diverse languages: Chinese, Czech, English, and German.

3.1 Setup

Most of the experiments use the CoNLL 2009 data sets with the training, development and test split used in the Shared Task (Hajič et al., 2009), but for better comparison with previous work we also report results for the standard benchmark data sets for Chinese and English. For Chinese, this is the Penn Chinese Treebank 5.1 (CTB5), converted

[2] For replicability, a complete description of all features can be found at http://stp.lingfil.uu.se/∼nivre/exp/emnlp12.html.

Parser		Chinese				Czech				English				German			
k	α	TLAS	LAS	UAS	POS	TLAS	LAS	UAS	POS	TLAS	LAS	UAS	POS	TLAS	LAS	UAS	POS
1	0.0	73.85	76.12	80.01	92.78	82.36	82.65	88.03	93.26	85.82	87.17	90.41	97.32	85.08	86.60	89.17	97.24
2	0.1	74.39	76.52	80.41	93.37	82.74	83.01	88.34	99.39	**86.43**	87.79	91.02	97.49	86.12	87.22	89.69	97.85
3	0.1	**74.47**	76.63	80.50	93.38	82.76	82.97	88.33	99.40	86.40	87.78	90.99	97.43	86.03	87.27	89.60	97.74
3	0.2	74.35	76.48	80.38	93.43	**82.85**	83.11	88.44	99.32	86.35	87.79	91.01	97.52	86.24	87.37	89.72	97.90
3	0.3	74.18	76.33	80.28	93.48	82.78	83.05	88.38	99.33	85.94	87.57	90.87	96.97	**86.35**	87.46	89.86	97.90
3	0.4													86.14	87.23	89.66	97.79

Table 1: Accuracy scores for the CoNLL 2009 shared task development sets as a function of the number of tags k and the score threshold α. Beam parameters fixed at b_1 = 40, b_2 = 4.

with the head-finding rules and conversion tools of Zhang and Clark (2008), and with the same split as in Zhang and Clark (2008) and Li et al. (2011).[3] For English, this is the WSJ section of the Penn Treebank, converted with the head-finding rules of Yamada and Matsumoto (2003) and the labeling rules of Nivre (2006).[4]

In order to assign k-best part-of-speech tags and scores to words in the training set, we used a perceptron tagger with 10-fold jack-knifing. The same type of tagger was trained on the entire training set in order to supply tags for the development and test sets. The feature set of the tagger was optimized for English and German and provides state-of-the-art accuracy for these two languages. The 1-best tagging accuracy for section 23 of the Penn Treebank is 97.28, which is on a par with Toutanova et al. (2003). For German, we obtain a tagging accuracy of 97.24, which is close to the 97.39 achieved by the RF-Tagger (Schmid and Laws, 2008), which to our knowledge is the best tagger for German.[5] The results are not directly comparable to the RF-Tagger as it was evaluated on a different part of the Tiger Treebank and trained on a larger part of the Treebank. We could not use the larger training set as it contains the test set of the CoNLL 2009 data that we use to evaluate the joint model. For Czech, the 1-best tagging accuracy is 99.11 and for Chinese 92.65 on the CoNLL 2009 test set.

We trained parsers with 25 iterations and report results for the model obtained after the last iteration. For cluster features, available only for English and German, we used standard Brown clusters based on the English and German Gigaword Corpus. We restricted the vocabulary to words that occur at least 10 times, used 800 clusters, and took cluster prefixes of length 6 to define features.

We report the following evaluation metrics: part-of-speech accuracy (POS), unlabeled attachment score (UAS), labeled attachment score (LAS), and tagged labeled attachment score (TLAS). TLAS is a new metric defined as the percentage of words that are assigned the correct part-of-speech tag, the correct head and the correct dependency label. In line with previous work, punctuation is included in the evaluation for the CoNLL data sets but excluded for the two benchmark data sets.

3.2 Results

Table 1 presents results on the development sets of the CoNLL 2009 shared task with varying values of the two tag parameters k (number of candidates) and α (maximum score difference to 1-best tag) and beam parameters fixed at b_1 = 40 and b_2 = 4. We use the combined TLAS score on the development set to select the optimal settings for each language. For Chinese, we obtain the best result with 3 tags and a threshold of 0.1.[6] Compared to the baseline, we observe a POS improvement of 0.60 and a LAS improvement of 0.51. For Czech, we get the best TLAS with k = 3 and α = 0.2, where POS improves by 0.06 and LAS by 0.46. For English, the best setting is k = 2 and α = 0.1 with a POS improvement of 0.17 and a LAS improvement of 0.62. For German, finally, we see the greatest improvement with k = 3

[3] Training: 001–815, 1001–1136. Development: 886–931, 1148–1151. Test: 816–885, 1137–1147.

[4] Training: 02-21. Development: 24. Test: 23.

[5] The RF-Tagger can take advantage of an additional lexicon and then reaches 97.97. The lexicon supplies entries for additional words that are not found in the training corpus and additional tags for words that do occur in the training data (Schmid and Laws, 2008).

[6] While tagging accuracy (POS) increases with larger values of α, TLAS decreases because of a drop in LAS.

Parser	**Chinese**				**Czech**				**English**				**German**			
	TLAS	LAS	UAS	POS	TLAS	LAS	UAS	POS	TLAS	LAS	UAS	POS	TLAS	LAS	UAS	POS
Gesmundo et al. (2009)		76.11		92.37		80.38		99.33		88.79		97.48		87.28		95.46
Bohnet (2010)		76.99		92.37		80.96		99.33		90.33		97.48		88.06		95.46
Baseline (k = 1), b_1 = 40	73.66	76.55	80.77	92.65	82.07	82.44	87.83	99.11	87.89	89.19	91.74	97.57	86.11	87.78	90.13	97.24
Best dev setting, b_1 = 40	74.72	77.00	81.18	93.06	82.56	82.70	88.07	99.32	88.26	89.54	92.06	97.77	86.91	88.23	90.43	97.63
Adding G, b_1 = 80	75.84	78.51	82.52	93.19	83.38	83.73	88.82	99.33	88.92	90.20	92.60	97.77	87.86	89.05	91.16	97.78
Adding G+C, b_1 = 80									89.22	90.60	92.87	97.84	88.31	89.38	91.37	98.05

Table 2: Accuracy scores for the CoNLL 2009 shared task test sets. Rows 1–2: Top performing systems in the shared CoNLL Shared Task 2009; Gesmundo et al. (2009) was placed first in the shared task; for Bohnet (2010), we include the updated scores later reported due to some improvements of the parser. Rows 3–4: Baseline (k = 1) and best settings for k and α on development set. Rows 5–6: Wider beam (b_1 = 80) and added graph features (G) and cluster features (C). Second beam parameter b_2 fixed at 4 in all cases.

and α = 0.3, where POS improves by 0.66 and LAS by 0.86.

Table 2 shows the results on the CoNLL 2009 test sets. For all languages except English, we obtain state-of-the-art results already with b_1 = 40 (row 4), and for all languages both tagging and parsing accuracy improve compared to the baseline (row 3). The improvement in TLAS is statistically significant with $p < 0.01$ for all languages (paired t-test). Row 5 shows the scores with a beam of 80 and the additional graph features. Here the LAS scores for Chinese, Czech and German are higher than the best results on the CoNLL 2009 data sets, and the score for English is highly competitive. For Chinese, we achieve 78.51 LAS, which is 1.5 percentage points higher than the reference score, while the POS score is 0.54 higher than our baseline. For Czech, we get 83.73 LAS, which is by far the highest score reported for this data set, together with state-of-the-art POS accuracy. For German, we obtain 89.05 LAS and 97.78 POS, which in both cases is substantially better than in the CoNLL shared task. We believe it is also the highest POS accuracy ever reported for a tagger/parser trained only on the Tiger Treebank. Row 6, finally, presents results with added cluster features for English and German, which results in additional improvements in all metrics.

Table 3 gives the results for the Penn Treebank converted with the head-finding rules of Yamada and Matsumoto (2003) and the labeling rules of Nivre (2006). We use k = 3 and α = 0.4, which gave the best results on the development set. The UAS improves by 0.24 when we do joint tagging and parsing. The POS accuracy improves slightly by 0.12 but to a lower degree than for the English CoNLL data where we observed an improvement of 0.20. Nonetheless, the improvement in the joint TLAS score is statistically significant at $p < 0.01$ (paired t-test). Our joint tagger and dependency parser with graph features gives very competitive unlabeled dependency scores for English with 93.38 UAS. To the best of our knowledge, this is the highest score reported for a (transition-based) dependency parser that does not use additional information sources. By adding cluster features and widening the beam to $b_1 = 80$, we achieve 93.67 UAS. We also obtain a POS accuracy of 97.42, which is on a par with the best results obtained using semi-supervised taggers

Parser	TLAS	UAS	LAS	POS
McDonald et al. (2005)		90.9		
McDonald and Pereira (2006)		91.5		
Zhang and Clark (2008)		92.1		
Huang and Sagae (2010)		92.1		
Koo and Collins (2010)		93.04		
Zhang and Nivre (2011)		92.9		
Martins et al. (2010)		93.26		
Koo et al. (2008) †		93.16		
Carreras et al. (2008) †		93.5		
Suzuki et al. (2009) †		93.79		
Baseline (k = 1), b_1 = 40	89.42	92.79	91.71	97.28
Best dev setting, b_1 = 40	89.75	93.03	91.92	97.40
Adding G, b_1 = 40	90.12	93.38	92.44	97.33
Adding G+C, b_1 = 80 †	90.41	93.67	92.68	97.42

Table 3: Accuracy scores for WSJ-PTB converted with head rules of Yamada and Matsumoto (2003) and labeling rules of Nivre (2006). Best dev setting: k = 3, α = 0.4. Results marked with † use additional information sources and are not directly comparable to the others.

Parser	TLAS	UAS	LAS	POS
MSTParser1		75.56		93.51
MSTParser2		77.73		93.51
Li et al. (2011) 3rd-order		80.60		92.80
Li et al. (2011) 2nd-order		80.55		93.08
Hatori et al. (2011) HS		79.60		94.01
Hatori et al. (2011) ZN		81.20		93.94
Baseline (k = 1), b_1 = 40	61.95	80.33	76.79	92.81
Best dev setting, b_1 = 40	62.54	80.59	77.06	93.11
Adding G, b_1 = 80	63.20	81.42	77.91	93.24

Table 4: Accuracy scores for Penn Chinese Treebank converted with the head rules of Zhang and Clark (2008). Best dev setting: k = 3, α = 0.1. MSTParser results from Li et al. (2011). UAS scores from Li et al. (2011) and Hatori et al. (2011) recalculated from the separate accuracy scores for root words and non-root words reported in the original papers.

(Søgaard, 2011).

Table 4 shows the results for the Chinese Penn Treebank CTB 5.1 together with related work. In experiments with the development set, we could confirm the results from the Chinese CoNLL data set and obtained the best results with the same settings (k = 3, α = 0.1). With b_1 = 40, UAS improves by 0.25 and POS by 0.30, and the TLAS improvement is again highly significant ($p < 0.01$, paired t-test). We get the highest UAS, 81.42, with a beam of 80 and added graph features, in which case POS accuracy increases from 92.81 to 93.24. Since our tagger was not optimized for Chinese, we have lower baseline results for the tagger than both Li et al. (2011) and Hatori et al. (2011) but still manage to achieve the highest reported UAS.

The speed of the joint tagger and dependency parser is quite reasonable with about 0.4 seconds per sentence on the WSJ-PTB test set, given that we perform tagging and labeled parsing with a beam of 80 while incorporating the features of a third-order graph-based model. Experiments were performed on a computer with an Intel i7-3960X CPU (3.3 GHz and 6 cores). These performance values are preliminary since we are still working on the speed-up of the parser.

3.3 Analysis

In order to better understand the benefits of the joint model, we performed an error analysis for German and English, where we compared the baseline and the joint model with respect to F-scores for individual part-of-speech categories and dependency labels. For the part-of-speech categories, we found an improvement across the board for both languages, with no category having a significant decrease in F-score, but we also found some interesting patterns for categories that improved more than the average.

Confusion	Baseline		Joint	
	Freq	F-score	Freq	F-score
VVINF → VVFIN	28	91.1	2	97.7
VVINF → VVPP|ADJ*|NN	5		9	
VVFIN → VVINF	43	94.2	5	98.5
VVFIN → VVPP	20		2	
VAINF → VAFIN	10	99.1	1	99.9
NE → NN	184	90.7	128	92.4
NE → ADJ*|ADV|FM	24		18	
NE → XY	12		21	
NN → NE	85	97.5	67	98.1
NN → ADJ*|XY|ADV|VV*	39		29	
PRELS → ART	13	92.9	5	95.4
PRELS → PWS	0		2	

Table 5: Selected entries from the confusion matrix for parts of speech in German with F-scores for the left-hand-side category. ADJ* (ADJD or ADJA) = adjective; ADV = adverb; ART = determiner; APPR = preposition; NE = proper noun; NN = common noun; PRELS = relative pronoun; VVFIN = finite verb; VVINF = non-finite verb; VAFIN = finite auxiliary verb; VAINF = non-finite auxiliary verb; VVPP = participle; XY = not a word. We use α* to denote the set of categories with α as a prefix.

Table 5 shows selected entries from the confusion matrix for German, where we see substantial improvements for finite and non-finite verbs, which are often morphologically ambiguous but which can be disambiguated using syntactic context. We also see improved accuracies for common and proper nouns, which are both capitalized in standard German orthography and therefore often mistagged, and for relative pronouns, which are less often confused for determiners in the joint model.

Table 6 gives a similar snapshot for English, and we again see improvements for verb categories that are often morphologically ambiguous, such as past participles, which can be confused for past tense verbs, and present tense verbs in third person singular, which can be confused for nouns. We also see some improvement for the singular noun catego-

Confusion	Baseline		Joint	
	Freq	F-score	Freq	F-score
VBN → VBD	40	90.5	19	91.5
VBN → JJ|VB|VBP|NN	13		18	
VBZ → NN|NNS	19	97.8	13	98.3
VBZ → POS|JJ|RB	6		6	
NN → VBG|VB|VBN|VBD	72	96.8	58	97.2
NN → JJ|JJR	79		69	
NN → NN*|RB|IN|DT	58		57	
RB → IN	126	92.4	93	92.9
RB → JJ*|RP|NN*|RBR|UH	86		89	

Table 6: Selected entries from the confusion matrix for parts of speech in English with F-scores for the left-hand-side category. DT = determiner; IN = preposition or subordinating conjunction; JJ = adjective; JJR = comparative adjective; NN = singular or mass noun; NNS = plural noun; POS = possessive clitic; RB = adverb; RBR = comparative adverb; RP = particle; UH = interjection; VB = base form verb; VBD = past tense verb; VBG = gerund or present participle; VBN = past participle; VBP = present tense verb, not 3rd person singular; VBZ = present tense verb, 3rd person singular. We use α* to denote the set of categories with α as a prefix.

ry and for adverbs, which are less often confused for prepositions or subordinating conjunctions thanks to the syntactic information in the joint model.

For dependency labels, it is hard to extract any striking patterns and it seems that we mainly see an improvement in overall parsing accuracy thanks to less severe tagging errors. However, it is worth observing that, for both English and German, we see significant F-score improvements for the core grammatical functions subject (91.3 → 92.1 for German, 95.6 → 96.1 for English) and object (86.9 → 87.9 for German, 90.2 → 91.9 for English).

4 Related Work

Our work is most closely related to Lee et al. (2011), Li et al. (2011) and Hatori et al. (2011), who all present discriminative models for joint tagging and dependency parsing. However, all three models only perform unlabeled parsing, while our model incorporates dependency labels into the parsing process. Whereas Lee et al. (2011) and Li et al. (2011) take a graph-based approach to dependency parsing, Hatori et al. (2011) use a transition-based model similar to ours but limited to projective dependency trees. Both Li et al. (2011) and Hatori et al. (2011) only evaluate their model on Chinese, and of these only Hatori et al. (2011) report consistent improvements in both tagging and parsing accuracy. Like our system, the parser of Lee et al. (2011) can handle non-projective trees and experimental results are presented for four languages, but their graph-based model is relatively simple and the baselines therefore well below the state of the art. We are thus the first to show consistent improvements in both tagging and (labeled) parsing accuracy across typologically diverse languages at the state-of-the-art level. Moreover, the capacity to handle non-projective dependencies, which is crucial to attain good performance on Czech and German, does not seem to hurt performance on English and Chinese, where the benchmark sets contain only projective trees.

The use of beam search in transition-based dependency parsing in order to mitigate the problem of error propagation was first proposed by Johansson and Nugues (2006), although they still used a locally trained model. Globally normalized models were first explored by Titov and Henderson (2007), who were also the first to use a parameterized SHIFT transition like the one found in both Hatori et al. (2011) and our own work, although Titov and Henderson (2007) used it to define a generative model by parameterizing the SHIFT transition by an input word. Zhang and Clark (2008) was the first to combine beam search with a globally normalized discriminative model, using structured perceptron learning and the early update strategy of Collins and Roark (2004), and also explored the addition of graph-based features to a transition-based parser. This approach was further pursued in Zhang and Clark (2011) and was used by Zhang and Nivre (2011) to achieve state-of-the-art results in dependency parsing for both Chinese and English through the addition of rich non-local features. Huang and Sagae (2010) combined structured perceptron learning and beam search with the use of a graph-structured stack to allow ambiguity packing in the beam, a technique that was reused by Hatori et al. (2011).

Finally, as noted in the introduction, although joint tagging and parsing is rare in dependency parsing, most state-of-the-art parsers based on PCFG models naturally incorporate part-of-speech tagging and usually achieve better parsing accuracy (albeit not always tagging accuracy) with a joint model than with a pipeline approach (Collins, 1997; Charniak, 2000; Charniak and Johnson, 2005; Petrov et al., 2006). Models that in addition incorporate morphological analysis and segmentation have been explored by Tsarfaty (2006), Cohen and Smith (2007), and Goldberg and Tsarfaty (2008) with special reference to Hebrew parsing.

5 Conclusion

We have presented the first system for joint part-of-speech tagging and labeled dependency parsing with non-projective dependency trees. Evaluation on four languages shows consistent improvements in both tagging and parsing accuracy over a pipeline system with state-of-the-art results across the board. The error analysis reveals improvements in tagging accuracy for syntactically central categories, mainly verbs, with improvement in syntactic accuracy for core grammatical functions as a result. In future work we intend to explore joint models that incorporate not only basic part-of-speech tags but also more fine-grained morphological features.

References

Giuseppe Attardi. 2006. Experiments with a multilanguage non-projective dependency parser. In *Proceedings of CoNLL*, pages 166–170.

Burton H. Bloom. 1970. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13:422–426.

Bernd Bohnet and Jonas Kuhn. 2012. The best of both worlds – a graph-based completion model for transition-based parsers. In *Proceedings of EACL*, pages 77–87.

Bernd Bohnet. 2010. Top accuracy and fast dependency parsing is not a contradiction. In *Proceedings of COLING*, pages 89–97.

Bernd Bohnet. 2011. Comparing advanced graph-based and transition-based dependency parsers. In *Proceedings of the International Conference on Dependency Linguistics (Depling)*, pages 282–289.

Xavier Carreras, Michael Collins, and Terry Koo. 2008. Tag, dynamic programming, and the perceptron for efficient, feature-rich parsing. In *Proceedings of CoNLL*, pages 9–16.

Xavier Carreras. 2007. Experiments with a higher-order projective dependency parser. In *Proceedings of the CoNLL Shared Task of EMNLP-CoNLL 2007*, pages 957–961.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and MaxEnt discriminative reranking. In *Proceedings of ACL*, pages 173–180.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of NAACL*, pages 132–139.

Shay B. Cohen and Noah A. Smith. 2007. Joint morphological and syntactic disambiguation. In *Proceedings of EMNLP-CoNLL*, pages 208–217.

Michael Collins and Brian Roark. 2004. Incremental parsing with the perceptron algorithm. In *Proceedings of ACL*, pages 112–119.

Michael Collins. 1997. Three generative, lexicalised models for statistical parsing. In *Proceedings of ACL-EACL*, pages 16–23.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of EMNLP*, pages 1–8.

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. 2006. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7:551–585.

Andrea Gesmundo, James Henderson, Paola Merlo, and Ivan Titov. 2009. A latent variable model of synchronous syntactic-semantic parsing for multiple languages. In *Proceedings of the 2009 CoNLL Shared Task*, pages 37–42.

Yoav Goldberg and Reut Tsarfaty. 2008. A single generative model for joint morphological segmentation and syntactic parsing. In *Proceedings of ACL*, pages 371–379.

Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, Pavel Straňák, Mihai Surdeanu, Nianwen Xue, and Yi Zhang. 2009. The conll-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proceedings of the 2009 CoNLL Shared Task*, pages 1–18.

Jun Hatori, Takuya Matsuzaki, Yusuke Miyao, and Jun'ichi Tsujii. 2011. Incremental joint pos tagging and dependency parsing in chinese. In *Proceedings of IJCNLP*, pages 1216–1224.

Liang Huang and Kenji Sagae. 2010. Dynamic programming for linear-time incremental parsing. In *Proceedings of ACL*, pages 1077–1086.

Richard Johansson and Pierre Nugues. 2006. Investigating multilingual dependency parsing. In *Proceedings of CoNLL*, pages 206–210.

Terry Koo and Michael Collins. 2010. Efficient third-order dependency parsers. In *Proceedings of ACL*, pages 1–11.

Terry Koo, Xavier Carreras, and Michael Collins. 2008. Simple semi-supervised dependency parsing. In *Proceedings of ACL*, pages 595–603.

Terry Koo, Alexander M. Rush, Michael Collins, Tommi Jaakkola, and David Sontag. 2010. Dual decomposition for parsing with non-projective head automata. In *Proceedings of EMNLP*, pages 1288–1298.

Sandra Kübler, Ryan McDonald, and Joakim Nivre. 2009. *Dependency Parsing*. Morgan and Claypool.

John Lee, Jason Naradowsky, and David A. Smith. 2011. A discriminative model for joint morphological disambiguation and dependency parsing. In *Proceedings of ACL*, pages 885–894.

Zhenghua Li, Min Zhang, Wanxiang Che, Ting Liu, Wenliang Chen, and Haizhou Li. 2011. Joint models for chinese POS tagging and dependency parsing. In *Proceedings of EMNLP*, pages 1180–1191.

Andre Martins, Noah Smith, and Eric Xing. 2009. Concise integer linear programming formulations for dependency parsing. In *Proceedings of ACL-IJCNLP*, pages 342–350.

Andre Martins, Noah Smith, Eric Xing, Pedro Aguiar, and Mario Figueiredo. 2010. Turbo parsers: Dependency parsing by approximate variational inference. In *Proceedings of EMNLP*, pages 34–44.

Ryan McDonald and Fernando Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *Proceedings of EACL*, pages 81–88.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Online large-margin training of dependency parsers. In *Proceedings of ACL*, pages 91–98.

Joakim Nivre, Johan Hall, and Jens Nilsson. 2004. Memory-based dependency parsing. In *Proceedings of CoNLL*, pages 49–56.

Joakim Nivre. 2006. *Inductive Dependency Parsing*. Springer.

Joakim Nivre. 2007. Incremental non-projective dependency parsing. In *Proceedings of NAACL HLT*, pages 396–403.

Joakim Nivre. 2008. Algorithms for deterministic incremental dependency parsing. *Computational Linguistics*, 34:513–553.

Joakim Nivre. 2009. Non-projective dependency parsing in expected linear time. In *Proceedings of ACL-IJCNLP*, pages 351–359.

Slav Petrov and Dan Klein. 2007. Improved inference for unlexicalized parsing. In *Proceedings of NAACL HLT*, pages 404–411.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of COLING/ACL*, pages 433–440.

Sebastian Riedel and James Clarke. 2006. Incremental integer linear programming for non-projective dependency parsing. In *Proceedings of EMNLP*, pages 129–137.

Helmut Schmid and Florian Laws. 2008. Estimation of conditional probabilities with decision trees and an application to fine-grained POS tagging. In *Proceedings of COLING*, pages 777–784.

Qinfeng Shi, James Petterson, Gideon Dror, John Langford, Alex Smola, Alex Strehl, and S V N Vishwanathan. 2009. Hash Kernels for Structured Data. In *Journal of Machine Learning*.

David Smith and Jason Eisner. 2008. Dependency parsing by belief propagation. In *Proceedings of EMNLP*, pages 145–156.

Anders Søgaard. 2011. Semi-supervised condensed nearest neighbor for part-of-speech tagging. In *Proceedings of ACL*, pages 48–52.

Jun Suzuki, Hideki Isozaki, Xavier Carreras, and Michael Collins. 2009. An empirical study of semi-supervised structured conditional models for dependency parsing. In *Proceedings of EMNLP*, pages 551–560.

Ivan Titov and James Henderson. 2007. A latent variable model for generative dependency parsing. In *Proceedings of IWPT*, pages 144–155.

Ivan Titov, James Henderson, Paola Merlo, and Gabriele Musillo. 2009. Online graph planarization for synchronous parsing of semantic and syntactic dependencies. In *Proceedings of IJCAI*.

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of NAACL*, pages 252–259.

Reut Tsarfaty. 2006. Integrated morphological and syntactic disambiguation for modern hebrew. In *Proceedings of the COLING/ACL 2006 Student Research Workshop*, pages 49–54.

Hiroyasu Yamada and Yuji Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proceedings of IWPT*, pages 195–206.

Yue Zhang and Stephen Clark. 2008. A tale of two parsers: Investigating and combining graph-based and transition-based dependency parsing. In *Proceedings of EMNLP*, pages 562–571.

Yue Zhang and Stephen Clark. 2011. Syntactic processing using the generalized perceptron and beam search. *Computational Linguistics*, 37:105–151.

Yue Zhang and Joakim Nivre. 2011. Transition-based parsing with rich non-local features. In *Proceedings of ACL*.