Corrective Modeling for Non-Projective Dependency Parsing

Keith Hall
Center for Language and Speech Processing
Johns Hopkins University
Baltimore, MD 21218
keith_hall@jhu.edu

Václav Novák
Institute of Formal and Applied Linguistics
Charles University
Prague, Czech Republic
novak@ufal.mff.cuni.cz

Abstract

We present a corrective model for recovering non-projective dependency structures from trees generated by state-of-the-art constituency-based parsers. The continuity constraint of these constituency-based parsers makes it impossible for them to posit non-projective dependency trees. Analysis of the types of dependency errors made by these parsers on a Czech corpus show that the correct governor is likely to be found within a local neighborhood of the governor proposed by the parser. Our model, based on a MaxEnt classifier, improves overall dependency accuracy by .7% (a 4.5% reduction in error) with over 50% accuracy for non-projective structures.

1 Introduction

Statistical parsing models have been shown to be successful in recovering labeled constituencies (Collins, 2003; Charniak and Johnson, 2005; Roark and Collins, 2004) and have also been shown to be adequate in recovering dependency relationships (Collins et al., 1999; Levy and Manning, 2004; Dubey and Keller, 2003). The most successful models are based on lexicalized probabilistic context free grammars (PCFGs) induced from constituency-based treebanks. The linear-precedence constraint of these grammars restricts the types of dependency structures that can be encoded in such trees.[1] A shortcoming of the constituency-based paradigm for parsing is that it is inherently incapable of representing non-projective dependencies trees (we define non-projectivity in the following section). This is particularly problematic when parsing free word-order languages, such as Czech, due to the frequency of sentences with non-projective constructions.

In this work, we explore a corrective model which recovers non-projective dependency structures by training a classifier to select correct dependency pairs from a set of candidates based on parses generated by a constituency-based parser. We chose to use this model due to the observations that the dependency errors made by the parsers are generally local errors. For the nodes with incorrect dependency links in the parser output, the correct governor of a node is often found within a local context of the proposed governor. By considering alternative dependencies based on local deviations of the parser output we constrain the set of candidate governors for each node during the corrective procedure. We examine two state-of-the-art constituency-based parsers in this work: the Collins Czech parser (1999) and a version of the Charniak parser (2001) that was modified to parse Czech.

Alternative efforts to recover dependency structure from English are based on reconstructing the movement *traces* encoded in constituency trees (Collins, 2003; Levy and Manning, 2004; Johnson, 2002; Dubey and Keller, 2003). In fact, the fea-

---

[1] In order to correctly capture the dependency structure, co-indexed movement *traces* are used in a form similar to government and Binding theory, GPSG, etc.

Figure 1: Examples of projective and non-projective trees. The trees on the left and center are both projective. The tree on the right is non-projective.

tures we use in the current model are similar to those proposed by Levy and Manning (2004). However, the approach we propose discards the constituency structure prior to the modeling phase; we model corrective transformations of dependency trees.

The technique proposed in this paper is similar to that of recent parser reranking approaches (Collins, 2000; Charniak and Johnson, 2005); however, while reranking approaches allow a parser to generate a likely candidate set according to a generative model, we consider a set of candidates based on local perturbations of the single most likely tree generated. The primary reason for such an approach is that we allow dependency structures which would never be hypothesized by the parser. Specifically, we allow for non-projective dependencies.

The corrective algorithm proposed in this paper shares the motivation of the transformation-based learning work (Brill, 1995). We do consider local transformations of the dependency trees; however, the technique presented here is based on a generative model that maximizes the likelihood of good dependents. We consider a finite set of local perturbations of the tree and use a fixed model to select the best tree by independently choosing optimal dependency links.

In the remainder of the paper we provide a definition of a dependency tree and the motivation for using such trees as well as a description of the particular dataset that we use in our experiments, the Prague Dependency Treebank (PDT). In Section 3 we describe the techniques used to adapt constituency-based parsers to train from and generate dependency trees. Section 4 describes corrective modeling as used in this work and Section 4.2 describes the particular features with which we have experimented. Section 5 presents the results of a set of experiments we performed on data from the PDT.

2 Syntactic Dependency Trees and the Prague Dependency Treebank

A dependency tree is a set of nodes $\Omega = \{w_0, w_1, \ldots, w_k\}$ where w_0 is the imaginary root node[2] and a set of dependency links $G = \{g_1, \ldots, g_k\}$ where g_i is an index into Ω representing the governor of w_i. In other words $g_3 = 1$ indicates that the governor of w_3 is w_1. Finally, every node has exactly one governor except for w_0, which has no governor (the tree constraints).[3] The index of the nodes represents the surface order of the nodes in the sequence (i.e., w_i precedes w_j in the sentence if $i < j$).

A tree is *projective* if for every three nodes: w_a, w_b, and w_c where $a < b < c$; if w_a is governed by w_c then w_b is transitively governed by w_c or if w_c is governed by w_a then w_b is transitively governed by w_a.[4] Figure 1 shows examples of projective and non-projective trees. The rightmost tree, which is non-projective, contains a subtree consisting of w_a and w_c but not w_b; however, w_b occurs between w_a and w_c in the linear ordering of the nodes. Projectivity in a dependency tree is akin to the continuity constraint in a constituency tree; such a constraint is

---

[2] The imaginary root node simplifies notation.

[3] The dependency structures here are very similar to those described by Mel'čuk (1988); however the nodes of the dependency trees discussed in this paper are limited to the words of the sentence and are always ordered according to the surface word-order.

[4] Node w_a is said to transitively govern node w_b if w_b is a descendant of w_a in the dependency tree.

implicitly imposed by trees generated from context free grammars (CFGs).

Strict word-order languages, such as English, exhibit non-projective dependency structures in a relatively constrained set of syntactic configurations (e.g., right-node raising). Traditionally, these movements are encoded in syntactic analyses as *traces*. In languages with free word-order, such as Czech, constituency-based representations are overly constrained (Sgall et al., 1986). Syntactic dependency trees encode syntactic subordination relationships allowing the structure to be non-specific about the *underlying deep representation*. The relationship between a node and its subordinates expresses a sense of syntactic (functional) entailment.

In this work we explore the dependency structures encoded in the Prague Dependency Treebank (Hajič, 1998; Böhmová et al., 2002). The PDT 1.0 analytical layer is a set of Czech syntactic dependency trees; the nodes of which contain the word forms, morphological features, and syntactic annotations. These trees were annotated by hand and are intended as an intermediate stage in the annotation of the Tectogrammatical Representation (TR), a *deep-syntactic* or syntacto-semantic theory of language (Sgall et al., 1986). All current automatic techniques for generating TR structures are based on syntactic dependency parsing.

When evaluating the correctness of dependency trees, we only consider the structural relationships between the words of the sentence (unlabeled dependencies). However, the model we propose contains features that are considered part of the dependency rather than the nodes in isolation (e.g., agreement features). We do not propose a model for correctly labeling dependency structures in this work.

3 Constituency Parsing for Dependency Trees

A pragmatic justification for using constituency-based parsers in order to predict dependency structures is that currently the best Czech dependency-tree parser is a constituency-based parser (Collins et al., 1999; Zeman, 2004). In fact both Charniak's and Collins' generative probabilistic models contain lexical dependency features.[5] From a generative modeling perspective, we use the constraints imposed by constituents (i.e., projectivity) to enable the encapsulation of syntactic substructures. This directly leads to efficient parsing algorithms such as the CKY algorithm and related agenda-based parsing algorithms (Manning and Schütze, 1999). Additionally, this allows for the efficient computation of the scores for the dynamic-programming state variables (i.e., the inside and outside probabilities) that are used in efficient statistical parsers. The computational complexity advantages of dynamic programming techniques along with efficient search techniques (Caraballo and Charniak, 1998; Klein and Manning, 2003) allow for richer predictive models which include local contextual information.

In an attempt to extend a constituency-based parsing model to train on dependency trees, Collins transforms the PDT dependency trees into constituency trees (Collins et al., 1999). In order to accomplish this task, he first normalizes the trees to remove non-projectivities. Then, he creates artificial constituents based on the parts-of-speech of the words associated with each dependency node. The mapping from dependency tree to constituency tree is not one-to-one. Collins describes a heuristic for choosing trees that work well with his parsing model.

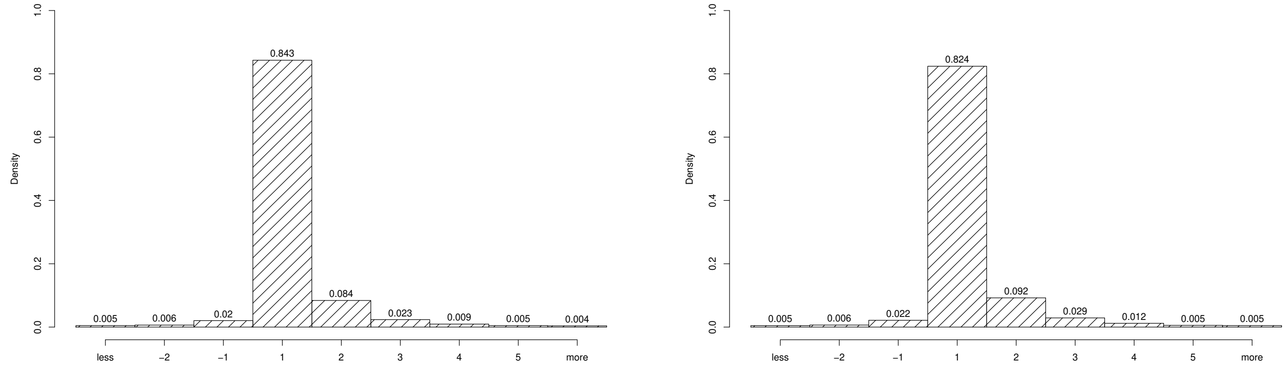
3.1 Training a Constituency-based Parser

We consider two approaches to creating projective trees from dependency trees exhibiting non-projectivities. The first is based on word-reordering and is the model that was used with the Collins parser. This algorithm identifies non-projective structures and deterministically reorders the words of the sentence to create projective trees. An alternative method, used by Charniak in the adaptation of his parser for Czech[6] and used by Nivre and Nilsson (2005), alters the dependency links by raising the governor to a higher node in the tree whenever

---

[5] Bilexical dependencies are components of both the Collins and Charniak parsers and effectively model the types of syntactic subordination that we wish to extract in a dependency tree. (Bilexical models were also proposed by Eisner (Eisner, 1996)). In the absence of lexicalization, both parsers have dependency features that are encoded as head-constituent to sibling features.

[6] This information was provided by Eugene Charniak in a personal communication.



$(a) Charniak$ $(b) Collins$

Figure 2: Statistical distribution of correct governor positions in the Charniak (left) and Collins (right) parser output of parsed PDT development data.

a non-projectivity is observed. The trees are then transformed into Penn Treebank style constituencies using the technique described in (Collins et al., 1999).

Both of these techniques have advantages and disadvantages which we briefly outline here:

Reordering The dependency structure is preserved, but the training procedure will learn statistics for structures over word-strings that may not be part of the language. The parser, however, may be capable of constructing parses for any string of words if a smoothed grammar is being used.

Governor–Raising The dependency structure is corrupted leading the parser to incorporate arbitrary dependency statistics into the model. However, the parser is trained on true sentences, the words of which are in the correct linear order. We expect the parser to predict similar incorrect dependencies when sentences similar to the training data are observed.

Although the results presented in (Collins et al., 1999) used the reordering technique, we have experimented with his parser using the governor–raising technique and observe an increase in dependency accuracy. For the remainder of the paper, we assume the governor–raising technique.

The process of generating dependency trees from parsed constituency trees is relatively straightforward. Both the Collins and Charniak parsers provide head-word annotation on each constituent. This is precisely the information that we encode in an unlabeled dependency tree, so the dependency structure can simply be extracted from the parsed constituency trees. Furthermore, the constituency labels can be used to identify the dependency labels; however, we do not attempt to identify correct dependency labels in this work.

3.2 Constituency-based errors

We now discuss a quantitative measure for the types of dependency errors made by constituency-based parsing techniques. For node w_i and the correct governor $w_{g_i^*}$ the *distance* between the two nodes in the hypothesized dependency tree is:

$$dist(w_i, w_{g_i^*}) = \begin{cases} d(w_i, w_{g_i^*}) & \text{iff } w_{g_i^*} \text{ is ancestor of } w_i \\ d(w_i, w_{g_i^*}) & \text{iff } w_{g_i^*} \text{ is sibling/cousin of } w_i \\ -d(w_i, w_{g_i^*}) & \text{iff } w_{g_i^*} \text{ is descendant of } w_i \end{cases}$$

Ancestor, sibling, cousin, and descendant have the standard interpretation in the context of a tree. The dependency distance $d(w_i, w_{g_i^*})$ is the minimum number of dependency links traversed on the undirected path from w_i to $w_{g_i^*}$ in the hypothesized dependency tree. The definition of the *dist* function makes a distinction between paths through the parent of w_i (positive values) and paths through chil-

```
CORRECT($W$)
Parse sentence $W$ using the constituency-based parser
Generate a dependency structure from the constituency tree
for $w_i \in W$
do for $w_c \in \mathcal{N}(w_{g_i^h})$        // Local neighborhood of proposed governor
    do $l(c) \leftarrow P(g_i^* = c | w_i, \mathcal{N}(w_{g_i^h}))$
        $g_i^{'} \leftarrow \arg\max_c l(c)$        // Pick the governor in which we are most confident
```

Table 1: Corrective Modeling Procedure

dren of w_i (negative values). We found that a vast majority of the correct governors were actually hypothesized as siblings or grandparents (a *dist* values of 2) – an extreme local error.

Figure 2 shows a histogram of the fraction of nodes whose correct governor was within a particular *dist* in the hypothesized tree. A *dist* of 1 indicates the correct governor was selected by the parser; in these graphs, the density at *dist* $= 1$ (on the x axis) shows the baseline dependency accuracy of each parser. Note that if we repaired only the nodes that are within a *dist* of 2 (grandparents and siblings), we can recover more than 50% of the incorrect dependency links (a raw accuracy improvement of up to 9%). We believe this distribution to be indirectly caused by the governor raising projectivization routine. In the cases where non-projective structures can be repaired by raising the node's governor to its parent, the correct governor becomes a sibling of the node.

4 Corrective Modeling

The error analysis of the previous section suggests that by looking only at a local neighborhood of the proposed governor in the hypothesized trees, we can correct many of the incorrect dependencies. This fact motivates the corrective modeling procedure employed here.

Table 1 presents the pseudo-code for the corrective procedure. The set g^h contains the indices of governors as predicted by the parser. The set of governors predicted by the corrective procedure is denoted as $g^{'}$. The procedure independently corrects each node of the parsed trees meaning that there is potential for inconsistent governor relationships to exist in the proposed set; specifically, the resulting dependency graph may have cycles. We employ a greedy search to remove cycles when they are present in the output graph.

The final line of the algorithm picks the governor in which we are most confident. We use the correct-governor classification likelihood, $P(g_i^* = j | w_i, \mathcal{N}(w_{g_i^h}))$, as a measure of the confidence that w_c is the correct governor of w_i where the parser had proposed $w_{g_i^h}$ as the governor. In effect, we create a decision list using the most likely decision if we can (i.e., there are no cycles). If the dependency graph resulting from the most likely decisions does not result in a tree, we use the decision lists to greedily select the tree for which the product of the independent decisions is maximal.

Training the corrective model requires pairs of dependency trees; each pair contains a manually-annotated tree (i.e., the gold standard tree) and a tree generated by the parser. This data is trivially transformed into per-node samples. For each node w_i in the tree, there are $|\mathcal{N}(w_{g_i^h})|$ samples; one for each governor candidate in the local neighborhood.

One advantage to the type of corrective algorithm presented here is that it is completely disconnected from the parser used to generate the tree hypotheses. This means that the original parser need not be statistical or even constituency based. What is critical for this technique to work is that the distribution of dependency errors be relatively local as is the case with the errors made by the Charniak and Collins parsers. This can be determined via data analysis using the *dist* metric. Determining the size of the local neighborhood is data dependent. If subordinate nodes are considered as candidate governors, then a more robust cycle removal technique is be required.

4.1 MaxEnt Estimation

We have chosen a MaxEnt model to estimate the governor distributions, $P(g_i^* = j | w_i, \mathcal{N}(w_{g_i^h}))$. In the next section we outline the feature set with which we have experimented, noting that the features are selected based on linguistic intuition (specifically for Czech). We choose not to factor the feature vector as it is not clear what constitutes a reasonable factorization of these features. For this reason we use the MaxEnt estimator which provides us with the flexibility to incorporate interdependent features independently while still optimizing for likelihood.

The maximum entropy principle states that we wish to find an estimate of $p(y|x) \in \mathcal{C}$ that maximizes the entropy over a sample set X for some set of observations Y, where $x \in X$ is an observation and $y \in Y$ is a outcome label assigned to that observation,

$$H(p) \equiv - \sum_{x \in X, y \in Y} \tilde{p}(x) p(y|x) \log p(y|x)$$

The set $\mathcal{C}$ is the candidate set of distributions from which we wish to select $p(y|x)$. We define this set as the $p(y|x)$ that meets a feature-based expectation constraint. Specifically, we want the expected count of a feature, $f(x, y)$, to be equivalent under the distribution $p(y|x)$ and under the observed distribution $\tilde{p}(y|x)$.

$$\begin{aligned} &\sum_{x \in X, y \in Y} \tilde{p}(x) p(y|x) f_i(x, y) \\ &\quad = \sum_{x \in X, y \in Y} \tilde{p}(x) \tilde{p}(y|x) f_i(x, y) \end{aligned}$$

$f_i(x, y)$ is a feature of our model with which we capture correlations between observations and outcomes. In the following section, we describe a set of features with which we have experimented to determine when a word is likely to be the correct governor of another word.

We incorporate the expected feature-count constraints into the maximum entropy objective using Lagrange multipliers (additionally, constraints are added to ensure the distributions $p(y|x)$ are consistent probability distributions):

$$\begin{aligned} & H(p) \\ & + \sum_i \alpha_i \sum_{x \in X, y \in Y} \Big(\tilde{p}(x) p(y|x) f_i(x, y) \\ & \qquad\qquad - \tilde{p}(x) \tilde{p}(y|x) f_i(x, y) \Big) \\ & + \gamma \sum_{y \in Y} p(y|x) - 1 \end{aligned}$$

Holding the α_i's constant, we compute the unconstrained maximum of the above Lagrangian form:

$$\begin{aligned} p_\alpha(y|x) &= \frac{1}{Z_\alpha(x)} \exp(\sum_i \alpha_i f_i(x, y)) \\ Z_\alpha(x) &= \sum_{y \in Y} \exp(\sum_i \alpha_i f_i(x, y)) \end{aligned}$$

giving us the log-linear form of the distributions $p(y|x)$ in $\mathcal{C}$ (Z is a normalization constant). Finally, we compute the α_i's that maximize the objective function:

$$- \sum_{x \in X} \tilde{p}(x) \log Z_\alpha(x) + \sum_i \alpha_i \tilde{p}(x, y) f_i(x, y)$$

A number of algorithms have been proposed to efficiently compute the optimization described in this derivation. For a more detailed introduction to maximum entropy estimation see (Berger et al., 1996).

4.2 Proposed Model

Given the above formulation of the MaxEnt estimation procedure, we define features over pairs of observations and outcomes. In our case, the observations are simply w_i, w_c, and $\mathcal{N}(w_{g_i^h})$ and the outcome is a binary variable indicating whether $c = g_i^*$ (i.e., w_c is the correct governor). In order to limit the dimensionality of the feature space, we consider feature functions over the outcome, the current node w_i, the candidate governor node w_c and the node proposed as the governor by the parser $w_{g_i^h}$.

Table 2 describes the general classes of features used. We write F_i to indicate the form of the current child node, F_c for the form of the candidate, and F_g as the form of the governor proposed by the parser. A combined feature is denoted as $L_i T_c$ and indicates we observed a particular lemma for the current node with a particular tag of the candidate.

Feature Type	Id	Description
Form	F	the fully inflected word form as it appears in the data
Lemma	L	the morphologically reduced lemma
MTag	T	a subset of the morphological tag as described in (Collins et al., 1999)
POS	P	major part-of-speech tag (first field of the morphological tag)
ParserGov	G	true if candidate was proposed as governor by parser
ChildCount	C	the number of children
Agreement	$A(x, y)$	check for case/number agreement between word x and y

Table 2: Description of the classes of features used

In all models, we include features containing the form, the lemma, the morphological tag, and the ParserGov feature. We have experimented with different sets of feature combinations. Each combination set is intended to capture some intuitive linguistic correlation. For example, the feature component $L_i T_c$ will fire if a particular child's lemma L_i is observed with a particular candidate's morphological tag T_c. This feature is intended to capture phenomena surrounding particles; for example, in Czech, the governor of the reflexive particle *se* will likely be a verb.

4.3 Related Work

Recent work by Nivre and Nilsson introduces a technique where the projectivization transformation is encoded in the non-terminals of constituents during parsing (Nivre and Nilsson, 2005). This allows for a deterministic procedure that undoes the projectivization in the generated parse trees, creating non-projective structures. This technique could be incorporated into a statistical parsing framework, however we believe the sparsity of such non-projective configurations may be problematic when using smoothed backed-off grammars. We suspect that the deterministic procedure employed by Nivre and Nilsson enables their parser to greedily consider non-projective constructions when possible. This may also explain the relatively low overall performance of their parser.

A primary difference between the Nivre and Nilsson approach and what we propose in this paper is that of determining the projectivization procedure. While we exploit particular side-effects of the projectivization procedure, we do not assume any particular algorithm. Additionally, we consider transformations for all dependency errors where their technique explicitly addresses non-projectivity errors.

We mentioned above that our approach appears to be similar to that of reranking for statistical parsing (Collins, 2000; Charniak and Johnson, 2005). While it is true that we are improving upon the output of the automatic parser, we are not considering multiple alternate parses. Instead, we consider a complete set of alternate trees that are minimal perturbations of the best tree generated by the parser. In the context of dependency parsing, we do this in order to generate structures that constituency-based parsers are incapable of generating (i.e., non-projectivities).

Recent work by Smith and Eisner (2005) on *contrastive estimation* suggests similar techniques to generate local neighborhoods of a parse; however, the purpose in their work is to define an approximation to the partition function for log-linear estimation (i.e., the normalization factor in a MaxEnt model).

5 Empirical Results

In this section we report results from experiments on the PDT Czech dataset. Approximately 1.9% of the words' dependencies are non-projective in version 1.0 of this corpus and these occur in 23.2% of the sentences (Hajičová et al., 2004). We used the standard training, development, and evaluation datasets defined in the PDT documentation for all experiments.[7] We use Zhang Lee's implementation of the

---

[7] We have used PDT 1.0 (2002) data for the Charniak experiments and PDT 2.0 (2005) data for the Collins experiments. We use the most recent version of each parser; however we do not have a training program for the Charniak parser and have used the pretrained parser provided by Charniak; this was trained on the training section of the PDT 1.0. We train our model on the

Model	Features	Description
Count	ChildCount	count of children for the three nodes
MTagL	$T_iT_c, L_iL_c, L_iT_c, T_iL_c, T_iP_g$	conjunctions of MTag and Lemmas
MTagF	$T_iT_c, F_iF_c, F_iT_c, T_iF_c, T_iP_g$	conjunctions of MTag and Forms
POSL	$P_i, P_c, P_g, P_iP_cP_g, P_iP_g, P_cL_c$	conjunctions of POS and Lemma
TTT	$T_iT_cT_g$	conjunction of tags for each of the three nodes
Agr	$A(T_i, T_c), A(T_i, T_g)$	binary feature if case/number agree
Trig	$L_iL_gT_c, T_iL_gT_c, L_iL_gL_c$	trigrams of Lemma/Tag

Table 3: Model feature descriptions.

Model	Charniak Parse Trees		Collins Parse Trees	
	Devel. Accuracy	NonP Accuracy	Devel. Accuracy	NonP Accuracy
Baseline	84.3%	15.9%	82.4%	12.0%
Simple	84.3%	16.0%	82.5%	12.2%
Simple + Count	84.3%	16.7%	82.5%	13.8%
Simple + MtagL	84.8%	43.5%	83.2%	44.1%
Simple + MtagF	84.8%	42.2%	83.2%	43.2%
Simple + POS	84.3%	16.0%	82.4%	12.1%
Simple + TTT	84.3%	16.0%	82.5%	12.2%
Simple + Agr	84.3%	16.2%	82.5%	12.2%
Simple + Trig	84.9%	47.9%	83.1%	47.7%
All Features	**85.0%**	**51.9%**	**83.5%**	**57.5%**

Table 4: Comparative results for different versions of our model on the Charniak and Collins parse trees for the PDT development data.

MaxEnt estimator using the L-BFGS optimization algorithms and Gaussian smoothing.[8]

Table 4 presents results on development data for the correction model with different feature sets. The features of the **Simple** model are the form (F), lemma (L), and morphological tag (M) for the each node, the parser-proposed governor node, and the candidate node; this model also contains the ParserGov feature. In the table's following rows, we show the results for the simple model augmented with feature sets of the categories described in Table 2. Table 3 provides a short description of each of the models. As we believe the **Simple** model provides the minimum information needed to perform this task, we experimented with the feature-classes as additions to it. The final row of Table 4 contains results for the model which includes all features from all other models.

We define **NonP Accuracy** as the accuracy for the nodes which were non-projective in the original trees. Although both the Charniak and the Collins parser can never produce non-projective trees, the baseline NonP accuracy is greater than zero. This is due to the parser making mistakes in the tree such that the originally non-projective node's dependency is projective.

Alternatively, we report the Non-Projective Precision and Recall for our experiment suite in Table 5. Here the numerator of the precision is the number of nodes that are non-projective in the correct tree and end up in a non-projective configuration; however, this new configuration may be based on incorrect dependencies. Recall is the obvious counterpart to precision. These values correspond to the NonP

Collins trees via a 20-fold Jackknife training procedure.

[8] Using held-out development data, we determined a Gaussian prior parameter setting of 4 worked best. The optimal number of training iterations was chosen on held-out data for each experiment. This was generally in the order of a couple hundred iterations of L-BFGS. The MaxEnt modeling implementation can be found at `http://homepages.inf.ed.ac.uk/s0450736/maxent_toolkit.html`.

Model	Charniak Parse Trees			Collins Parse Trees		
	Precision	Recall	F-measure	Precision	Recall	F-measure
Baseline	N/A	0.0%	0.000	N/A	0.0%	0.000
Simple	22.6%	0.3%	0.592	5.0%	0.2%	0.385
Simple + Count	37.3%	1.1%	2.137	16.8%	2.0%	3.574
Simple + MtagL	78.0%	29.7%	43.020	62.4%	35.0%	44.846
Simple + MtagF	78.7%	28.6%	41.953	62.0%	34.3%	44.166
Simple + POS	23.3%	0.3%	0.592	2.5%	0.1%	0.192
Simple + TTT	20.7%	0.3%	0.591	6.1%	0.2%	0.387
Simple + Agr	40.0%	0.5%	0.988	5.7%	0.2%	0.386
Simple + Trig	74.6%	35.0%	47.646	52.3%	40.2%	45.459
All Features	75.7%	39.0%	51.479	48.1%	51.6%	49.789

Table 5: Alternative non-projectivity scores for different versions of our model on the Charniak and Collins parse trees.

accuracy results reported in Table 4. From these tables, we see that the most effective features (when used in isolation) are the conjunctive MTag/Lemma, MTag/Form, and Trigram MTag/Lemma features.

Model	Dependency Accuracy	NonP Accuracy
Collins	81.6%	N/A
Collins + Corrective	**82.8%**	**53.1%**
Charniak	84.4%	N/A
Charniak + Corrective	**85.1%**	**53.9%**

Table 6: Final results on PDT evaluation datasets for Collins' and Charniak's trees with and without the corrective model

Finally, Table 6 shows the results of the full model run on the evaluation data for the Collins and Charniak parse trees. It appears that the Charniak parser fares better on the evaluation data than does the Collins parser. However, the corrective model is still successful at recovering non-projective structures. Overall, we see a significant improvement in the dependency accuracy.

Top Five Good/Bad Repairs	
Well repaired child	se i si až jen
Well repaired false governor	v však li na o
Well repaired real governor	a je stát ba ,
Poorly repaired child	, se na že -
Poorly repaired false governor	a , však musí li
Poorly repaired real governor	*root* sklo , je -

Table 7: Categorization of corrections and errors made by our model on trees from the Charniak parser. *root* is the artificial root node of the PDT tree. For each node position (child, proposed parent, and correct parent), the top five words are reported (based on absolute count of occurrences). The particle 'se' occurs frequently explaining why it occurs in the top five good and top five bad repairs.

	Charniak	Collins
Correct to incorrect	13.0%	20.0%
Incorrect to incorrect	21.6%	25.8%
Incorrect to correct	65.5%	54.1%

Table 8: Categorization of corrections made by our model on Charniak and Collins trees.

We have performed a review of the errors that the corrective process makes and observed that the model does a poor job dealing with punctuation. This is shown in Table 7 along with other types of nodes on which we performed well and poorly, respectively. Collins (1999) explicitly added features to his parser to improve punctuation dependency parsing accuracy. The PARSEVAL evaluation metric for constituency-based parsing explicitly ignores punctuation in determining the correct boundaries of constituents (Harrison et al., 1991) and so should the dependency evaluation. However, the reported results include punctuation for comparative purposes. Finally, we show in Table 8 a coarse analysis of the corrective performance of our model. We are repairing more dependencies than we are corrupting.

6 Conclusion

We have presented a Maximum Entropy-based corrective model for dependency parsing. The goal is to recover non-projective dependency structures that are lost when using state-of-the-art constituency-based parsers; we show that our technique recovers over 50% of these dependencies. Our algorithm provides a simple framework for corrective modeling of dependency trees, making no prior assumptions about the trees. However, in the current model, we focus on trees with local errors. Overall, our technique improves dependency parsing and provides the necessary mechanism to recover non-projective structures.

References

Adam L. Berger, Stephen A. Della Pietra, and Vincent J. Della Pietra. 1996. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71.

Alena Böhmová, Jan Hajič, Eva Hajičová, and Barbora Vidová Hladká. 2002. The prague dependency treebank: Three-level annotation scenario. In Anne Abeille, editor, *In Treebanks: Building and Using Syntactically Annotated Corpora*. Dordrecht, Kluwer Academic Publishers, The Neterlands.

Eric Brill. 1995. Transformation-based error-driven learning and natural language processing: A case study in part of speech tagging. *Computational Linguistics*, 21(4):543–565, December.

Sharon Caraballo and Eugene Charniak. 1998. New figures of merit for best-first probabilistic chart parsing. *Computational Linguistics*, 24(2):275–298, June.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*.

Eugene Charniak. 2001. Immediate-head parsing for language models. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*.

Michael Collins, Lance Ramshaw, Jan Hajič, and Christoph Tillmann. 1999. A statistical parser for czech. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics*, pages 505–512.

Michael Collins. 2000. Discriminative reranking for natural language parsing. In *Proceedings of the 17th International Conference on Machine Learning 2000*.

Michael Collins. 2003. Head-driven statistical models for natural language processing. *Computational Linguistics*, 29(4):589–637.

Amit Dubey and Frank Keller. 2003. Probabilistic parsing for German using sister-head dependencies. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 96–103, Sapporo.

Jason Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING)*, pages 340–345.

Jan Hajič, Eva Hajičová, Petr Pajas, Jarmila Panevová, Petr Sgall, and Barbora Vidová Hladká. 2005. The prague dependency treebank 2.0. http://ufal.mff.cuni.cz/pdt2.0.

Jan Hajič. 1998. Building a syntactically annotated corpus: The prague dependency treebank. In *Issues of Valency and Meaning*, pages 106–132. Karolinum, Praha.

Eva Hajičová, Jiří Havelka, Petr Sgall, Kateřina Veselá, and Daniel Zeman. 2004. Issues of projectivity in the prague dependency treebank. *Prague Bulletin of Mathematical Linguistics*, 81:5–22.

P. Harrison, S. Abney, D. Fleckenger, C. Gdaniec, R. Grishman, D. Hindle, B. Ingria, M. Marcus, B. Santorini, , and T. Strzalkowski. 1991. Evaluating syntax performance of parser/grammars of english. In *Proceedings of the Workshop on Evaluating Natural Language Processing Systems, ACL.*

Mark Johnson. 2002. A simple pattern-matching algorithm for recovering empty nodes and their antecedents. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*.

Dan Klein and Christopher D. Manning. 2003. Factored A* search for models over sequences and trees. In *Proceedings of IJCAI 2003*.

Roger Levy and Christopher Manning. 2004. Deep dependencies from context-free statistical parsers: Correcting the surface dependency approximation. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, pages 327–334, Barcelona, Spain.

Christopher D. Manning and Hinrich Schütze. 1999. *Foundations of statistical natural language processing*. MIT Press.

Igor Mel'čuk. 1988. *Dependency Syntax: Theory and Practice*. SUNY Press, Albany, NY.

Joakim Nivre and Jens Nilsson. 2005. Pseudo-projective dependency parsing. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pages 99–106, Ann Arbor.

Brian Roark and Michael Collins. 2004. Incremental parsing with the perceptron algorithm. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, Barcelona.

Petr Sgall, Eva Hajičová, and Jarmila Panevová. 1986. *The Meaning of the Sentence in Its Semantic and Pragmatic Aspects*. Kluwer Academic, Boston.

Noah A. Smith and Jason Eisner. 2005. Contrastive estimation: Training log-linear models on unlabeled data. In *Proceedings of the Association for Computational Linguistics (ACL 2005)*, Ann Arbor, Michigan.

Daniel Zeman. 2004. *Parsing with a statistical dependency model*. Ph.D. thesis, Charles University in Prague.