Global Transition-based Non-projective Dependency Parsing

Carlos Gómez-Rodríguez
Universidade da Coruña
carlos.gomez@udc.es

Tianze Shi
Cornell University
tianze@cs.cornell.edu

Lillian Lee
Cornell University
llee@cs.cornell.edu

Abstract

Shi, Huang, and Lee (2017a) obtained state-of-the-art results for English and Chinese dependency parsing by combining dynamic-programming implementations of transition-based dependency parsers with a minimal set of bidirectional LSTM features. However, their results were limited to projective parsing. In this paper, we extend their approach to support non-projectivity by providing the first practical implementation of the MH_4 algorithm, an $O(n^4)$ mildly non-projective dynamic-programming parser with very high coverage on non-projective treebanks. To make MH_4 compatible with minimal transition-based feature sets, we introduce a transition-based interpretation of it in which parser items are mapped to sequences of transitions. We thus obtain the first implementation of global decoding for non-projective transition-based parsing, and demonstrate empirically that it is more effective than its projective counterpart in parsing a number of highly non-projective languages.

1 Introduction

Transition-based dependency parsers are a popular approach to natural language parsing, as they achieve good results in terms of accuracy and efficiency (Yamada and Matsumoto, 2003; Nivre and Scholz, 2004; Zhang and Nivre, 2011; Chen and Manning, 2014; Dyer et al., 2015; Andor et al., 2016; Kiperwasser and Goldberg, 2016). Until very recently, practical implementations of transition-based parsing were limited to approximate inference, mainly in the form of greedy search or beam search. While cubic-time exact inference algorithms for several well-known projective transition systems had been known since the work of Huang and Sagae (2010) and Kuhlmann et al. (2011), they had been considered of theoretical interest only due to their incompatibility with rich feature models: incorporation of complex features resulted in jumps in asymptotic runtime complexity to impractical levels.

However, the recent popularization of bidirectional long-short term memory networks (bi-LSTMs; Hochreiter and Schmidhuber, 1997) to derive feature representations for parsing, given their capacity to capture long-range information, has demonstrated that one may not need to use complex feature models to obtain good accuracy (Kiperwasser and Goldberg, 2016; Cross and Huang, 2016). In this context, Shi et al. (2017a) presented an implementation of the exact inference algorithms of Kuhlmann et al. (2011) with a minimal set of only two bi-LSTM-based feature vectors. This not only kept the complexity cubic, but also obtained state-of-the-art results in English and Chinese parsing.

While their approach provides both accurate parsing and the flexibility to use any of greedy, beam, or exact decoding with the same underlying transition systems, it does not support non-projectivity. Trees with crossing dependencies make up a significant portion of many treebanks, going as high as 63% for the Ancient Greek treebank in the Universal Dependencies[1] (UD) dataset version 2.0 and averaging around 12% over all languages in UD 2.0. In this paper, we extend Shi et al.'s (2017a) approach to mildly non-projective parsing in what, to our knowledge, is the first implementation of exact decoding for a non-projective transition-based parser.

As in the projective case, a mildly non-

[1] http://universaldependencies.org/

Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, pages 1–12
Melbourne, Australia, July 15 - 20, 2018. ©2018 Association for Computational Linguistics

projective decoder has been known for several years (Cohen et al., 2011), corresponding to a variant of the transition-based parser of Attardi (2006). However, its $O(n^7)$ runtime — or the $O(n^6)$ of a recently introduced improved-coverage variant (Shi et al., 2018) — is still prohibitively costly in practice. Instead, we seek a more efficient algorithm to adapt, and thus develop a transition-based interpretation of Gómez-Rodríguez et al.'s (2011) MH_4 dynamic programming parser, which has been shown to provide very good non-projective coverage in $O(n^4)$ time (Gómez-Rodríguez, 2016). While the MH_4 parser was originally presented as a non-projective generalization of the dynamic program that later led to the arc-hybrid transition system (Gómez-Rodríguez et al., 2008; Kuhlmann et al., 2011), its own relation to transition-based parsing was not known. Here, we show that MH_4 can be interpreted as exploring a subset of the search space of a transition-based parser that generalizes the arc-hybrid system, under a mapping that differs from the "push computation" paradigm used by the previously-known dynamic-programming decoders for transition systems. This allows us to extend Shi et al. (2017a)'s work to non-projective parsing, by implementing MH_4 with a minimal set of transition-based features.

Experimental results show that our approach outperforms the projective approach of Shi et al. (2017a) and maximum-spanning-tree non-projective parsing on the most highly non-projective languages in the CoNLL 2017 shared-task data that have a single treebank. We also compare with the third-order 1-Endpoint-Crossing (1EC) parser of Pitler (2014), the only other practical implementation of an exact mildly non-projective decoder that we know of, which also runs in $O(n^4)$ but without a transition-based interpretation. We obtain comparable results for these two algorithms, in spite of the fact that the MH_4 algorithm is notably simpler than 1EC. The MH_4 parser remains effective in parsing projective treebanks, while our baseline parser, the fully non-projective maximum spanning tree algorithm, falls behind due to its unnecessarily large search space in parsing these languages. Our code, including our re-implementation of the third-order 1EC parser with neural scoring, is available at `https://github.com/tzshi/mh4-parser-acl18`.

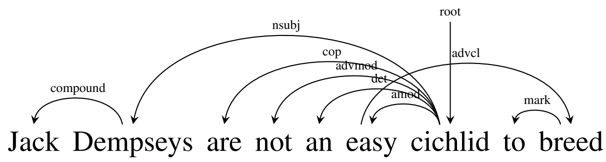


Figure 1: A non-projective dependency parse from the UD 2.0 English treebank.

2 Non-projective Dependency Parsing

In dependency grammar, syntactic structures are modeled as word-word asymmetrical subordinate relations among lexical entries (Kübler et al., 2009). These relations can be represented in a graph. For a sentence $w = w_1, ..., w_n$, we first define a corresponding set of nodes $\{0, 1, 2, ..., n\}$, where 0 is an artificial node denoting the root of the sentence. Dependency relations are encoded by edges of the form (h, m), where h is the head and m the modifier of the bilexical subordinate relation.[2]

As is conventional, we assume two more properties on dependency structures. First, each word has exactly one syntactic head, and second, the structure is acyclic. As a consequence, the edges form a directed tree rooted at node 0.

We say that a dependency structure is *projective* if it has no crossing edges. While in the CoNLL and Stanford conversions of the English Penn Treebank, over 99% of the sentences are projective (Chen and Manning, 2014) — see Fig. 1 for a non-projective English example — for other languages' treebanks, non-projectivity is a common occurrence (see Table 3 for some statistics). This paper is targeted at learning parsers that can handle non-projective dependency trees.

3 MH_4 Deduction System and Its Underlying Transition System

3.1 The MH_4 Deduction System

The MH_4 parser is the instantiation for $k = 4$ of Gómez-Rodríguez et al.'s (2011) more general MH_k parser. MH_k stands for "multi-headed with at most k heads per item": items in its deduction system take the form $[h_1, \ldots, h_p]$ for $p \leqslant k$, indicating the existence of a forest of p dependency subtrees headed by $h_1, \ldots, h_p$ such that their yields are disjoint and the union of their

[2]To simplify exposition here, we only consider the unlabeled case. We use a separately-trained labeling module to obtain labeled parsing results in §5.

$$\textit{Axiom:}\ [0,1] \qquad \textsc{Shift}: \frac{[h_1,\ldots,h_m]}{[h_m, h_m+1]}\ (h_m \leqslant n) \qquad \textsc{Combine}: \frac{[h_1,\ldots,h_m] \qquad [h_m, h_{m+1}, \ldots, h_p]}{[h_1,\ldots,h_p]}\ (p \leqslant k)$$

$$\textit{Goal:}\ [0,n+1] \qquad \textsc{Link}: \frac{[h_1,\ldots,h_m]}{[h_1,\ldots,h_{j-1},h_{j+1},\ldots,h_m]}\ h_i \rightarrow h_j (1 \leqslant i \leqslant m \wedge 1 < j < m \wedge j \neq i)$$

Figure 2: MH_k's deduction steps.

yields is the contiguous substring $h_1 \ldots h_p$ of the input. Deduction steps, shown in Figure 2, can be used to join two such forests that have an endpoint in common via graph union (COMBINE); or to add a dependency arc to a forest that attaches an interior head as a dependent of any of the other heads (LINK).

In the original formulation by Gómez-Rodríguez et al. (2011), all valid items of the form $[i, i+1]$ are considered to be axioms. In contrast, we follow Kuhlmann et al.'s (2011) treatment of MH_3: we consider $[0,1]$ as the only axiom and include an extra SHIFT step to generate the rest of the items of that form. Both formulations are equivalent, but including this SHIFT rule facilitates giving the parser a transition-based interpretation.

Higher values of k provide wider coverage of non-projective structures at an asymptotic runtime complexity of $O(n^k)$. When k is at its minimum value of 3, the parser covers exactly the set of projective trees, and in fact, it can be seen as a transformation[3] of the deduction system described in Gómez-Rodríguez et al. (2008) that gave rise to the projective arc-hybrid parser (Kuhlmann et al., 2011). For $k \geqslant 4$, the parser covers an increasingly larger set of non-projective structures. While a simple characterization of these sets has been lacking[4], empirical evaluation on a large number of treebanks (Gómez-Rodríguez, 2016) has shown MH_k to provide the best known tradeoff between asymptotic complexity and efficiency for $k > 4$. When $k = 4$, its coverage is second only to the 1-Endpoint-Crossing parser of Pitler et al. (2013). Both parsers fully cover well over 80% of the non-projective trees observed in the studied treebanks.

[3]Formally, it is a step refinement; see Gómez-Rodríguez et al. (2011).

[4]This is a common issue with parsers based on the general idea of arcs between non-contiguous heads, such as those deriving from Attardi (2006).

3.2 The MH_4 Transition System

Kuhlmann et al. (2011) show how the items of a variant of MH_3 can be given a transition-based interpretation under the "push computation" framework, yielding the arc-hybrid projective transition system. However, such a derivation has not been made for the non-projective case ($k > 3$), and the known techniques used to derive previous associations between tabular and transition-based parsers do not seem to be applicable in this case. The specific issue is that the deduction systems of Kuhlmann et al. (2011) and Cohen et al. (2011) have in common that the structure of their derivations is similar to that of a Dyck (or balanced-brackets) language, where steps corresponding to shift transitions are balanced with those corresponding to reduce transitions. This makes it possible to group derivation subtrees, and the transition sequences that they yield, into "push computations" that increase the length of the stack by a constant amount. However, this does not seem possible in MH_4.

Instead, we derive a transition-based interpretation of MH_4 by a generalization of that of MH_3 that departs from push computations.

To do so, we start with the MH_3 interpretation of an item $[i, j]$ given by Kuhlmann et al. (2011). This item represents a set of computations (transition sequences) that start from a configuration of the form $(\sigma, i|\beta, A)$ (where σ is the stack and $i|\beta$ is the buffer, with i being the first buffer node) and take the parser to a configuration of the form $(\sigma|i, j|\beta', A)$. That is, the computation has the net effect of placing node i on top of the previous contents of the stack, and it ends in a state where the first buffer element is j.

Under this item semantics, the COMBINE deduction step of the MH_3 parser (i.e., the instantiation of the one in Fig. 2 for $k = 3$) simply concatenates transition sequences. The SHIFT step generates a sequence with a single arc-hybrid sh transition:

$$\mathsf{sh} : (\sigma, h_m|\beta, A) \vdash (\sigma|h_m, \beta, A)$$

and the two possible instantiations of the COMBINE step when $k = 3$ take the antecedent transition sequence and add a transition to it, namely, one of the two arc-hybrid reduce transitions. Written in the context of the node indexes used in Figure 2, these are the following:

$$(\sigma|h_1|h_2, h_3|\beta, A) \vdash (\sigma|h_1, h_3|\beta, A \cup \{h_3 \to h_2\})$$
$$(\sigma|h_1|h_2, h_3|\beta, A) \vdash (\sigma|h_1, h_3|\beta, A \cup \{h_1 \to h_2\})$$

where h_1 and h_3 respectively can be simplified out to obtain the well-known arc-hybrid transitions:

$$\mathsf{la} : (\sigma|h_2, h_3|\beta, A) \vdash (\sigma, h_3|\beta, A \cup \{h_3 \to h_2\})$$
$$\mathsf{ra} : (\sigma|h_1|h_2, \beta, A) \vdash (\sigma|h_1, \beta, A \cup \{h_1 \to h_2\})$$

Now, we assume the following generalization of the item semantics: an item $[h_1, \ldots, h_m]$ represents a set of computations that start from a configuration of the form $(\sigma, h_1|\beta, A)$ and lead to a configuration of the form $(\sigma|h_1|\ldots|h_{m-1}, h_m|\beta', A)$. Note that this generalization no longer follows the "push computation" paradigm of Kuhlmann et al. (2011) and Cohen et al. (2011) because the number of nodes pushed onto the stack depends on the value of m.

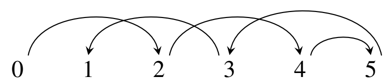
Under this item semantics, the SHIFT and COMBINE steps have the same interpretation as for MH_3. In the case of the LINK step, following the same reasoning as for the MH_3 case, we obtain the following transitions:

$$\mathsf{la} : (\sigma|h_3, h_4|\beta, A) \vdash (\sigma, h_4|\beta, A \cup \{h_4 \to h_3\})$$
$$\mathsf{ra} : (\sigma|h_2|h_3, \beta, A) \vdash (\sigma|h_2, \beta, A \cup \{h_2 \to h_3\})$$
$$\mathsf{la}' : (\sigma|h_2|h_3, h_4|\beta, A) \vdash (\sigma|h_3, h_4|\beta, A \cup \{h_3 \to h_2\})$$
$$\mathsf{ra}' : (\sigma|h_1|h_2|h_3, \beta, A) \vdash (\sigma|h_1|h_3, \beta, A \cup \{h_1 \to h_2\})$$
$$\mathsf{la}_2 : (\sigma|h_2|h_3, h_4|\beta, A) \vdash (\sigma|h_3, h_4|\beta, A \cup \{h_4 \to h_2\})$$
$$\mathsf{ra}_2 : (\sigma|h_1|h_2|h_3, \beta, A) \vdash (\sigma|h_1|h_2, \beta, A \cup \{h_1 \to h_3\})$$

These transitions give us the MH_4 transition system: a parser with four projective reduce transitions (la,ra,la′,ra′) and two Attardi-like, non-adjacent-arc reduce transitions (la_2 and ra_2).

It is worth mentioning that this MH_4 transition system we have obtained is the same as one of the variants of Attardi's algorithm introduced by Shi et al. (2018), there called ALLs_0s_1. However, in that paper they show that it can be tabularized in $O(n^6)$ using the push computation framework. Here, we have derived it as an interpretation of the $O(n^4)$ MH_4 parser.

However, in this case the dynamic programming algorithm does not cover the full search space of the transition system: while each item in the MH_4 parser can be mapped into a computation of this MH_4 transition-based parser, the opposite is not true. This tree:



can be parsed by the transition system using the computation

$$\mathsf{sh}(0); \mathsf{sh}(1); \mathsf{sh}(2); \mathsf{la}_2(3 \to 1); \mathsf{sh}(3); \mathsf{sh}(4);$$
$$\mathsf{la}_2(5 \to 3); \mathsf{sh}(5); \mathsf{ra}(4 \to 5); \mathsf{ra}(2 \to 4); \mathsf{ra}(0 \to 2)$$

but it is not covered by the dynamic programming algorithm, as no deduction sequence will yield an item representing this transition sequence. As we will see, this issue will not prevent us from implementing a dynamic-programming parser with transition-based scoring functions, or from achieving good practical accuracy.

4 Model

Given the transition-based interpretation of the MH_4 system, the learning objective becomes to find a computation that gives the gold-standard parse. For each sentence $w_1, \ldots, w_n$, we train parsers to produce the transition sequence $\mathbf{t}^*$ that corresponds to the annotated dependency structure. Thus, the model consists of two components: a parameterized scorer $S(\mathbf{t})$, and a decoder that finds a sequence $\hat{\mathbf{t}}$ as prediction based on the scoring.

As discussed by Shi et al. (2017a), there exists some tension between rich-feature scoring models and choices of decoders. Ideally, a globally-optimal decoder finds the maximum-scoring transition sequence $\hat{\mathbf{t}}$ without brute-force searching the exponentially-large output space. To keep the runtime of our exact decoder at a practical low-order polynomial, we want its feature set to be

Features	$\{\mathbf{s}_0, \mathbf{b}_0\}$	$\{\mathbf{s}_1, \mathbf{s}_0, \mathbf{b}_0\}$	$\{\mathbf{s}_2, \mathbf{s}_1, \mathbf{s}_0, \mathbf{b}_0\}$
UAS	49.83	85.17	85.27

Table 1: Performance of *local* parsing models with varying number of features. We report average UAS over 10 languages on UD 2.0.

Features	$\{\mathbf{s}_0, \mathbf{b}_0\}$	Hybrid
UAS	86.79	87.27

Table 2: Performance of *global* parsing models with varying number of features.

minimal, consulting as few stack and buffer positions as possible. In what follows, we use s_0 and s_1 to denote the top two stack items and b_0 and b_1 to denote the first two buffer items.

4.1 Scoring and Minimal Features

This section empirically explores the lower limit on the number of necessary positional features. We experiment with both *local* and *global* decoding strategies. The parsers take features extracted from parser configuration c, and score each valid transition t with $S(t; c)$. The *local* parsers greedily take transitions with the highest score until termination, while the *global* parsers use the scores to find the globally-optimal solutions $\hat{\mathbf{t}} = \arg\max_{\mathbf{t}} S(\mathbf{t})$, where $S(\mathbf{t})$ is the sum of scores for the component transitions.

Following prior work, we employ bi-LSTMs for compact feature representation. A bi-LSTM runs in both directions on the input sentence, and assigns a context-sensitive vector encoding to each token in the sentence: $\mathbf{w}_1, \ldots, \mathbf{w}_n$. When we need to extract features, say, $\mathbf{s}_0$, from a particular stack or buffer position, say s_0, we directly use the bi-LSTM vector $\mathbf{w}_{i_{s_0}}$, where i_{s_0} gives the index of the subroot of s_0 into the sentence.

Shi et al. (2017a) showed that feature vectors $\{\mathbf{s}_0, \mathbf{b}_0\}$ suffice for MH_3. Table 1 and Table 2 show the use of small feature sets for MH_4, for *local* and *global* parsing models, respectively. For a *local* parser to exhibit decent performance, we need at least $\{\mathbf{s}_1, \mathbf{s}_0, \mathbf{b}_0\}$, but adding $\mathbf{s}_2$ on top of that does not show any significant impact on the performance. Interestingly, in the case of *global* models, the two-vector feature set $\{\mathbf{s}_0, \mathbf{b}_0\}$ already suffices. Adding $\mathbf{s}_1$ to the global setting (column "Hybrid" in Table 2) seems attractive, but entails resolving a technical challenge that we discuss in the following section.

4.2 Global Decoder

In our transition-system interpretation of MH_k, sh transitions correspond to SHIFT and reduce transitions reflect the LINK steps. Since the SHIFT conclusions lose the contexts needed to score the transitions, we set the scores for all SHIFT rules to zero and delegate the scoring of the sh transitions to the COMBINE steps, as as in Shi et al. (2017a); for example,

$$\frac{[h_1, h_2] : v_1 \quad [h_2, h_3, h_4] : v_2}{[h_1, h_2, h_3, h_4] : v_1 + v_2 + S(\mathsf{sh}; \{\mathbf{h}_1, \mathbf{h}_2\})}$$

Here the transition sequence denoted by $[h_2, h_3, h_4]$ starts from a sh, with h_1 and h_2 taking the s_0 and b_0 positions. If we further wish to access s_1, such information is not readily available in the deduction step, apparently requiring extra bookkeeping that pushes the space and time complexity to an impractical $O(n^4)$ and $O(n^5)$, respectively. But, consider the scoring for the reduce transitions in the LINK steps:

$$\frac{[h_1, h_2, h_3, h_4] : v}{[h_1, h_2, h_4] : v + S(\mathsf{la}; \{\mathbf{h}_2, \mathbf{h}_3, \mathbf{h}_4\})}$$

$$\frac{[h_1, h_2, h_3] : v}{[h_1, h_3] : v + S(\mathsf{la}; \{\mathbf{h}_1, \mathbf{h}_2, \mathbf{h}_3\})}$$

The deduction steps *already* keep indices for s_1 (h_2 in the first rule, h_1 in the second) and thus provide direct access without any modification. To resolve the conflict between including s_1 for richer representations and the unavailability of s_1 in scoring the sh transitions in the COMBINE steps, we propose a hybrid scoring approach — we use features $\{\mathbf{s}_0, \mathbf{b}_0\}$ when scoring a sh transition, and features $\{\mathbf{s}_1, \mathbf{s}_0, \mathbf{b}_0\}$ for consideration of reduce transitions. We call this method MH_4-hybrid, in contrast to MH_4-two, where we simply take $\{\mathbf{s}_0, \mathbf{b}_0\}$ for scoring all transitions.

4.3 Large-Margin Training

We train the greedy parsers with hinge loss, and the global parsers with its structured version (Taskar et al., 2005). The loss function for each sentence is formally defined as:

$$\max_{\hat{\mathbf{t}}} \left(S(\hat{\mathbf{t}}) + cost(\mathbf{t}^*, \hat{\mathbf{t}}) - S(\mathbf{t}^*) \right)$$

where the margin $cost(\mathbf{t}^*, \hat{\mathbf{t}})$ counts the number of mis-attached nodes for taking sequence $\hat{\mathbf{t}}$ instead of $\mathbf{t}^*$. Minimizing this loss can be thought of as optimizing for the attachment scores.

The calculation of the above loss function can be solved as efficiently as the deduction system if the $cost$ function decomposes into the dynamic program. We achieve this by replacing the scoring of each reduce step by its cost-augmented version:

$$\frac{[h_1, h_2, h_3, h_4] : v}{[h_1, h_2, h_4] : v + S(\mathsf{la}_2; \{\mathbf{h}_2, \mathbf{h}_3, \mathbf{h}_4\}) + \Delta}$$

where $\Delta = \mathbf{1}(head(w_{h_3}) \neq w_{h_4})$. This loss function encourages the model to give higher contrast between gold-standard and wrong predictions, yielding better generalization results.

5 Experiments

Data and Evaluation We experiment with the Universal Dependencies (UD) 2.0 dataset used for the CoNLL 2017 shared task (Zeman et al., 2017). We restrict our choice of languages to be those with only one training treebank, for a better comparison with the shared task results.[5] Among these languages, we pick the top 10 most non-projective languages. Their basic statistics are listed in Table 3. For all development-set results, we assume gold-standard tokenization and sentence delimitation. When comparing to the shared task results on test sets, we use the provided baseline UDPipe (Straka et al., 2016) segmentation. Our models do not use part-of-speech tags or morphological tags as features, but rather leverage such information via stack propagation (Zhang and Weiss, 2016), i.e., we learn to predict them as a secondary training objective. We report unlabeled attachment F1-scores (UAS) on the development sets for better focus on comparing our (unlabeled) parsing modules. We report its labeled variant (LAS), the main metric of the shared task, on the test sets. For each experiment setting, we ran the model with 5 different random initializations, and report the mean and standard deviation. We detail the implementation details in the supplementary material.

Baseline Systems For comparison, we include three baseline systems with the same underlying feature representations and scoring paradigm. All the following baseline systems are trained with the cost-augmented large-margin loss function.

The MH_3 *parser* is the projective instantiation of the MH_k parser family. This corresponds to the global version of the arc-hybrid transition system (Kuhlmann et al., 2011). We adopt the minimal feature representation $\{\mathbf{s}_0, \mathbf{b}_0\}$, following Shi et al. (2017a). For this model, we also implement a greedy incremental version.

The *edge-factored non-projective maximal spanning tree (MST) parser* allows arbitrary non-projective structures. This decoding approach has been shown to be very competitive in parsing non-projective treebanks (McDonald et al., 2005), and was deployed in the top-performing system at the CoNLL 2017 shared task (Dozat et al., 2017). We score each edge individually, with the features being the bi-LSTM vectors $\{\mathbf{h}, \mathbf{m}\}$, where h is the head, and m the modifier of the edge.

The *crossing-sensitive third-order 1EC parser* provides a hybrid dynamic program for parsing 1-Endpoint-Crossing non-projective dependency trees with higher-order factorization (Pitler, 2014). Depending on whether an edge is crossed, we can access the modifier's grandparent g, head h, and sibling si. We take their corresponding bi-LSTM features $\{\mathbf{g}, \mathbf{h}, \mathbf{m}, \mathbf{si}\}$ for scoring each edge. This is a re-implementation of Pitler (2014) with neural scoring functions.

Main Results Table 4 shows the development-set performance of our models as compared with baseline systems. MST considers non-projective structures, and thus enjoys a theoretical advantage over projective MH_3, especially for the most non-projective languages. However, it has a vastly larger output space, making the selection of correct structures difficult. Further, the scoring is edge-factored, and does not take any structural contexts into consideration. This tradeoff leads to the similar performance of MST comparing to MH_3. In comparison, both 1EC and MH_4 are mildly non-projective parsing algorithms, limiting the size of the output space. 1EC includes higher-order features that look at tree-structural contexts; MH_4 derives its features from parsing configurations of a transition system, hence leveraging contexts within transition sequences. These considerations explain their significant improvements over MST. We also observe that MH_4 recovers more short dependencies than 1EC, while 1EC is better at longer-distance ones.

[5] When multiple treebanks are available, one can develop domain transfer strategies, which is not the focus of this work.

Language	Code	# Sent.	# Words	Sentence Coverage (%) Proj. ↓	MH_4	1EC	Edge Coverage (%) Proj.	MH_4	1EC
Basque	eu	5,396	72,974	66.48	91.48	93.29	95.98	99.27	99.42
Urdu	ur	4,043	108,690	76.97	95.89	95.77	98.89	99.83	99.81
Gothic	got	3,387	35,024	78.42	97.25	97.58	97.04	99.73	99.75
Hungarian	hu	910	20,166	79.01	98.35	97.69	98.51	99.92	99.89
Old Church Slavonic	cu	4,123	37,432	80.16	98.33	98.74	97.22	99.80	99.85
Danish	da	4,383	80,378	80.56	97.70	98.97	98.60	99.87	99.94
Greek	el	1,662	41,212	85.98	99.52	99.40	99.32	99.98	99.98
Hindi	hi	13,304	281,057	86.16	98.38	98.95	99.26	99.92	99.94
German	de	14,118	269,626	87.07	99.19	99.27	99.15	99.95	99.96
Romanian	ro	8,043	185,113	88.61	99.42	99.52	99.42	99.97	99.98

Table 3: Statistics of selected training treebanks from Universal Dependencies 2.0 for the CoNLL 2017 shared task (Zeman et al., 2017), sorted by per-sentence projective ratio.

Lan.	Global Models MH_3	MST	MH_4-two	MH_4-hybrid	1EC	Greedy Models MH_3	MH_4
eu	$82.07_{\pm 0.17}$	$83.61_{\pm 0.16}$	$82.94_{\pm 0.24}$	$\mathbf{84.13}_{\pm 0.13}$	$84.09_{\pm 0.19}$	$81.27_{\pm 0.20}$	$81.71_{\pm 0.33}$
ur	$86.89_{\pm 0.18}$	$86.78_{\pm 0.13}$	$86.84_{\pm 0.26}$	$87.06_{\pm 0.24}$	$\mathbf{87.11}_{\pm 0.11}$	$86.40_{\pm 0.16}$	$86.05_{\pm 0.18}$
got	$83.72_{\pm 0.19}$	$84.74_{\pm 0.28}$	$83.85_{\pm 0.19}$	$84.59_{\pm 0.38}$	$\mathbf{84.77}_{\pm 0.27}$	$82.28_{\pm 0.18}$	$81.40_{\pm 0.45}$
hu	$83.05_{\pm 0.17}$	$82.81_{\pm 0.49}$	$83.69_{\pm 0.20}$	$\mathbf{84.59}_{\pm 0.50}$	$83.48_{\pm 0.27}$	$81.75_{\pm 0.47}$	$80.75_{\pm 0.54}$
cu	$86.70_{\pm 0.30}$	$88.02_{\pm 0.25}$	$87.57_{\pm 0.14}$	$88.09_{\pm 0.28}$	$\mathbf{88.27}_{\pm 0.32}$	$86.05_{\pm 0.23}$	$86.01_{\pm 0.11}$
da	$85.09_{\pm 0.16}$	$84.68_{\pm 0.36}$	$85.45_{\pm 0.43}$	$\mathbf{85.77}_{\pm 0.39}$	$\mathbf{85.77}_{\pm 0.16}$	$83.90_{\pm 0.24}$	$83.59_{\pm 0.06}$
el	$87.82_{\pm 0.24}$	$87.27_{\pm 0.22}$	$87.77_{\pm 0.20}$	$87.83_{\pm 0.36}$	$\mathbf{87.95}_{\pm 0.23}$	$87.14_{\pm 0.25}$	$86.95_{\pm 0.25}$
hi	$93.75_{\pm 0.14}$	$93.91_{\pm 0.26}$	$93.99_{\pm 0.15}$	$\mathbf{94.27}_{\pm 0.08}$	$94.24_{\pm 0.04}$	$93.44_{\pm 0.09}$	$93.02_{\pm 0.10}$
de	$86.46_{\pm 0.13}$	$86.34_{\pm 0.24}$	$86.53_{\pm 0.22}$	$86.89_{\pm 0.17}$	$\mathbf{86.95}_{\pm 0.32}$	$84.99_{\pm 0.26}$	$85.27_{\pm 0.32}$
ro	$89.34_{\pm 0.27}$	$88.79_{\pm 0.43}$	$89.25_{\pm 0.15}$	$\mathbf{89.53}_{\pm 0.20}$	$89.52_{\pm 0.25}$	$88.76_{\pm 0.30}$	$87.97_{\pm 0.31}$
Avg.	86.49	86.69	86.79	**87.27**	87.21	85.60	85.27

Table 4: Experiment results (UAS, %) on the UD 2.0 development set. Bold: best result per language.

In comparison to MH_4-two, the richer feature representation of MH_4-hybrid helps in all our languages.

Interestingly, MH_4 and MH_3 react differently to switching from global to greedy models. MH_4 covers more structures than MH_3, and is naturally more capable in the global case, even when the feature functions are the same (MH_4-two). However, its greedy version is outperformed by MH_3. We conjecture that this is because MH_4 explores only the same number of configurations as MH_3, despite the fact that introducing non-projectivity expands the search space dramatically.

Comparison with CoNLL Shared Task Results (Table 5) We compare our models on the test sets, along with the best single model (#1; Dozat et al., 2017) and the best ensemble model (#2; Shi et al., 2017b) from the CoNLL 2017 shared task. MH_4 outperforms 1EC in 7 out of the 10 languages. Additionally, we take our non-projective parsing models (MST, MH_4-hybrid, 1EC) and combine them into an ensemble. The average result is competitive with the best CoNLL submissions. Interestingly, Dozat et al. (2017) uses fully non-projective parsing algorithms (MST), and our ensemble system sees larger gains in the more non-projective languages, confirming the potential benefit of global mildly non-projective parsing.

Results on Projective Languages (Table 6) For completeness, we also test our models on the 10 most projective languages that have a single treebank. MH_4 remains the most effective, but by a much smaller margin. Interestingly, MH_3, which is strictly projective, matches the performance of 1EC; both outperform the fully non-projective MST by half a point.

6 Related Work

Exact inference for dependency parsing can be achieved in cubic time if the model is restricted to projective trees (Eisner, 1996). However, non-projectivity is needed for natural language parsers to satisfactorily deal with linguistic phenomena like topicalization, scrambling and extraposition, which cause crossing dependencies. In UD 2.0, 68 out of 70 treebanks were reported to contain

Lan.	MH_3	MST	MH_4-hybrid		1EC	Ensemble	CoNLL #1	CoNLL #2
	Same Model Architecture					For Reference		
eu	$78.17_{\pm 0.33}$	$79.90_{\pm 0.08}$	$\mathbf{80.22}_{\pm 0.48}$	>	$80.17_{\pm 0.32}$	**81.55**	81.44	79.61
ur	$\mathbf{80.91}_{\pm 0.10}$	$80.05_{\pm 0.13}$	$80.69_{\pm 0.19}$	>	$80.59_{\pm 0.19}$	81.37	**82.28**	81.06
got	$67.10_{\pm 0.10}$	$67.26_{\pm 0.45}$	$\mathbf{67.92}_{\pm 0.29}$	>	$67.66_{\pm 0.20}$	**69.83**	66.82	68.34
hu	$76.09_{\pm 0.25}$	$75.79_{\pm 0.36}$	$\mathbf{76.90}_{\pm 0.31}$	>	$76.07_{\pm 0.20}$	**79.35**	77.56	76.55
cu	$71.28_{\pm 0.29}$	$72.18_{\pm 0.20}$	$72.51_{\pm 0.23}$	<	$\mathbf{72.53}_{\pm 0.27}$	**74.38**	71.84	72.35
da	$80.00_{\pm 0.15}$	$79.69_{\pm 0.24}$	$\mathbf{80.89}_{\pm 0.17}$	>	$80.83_{\pm 0.27}$	82.09	**82.97**	81.55
el	$85.89_{\pm 0.29}$	$85.48_{\pm 0.25}$	$\mathbf{86.28}_{\pm 0.44}$	>	$86.07_{\pm 0.37}$	87.06	**87.38**	86.90
hi	$89.88_{\pm 0.18}$	$89.93_{\pm 0.12}$	$90.22_{\pm 0.12}$	<	$\mathbf{90.28}_{\pm 0.21}$	90.78	**91.59**	90.40
de	$76.23_{\pm 0.21}$	$75.99_{\pm 0.23}$	$\mathbf{76.46}_{\pm 0.20}$	>	$76.42_{\pm 0.35}$	77.38	**80.71**	77.17
ro	$83.53_{\pm 0.35}$	$82.73_{\pm 0.36}$	$83.67_{\pm 0.21}$	<	$\mathbf{83.83}_{\pm 0.18}$	84.51	**85.92**	84.40
Avg.	78.91	78.90	**79.57**	>	79.44	80.83	**80.85**	79.83

Table 5: Evaluation results (LAS, %) on the test set using the CoNLL 2017 shared task setup. The best results for each language within each block are highlighted in bold.

Lan.	MH_3	MST	MH_4-hybrid	1EC	Ensemble	CoNLL #1	CoNLL #2
	Same Model Architecture				For Reference		
ja	$\mathbf{74.29}_{\pm 0.10}$	$73.93_{\pm 0.16}$	$74.23_{\pm 0.11}$	$74.12_{\pm 0.12}$	74.51	**74.72**	74.51
zh	$\mathbf{63.54}_{\pm 0.13}$	$62.71_{\pm 0.17}$	$63.48_{\pm 0.33}$	$\mathbf{63.54}_{\pm 0.26}$	64.65	**65.88**	64.14
pl	$86.49_{\pm 0.19}$	$85.76_{\pm 0.31}$	$\mathbf{86.60}_{\pm 0.26}$	$86.36_{\pm 0.28}$	87.38	**90.32**	87.15
he	$61.47_{\pm 0.24}$	$61.28_{\pm 0.24}$	$\mathbf{61.93}_{\pm 0.22}$	$61.75_{\pm 0.22}$	62.40	**63.94**	62.33
vi	$41.26_{\pm 0.39}$	$41.04_{\pm 0.19}$	$\mathbf{41.33}_{\pm 0.32}$	$40.96_{\pm 0.36}$	**42.95**	42.13	41.68
bg	$87.50_{\pm 0.20}$	$87.03_{\pm 0.17}$	$\mathbf{87.63}_{\pm 0.17}$	$87.56_{\pm 0.14}$	88.22	**89.81**	88.39
sk	$80.48_{\pm 0.22}$	$80.25_{\pm 0.32}$	$\mathbf{81.27}_{\pm 0.14}$	$80.94_{\pm 0.25}$	82.38	**86.04**	81.75
it	$87.90_{\pm 0.07}$	$87.26_{\pm 0.23}$	$\mathbf{88.06}_{\pm 0.27}$	$87.98_{\pm 0.19}$	88.74	**90.68**	89.08
id	$\mathbf{77.66}_{\pm 0.13}$	$76.95_{\pm 0.32}$	$77.64_{\pm 0.17}$	$77.60_{\pm 0.18}$	78.27	**79.19**	78.55
lv	$69.62_{\pm 0.55}$	$69.33_{\pm 0.51}$	$\mathbf{70.54}_{\pm 0.51}$	$69.52_{\pm 0.29}$	72.34	**74.01**	71.35
Avg.	73.02	72.55	**73.27**	73.03	74.18	**75.67**	73.89

Table 6: CoNLL 2017 test set results (LAS, %) on the most projective languages (sorted by projective ratio; ja (Japanese) is fully projective).

non-projectivity (Wang et al., 2017).

However, exact inference has been shown to be intractable for models that support arbitrary non-projectivity, except under strong independence assumptions (McDonald and Satta, 2007). Thus, exact inference parsers that support unrestricted non-projectivity are limited to edge-factored models (McDonald et al., 2005; Dozat et al., 2017). Alternatives include treebank transformation and pseudo-projective parsing (Kahane et al., 1998; Nivre and Nilsson, 2005), approximate inference (e.g. McDonald and Pereira (2006); Attardi (2006); Nivre (2009); Fernández-González and Gómez-Rodríguez (2017)) or focusing on sets of dependency trees that allow only restricted forms of non-projectivity. A number of such sets, called mildly non-projective classes of trees, have been identified that both exhibit good empirical coverage of the non-projective phenomena found in natural languages and are known to have polynomial-time exact parsing algorithms; see Gómez-Rodríguez (2016) for a survey.

However, most of these algorithms have not been implemented in practice due to their prohibitive complexity. For example, Corro et al. (2016) report an implementation of the WG_1 parser, a $O(n^7)$ mildly non-projective parser introduced in Gómez-Rodríguez et al. (2009), but it could not be run for real sentences of length greater than 20. On the other hand, Pitler et al. (2012) provide an implementation of an $O(n^5)$ parser for a mildly non-projective class of structures called gap-minding trees, but they need to resort to aggressive pruning to make it practical, exploring only a part of the search space in $O(n^4)$ time.

To the best of our knowledge, the only practical system that actually implements exact inference for mildly non-projective parsing is the 1-Endpoint-Crossing (1EC) parser of Pitler (2013; 2014), which runs in $O(n^4)$ worst-case time like the MH_4 algorithm used in this paper. Thus, the system presented here is the second practical implementation of exact mildly non-projective parsing that has successfully been executed on real corpora.[6]

Comparing with Pitler (2014)'s 1EC, our parser has the following disadvantages: (-1) It has slightly lower coverage, at least on the treebanks considered by Gómez-Rodríguez (2016). (-2) The set of trees covered by MH_4 has not been characterized with a non-operational definition, while the set of 1-Endpoint-Crossing trees can be simply defined.

However, it also has the following advantages: (+1) It can be given a transition-based interpretation, allowing us to use transition-based scoring functions and to implement the analogous algorithm with greedy or beam search apart from exact inference. No transition-based interpretation is known for 1EC. While a transition-based algorithm has been defined for a strict subset of 1-Endpoint-Crossing trees, called 2-Crossing Interval trees (Pitler and McDonald, 2015), this is a separate algorithm with no known mapping or relation to 1EC or any other dynamic programming model. Thus, we provide the first exact inference algorithm for a non-projective transition-based parser with practical complexity. (+2) It is conceptually much simpler, with one kind of item and two deduction steps, while the 1-Endpoint-Crossing parser has five classes of items and several dozen distinct deduction steps. It is also a purely bottom-up parser, whereas the 1-Endpoint-Crossing parser does not have the bottom-up property. This property is necessary for models that involve compositional representations of subtrees (Dyer et al., 2015), and facilitates parallelization and partial parsing. (+3) It can be easily generalized to MH_k for $k > 4$, providing higher coverage, with time complexity $O(n^k)$. Out of the mildly non-projective parsers studied in Gómez-Rodríguez (2016), MH_4 provides the maximum coverage with respect to its complexity for $k > 4$. (+4) As shown in §5, MH_4 obtains slightly higher accuracy than 1EC on average, albeit not by a conclusive margin.

It is worth noting that 1EC has recently been extended to graph parsing by Kurtz and Kuhlmann (2017), Kummerfeld and Klein (2017), and Cao et al. (2017a,b), with the latter providing a practical implementation of a parser for 1-Endpoint-Crossing, pagenumber-2 graphs.

[6] Corro et al. (2016) describe a parser that enforces mildly non-projective constraints (bounded block degree and well-nestedness), but it is an arc-factored model, so it is subject to the same strong independence assumptions as maximum-spanning-tree parsers like McDonald et al. (2005) and does not support the greater flexibility in scoring that is the main advantage of mildly non-projective parsers over these. Instead, mild non-projectivity is exclusively used as a criterion to discard nonconforming trees.

7 Conclusion

We have extended the parsing architecture of Shi et al. (2017a) to non-projective dependency parsing by implementing the MH_4 parser, a mildly non-projective $O(n^4)$ chart parsing algorithm, using a minimal set of transition-based bi-LSTM features. For this purpose, we have established a mapping between MH_4 items and transition sequences of an underlying non-projective transition-based parser.

To our knowledge, this is the first practical implementation of exact inference for non-projective transition-based parsing. Empirical results on a collection of highly non-projective datasets from Universal Dependencies show improvements in accuracy over the projective approach of Shi et al. (2017a), as well as edge-factored maximum-spanning-tree parsing. The results are on par with the 1-Endpoint-Crossing parser of Pitler (2014) (re-implemented under the same neural framework), but our algorithm is notably simpler and has additional desirable properties: it is purely bottom-up, generalizable to higher coverage, and compatible with transition-based semantics.

Acknowledgments

We thank the three anonymous reviewers for their helpful comments. CG has received funding from the European Research Council (ERC), under the European Union's Horizon 2020 research and innovation programme (FASTPARSE, grant agreement No 714150), from the TELEPARES-UDC project (FFI2014-51978-C2-2-R) and the ANSWER-ASAP project (TIN2017-85160-C2-1-R) from MINECO, and from Xunta de Galicia (ED431B 2017/01). TS and LL were supported in part by a Google Focused Research Grant to Cornell University. LL was also supported in part by NSF grant SES-1741441. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation or other sponsors.

References

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally normalized transition-based neural networks. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2442–2452, Berlin, Germany. Association for Computational Linguistics.

Giuseppe Attardi. 2006. Experiments with a multilanguage non-projective dependency parser. In *Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL-X)*, pages 166–170, New York City, New York, USA.

Junjie Cao, Sheng Huang, Weiwei Sun, and Xiaojun Wan. 2017a. Parsing to 1-endpoint-crossing, pagenumber-2 graphs. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2110–2120, Vancouver, Canada. Association for Computational Linguistics.

Junjie Cao, Sheng Huang, Weiwei Sun, and Xiaojun Wan. 2017b. Quasi-second-order parsing for 1-endpoint-crossing, pagenumber-2 graphs. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 24–34, Copenhagen, Denmark. Association for Computational Linguistics.

Danqi Chen and Christopher D. Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 740–750, Doha, Qatar.

Shay B. Cohen, Carlos Gómez-Rodríguez, and Giorgio Satta. 2011. Exact inference for generative probabilistic non-projective dependency parsing. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1234–1245, Edinburgh, Scotland, UK.

Caio Corro, Joseph Le Roux, Mathieu Lacroix, Antoine Rozenknop, and Roberto Wolfler Calvo. 2016. Dependency parsing with bounded block degree and well-nestedness via Lagrangian relaxation and branch-and-bound. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 355–366, Berlin, Germany. Association for Computational Linguistics.

James Cross and Liang Huang. 2016. Incremental parsing with minimal features using bi-directional LSTM. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 32–37.

Timothy Dozat, Peng Qi, and Christopher D. Manning. 2017. Stanford's graph-based neural dependency parser at the CoNLL 2017 shared task. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 20–30, Vancouver, Canada. Association for Computational Linguistics.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 334–343.

Jason Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING-96)*, pages 340–345, Copenhagen.

Daniel Fernández-González and Carlos Gómez-Rodríguez. 2017. A full non-monotonic transition system for unrestricted non-projective parsing. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 288–298, Vancouver, Canada. Association for Computational Linguistics.

Carlos Gómez-Rodríguez. 2016. Restricted non-projectivity: Coverage vs. efficiency. *Computational Linguistics*, 42(4):809–817.

Carlos Gómez-Rodríguez, John Carroll, and David Weir. 2008. A deductive approach to dependency parsing. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technology*, pages 968–976.

Carlos Gómez-Rodríguez, John Carroll, and David Weir. 2011. Dependency parsing schemata and mildly non-projective dependency parsing. *Computational Linguistics*, 37(3):541–586.

Carlos Gómez-Rodríguez, David Weir, and John Carroll. 2009. Parsing mildly non-projective dependency structures. In *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*, pages 291–299, Athens, Greece. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Liang Huang and Kenji Sagae. 2010. Dynamic programming for linear-time incremental parsing. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1077–1086, Uppsala, Sweden.

Sylvain Kahane, Alexis Nasr, and Owen Rambow. 1998. Pseudo-projectivity: A polynomially parsable non-projective dependency grammar. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics - Volume 1*, pages 646–652, Montreal, Quebec, Canada. Association for Computational Linguistics.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional LSTM feature representations. *Transactions of the Association for Computational Linguistics*, 4:313–327.

Sandra Kübler, Ryan McDonald, and Joakim Nivre. 2009. *Dependency parsing*, volume 2 of *Synthesis Lectures on Human Language Technologies*. Morgan & Claypool Publishers.

Marco Kuhlmann, Carlos Gómez-Rodríguez, and Giorgio Satta. 2011. Dynamic programming algorithms for transition-based dependency parsers. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL-HLT)*, pages 673–682, Portland, Oregon, USA.

Jonathan K. Kummerfeld and Dan Klein. 2017. Parsing with traces: An $O(n^4)$ algorithm and a structural representation. *Transactions of the Association for Computational Linguistics*, 5:441–454.

Robin Kurtz and Marco Kuhlmann. 2017. Exploiting structure in parsing to 1-endpoint-crossing graphs. In *Proceedings of the 15th International Conference on Parsing Technologies*, pages 78–87, Pisa, Italy. Association for Computational Linguistics.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajic. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 523–530, Vancouver, British Columbia, Canada. Association for Computational Linguistics.

Ryan McDonald and Giorgio Satta. 2007. On the complexity of non-projective data-driven dependency parsing. In *Proceedings of the Tenth International Conference on Parsing Technologies (IWPT)*, pages 121–132, Prague, Czech Republic. Association for Computational Linguistics.

Ryan T. McDonald and Fernando C. N. Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics*, pages 81–88, Trento, Italy. Association for Computational Linguistics.

Joakim Nivre. 2009. Non-projective dependency parsing in expected linear time. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 351–359, Suntec, Singapore. Association for Computational Linguistics.

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, Michigan. Association for Computational Linguistics.

Joakim Nivre and Mario Scholz. 2004. Deterministic dependency parsing of English text. In *Proceedings of the 20th International Conference on Computational Linguistics*, pages 64–70, Geneva, Switzerland. COLING.

Emily Pitler. 2014. A crossing-sensitive third-order factorization for dependency parsing. *Transactions of the Association for Computational Linguistics*, 2:41–54.

Emily Pitler, Sampath Kannan, and Mitchell Marcus. 2012. Dynamic programming for higher order parsing of gap-minding trees. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 478–488. Association for Computational Linguistics.

Emily Pitler, Sampath Kannan, and Mitchell Marcus. 2013. Finding optimal 1-endpoint-crossing trees. *Transactions of the Association of Computational Linguistics*, 1:13–24.

Emily Pitler and Ryan McDonald. 2015. A linear-time transition system for crossing interval trees. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 662–671, Denver, Colorado. Association for Computational Linguistics.

Tianze Shi, Carlos Gómez-Rodríguez, and Lillian Lee. 2018. Improving coverage and runtime complexity for exact inference in non-projective transition-based dependency parsers. In *Proceedings of the 16th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, page (in press), New Orleans, Louisiana. Association for Computational Linguistics.

Tianze Shi, Liang Huang, and Lillian Lee. 2017a. Fast(er) exact decoding and global training for transition-based dependency parsing via a minimal feature set. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 12–23, Copenhagen, Denmark.

Tianze Shi, Felix G. Wu, Xilun Chen, and Yao Cheng. 2017b. Combining global models for parsing Universal Dependencies. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 31–39, Vancouver, Canada. Association for Computational Linguistics.

Milan Straka, Jan Hajic, and Jana Straková. 2016. UDPipe: Trainable pipeline for processing CoNLL-U files performing tokenization, morphological analysis, POS tagging and parsing. In *Proceedings* *of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France. European Language Resources Association (ELRA).

Ben Taskar, Vassil Chatalbashev, Daphne Koller, and Carlos Guestrin. 2005. Learning structured prediction models: A large margin approach. In *Proceedings of the 22nd International Conference on Machine Learning*, pages 896–903.

Hao Wang, Hai Zhao, and Zhisong Zhang. 2017. A transition-based system for universal dependency parsing. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 191–197, Vancouver, Canada. Association for Computational Linguistics.

Hiroyasu Yamada and Yuji Matsumoto. 2003. Statistical dependency analysis with Support Vector Machines. In *Proceedings of the 8th International Workshop on Parsing Technologies*, pages 195–206.

Daniel Zeman, Martin Popel, Milan Straka, Jan Hajic, Joakim Nivre, Filip Ginter, Juhani Luotolahti, Sampo Pyysalo, Slav Petrov, Martin Potthast, Francis Tyers, Elena Badmaeva, Memduh Gokirmak, Anna Nedoluzhko, Silvie Cinkova, Jan Hajic jr., Jaroslava Hlavacova, Václava Kettnerová, Zdenka Uresova, Jenna Kanerva, Stina Ojala, Anna Missilä, Christopher D. Manning, Sebastian Schuster, Siva Reddy, Dima Taji, Nizar Habash, Herman Leung, Marie-Catherine de Marneffe, Manuela Sanguinetti, Maria Simi, Hiroshi Kanayama, Valeria dePaiva, Kira Droganova, Héctor Martínez Alonso, Çağrı Çöltekin, Umut Sulubacak, Hans Uszkoreit, Vivien Macketanz, Aljoscha Burchardt, Kim Harris, Katrin Marheinecke, Georg Rehm, Tolga Kayadelen, Mohammed Attia, Ali Elkahky, Zhuoran Yu, Emily Pitler, Saran Lertpradit, Michael Mandl, Jesse Kirchner, Hector Fernandez Alcalde, Jana Strnadová, Esha Banerjee, Ruli Manurung, Antonio Stella, Atsuko Shimada, Sookyoung Kwak, Gustavo Mendonca, Tatiana Lando, Rattima Nitisaroj, and Josie Li. 2017. CoNLL 2017 shared task: Multilingual parsing from raw text to Universal Dependencies. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 1–19, Vancouver, Canada. Association for Computational Linguistics.

Yuan Zhang and David Weiss. 2016. Stack-propagation: Improved representation learning for syntax. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1557–1566, Berlin, Germany. Association for Computational Linguistics.

Yue Zhang and Joakim Nivre. 2011. Transition-based dependency parsing with rich non-local features. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 188–193, Portland, Oregon, USA. Association for Computational Linguistics.