Improving Coverage and Runtime Complexity for Exact Inference in Non-Projective Transition-Based Dependency Parsers

Tianze Shi
Cornell University
tianze@cs.cornell.edu

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

Lillian Lee
Cornell University
llee@cs.cornell.edu

Abstract

We generalize Cohen, Gómez-Rodríguez, and Satta's (2011) parser to a family of non-projective transition-based dependency parsers allowing polynomial-time exact inference. This includes novel parsers with better coverage than Cohen et al. (2011), and even a variant that reduces time complexity to $O(n^6)$, improving on prior bounds. We hope that this piece of theoretical work inspires design of novel transition systems with better coverage and better run-time guarantees.

1 Introduction

Non-projective dependency trees are those containing crossing edges. They account for 12.59% of all training sentences in the annotated Universal Dependencies (UD) 2.1 data (Nivre et al., 2017), and more than 20% in each of 10 languages among the 54 in UD 2.1 with training treebanks. But modeling non-projectivity is computationally costly (McDonald and Satta, 2007).

Some transition-based dependency parsers have deduction systems that use dynamic programming to enable *exact* inference in polynomial time and space (Huang and Sagae, 2010; Kuhlmann et al., 2011). For non-projective parsing, though, the only tabularization of a transition-based parser is, to our knowledge, that of Cohen et al. (2011). They define a deduction system for (an isomorphic variant of) Attardi's (2006) transition system, which covers a subset of non-projective trees. The exact inference algorithm runs in $O(n^7)$ time, where n denotes sentence length.

In this paper, we show how Cohen et al.'s (2011) system can be modified to generate a new family of deduction systems with corresponding transition systems. In particular, we present three novel variants of the degree-2 Attardi parser, summarized in Fig. 1 (our technique can also be applied to generalized Attardi (2006) systems; see §3.2). The first two bring non-projective coverage for UD 2.1 to as high as 95.99% by adding extra transitions, and yet retain the same time complexity. The third reduces time complexity for exact inference to $O(n^6)$ and space complexity from $O(n^5)$ to $O(n^4)$, while still improving empirical coverage from 87.24% to 93.16%.[1] Code and full statistics for all treebanks can be found at https://github.com/tzshi/nonproj-dp-variants-naacl2018.

These theoretical improvements are a step towards making recent state-of-the-art results in transition-based parsing with exact inference (Shi et al., 2017) extensible to practical non-projective parsing, by exemplifying the design of transition systems with better coverage on highly non-projective datasets and, for one variant, bringing the runtime complexity one level closer to feasibility.

2 Transition-based Parsing

We first introduce necessary definitions and notation.

2.1 A General Class of Transition Systems

A *transition system* is given by a 4-tuple (C, T, c^s, C_τ), where C is a set of configurations, T is a set of transition functions between configurations, c^s is an initialization function mapping an input sentence to an initial configuration, and $C_\tau \subset C$ defines a set of terminal configurations.

[1]Faster exact inference algorithms have been defined for some sets of mildly non-projective trees (e.g. Pitler et al. (2013); see Gómez-Rodríguez (2016) for more), but lack an underlying transition system. Having one has the practical advantage of allowing generative models, as in Cohen et al. (2011), and transition-based scoring functions, which have yielded good projective-parsing results (Shi et al., 2017); plus the theoretical advantage of providing a single framework supporting greedy, beam-search, and exact inference.

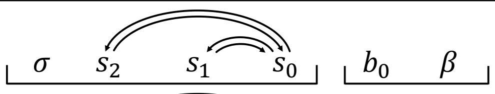
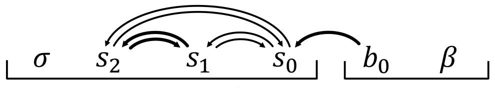
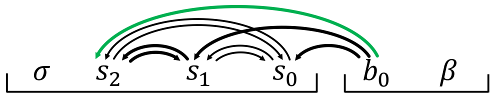
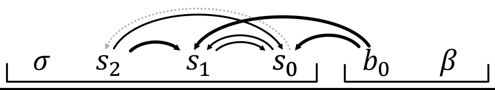
System	Reduce Transitions	Non-proj. Coverage	Time Complexity	Max. Degree
Attardi (2006)		87.24%	$O(n^7)$	2
Var. 1: AllDeg1		93.32%	$O(n^7)$	2
Var. 2: All		**95.99%**	$O(n^7)$	3
Var. 3: Alls_0s_1		93.16%	$\boldsymbol{O(n^6)}$	2
	stack buffer			

Figure 1: Attardi's (2006) transition system of degree 2 and our variants. Solid arrows denote the inventory of reduce transitions; each arrow points from the head to the modifier of the edge created by that transition. The *degree* of a transition is the distance between the head and modifier. Green highlights the single degree-3 transition. Thick arrows and gray dotted arrows represent additional and deleted transitions with respect to the original Attardi (2006) system. Coverage refers to the percentage of non-projective sentences (a total of 76,084 extracted from 604,273 training sentences in UD 2.1) that the systems are able to handle.

We employ a tripartite representation for configurations: (σ, β, A), where the three elements are as follows. σ and β are disjoint lists called the *stack* and *buffer*, respectively. Each dependency arc (h, m) in the *resolved arcs set* A has head h and modifier m. For a length-n input sentence w, the initial configuration is $c^s(w) = ([], [0, 1, ..., n], \varnothing)$ where the 0 in the initial buffer denotes a special node representing the root of the parse tree. All terminal configurations have an empty buffer and a stack containing only 0.

Indexing from 0, we write s_i and b_j to denote item i on the stack (starting from the right) and item j on the buffer (from the left), respectively. We use vertical bars to separate different parts of the buffer or stack. For example, when concerned with the top three stack items and the first item on the buffer, we may write $\sigma|s_2|s_1|s_0$ and $b_0|\beta$.

2.2 Attardi's (2006) System

We now introduce the widely-used Attardi (2006) system, which includes transitions that create arcs between non-consecutive subtrees, thus allowing it to produce some non-projective trees. To simplify exposition, here we present Cohen et al.'s (2011) isomorphic version.

The set of transitions consists of a shift transition (sh) and four reduce transitions (re). A *shift* moves the first buffer item onto the stack:

$$\mathsf{sh}[(\sigma, b_0|\beta, A)] = (\sigma|b_0, \beta, A).$$

A *reduce* transition $\mathsf{re}_{h,m}$ creates a dependency arc between h (head) and m (modifier) and reduces m. For example,

$$\mathsf{re}_{s_0,s_1}[(\sigma|s_1|s_0, \beta, A)] = (\sigma|s_0, \beta, A \cup \{(s_0, s_1)\}) .$$

Row 1 of Fig. 1 depicts the four Attardi reduces.

The distance between h and m in a $\mathsf{re}_{h,m}$ transition is called its *degree*. A system limited to degree-1 transitions can only parse projective sentences. As shown in Fig. 1, Attardi's (2006) system has two degree-2 transitions (re_{s_0,s_2} and re_{s_2,s_0}) that allow it to cover 87.24% of the *non-projective trees* in UD 2.1. More generally, an Attardi system of degree D adds re_{s_0,s_D} and re_{s_D,s_0} to the system of degree $D - 1$.

3 Improving Coverage

A key observation is that a degree-D Attardi system does not contain all possible transitions of degree within D. Since prior empirical work has ascertained that transition systems using more transitions with degree greater than 1 can handle more non-projective treebank trees (Attardi, 2006; Gómez-Rodríguez, 2016), we hypothesize that adding some of these "missing" reduce transitions into the system's inventory should increase coverage. The challenge is to simultaneously maintain run-time guarantees, as there exists a known trade-off between coverage and complexity (Gómez-Rodríguez, 2016). We want to use Cohen et al.'s (2011)'s exact-inference algorithm for Attardi-based degree-D non-projective dependency parsing systems, which was previously analyzed as having $O(n^{3D+1})$ time complexity.[2] **Our contribution** *is systems that improve the degree-2 Attardi (2006) system's non-projective coverage, and yet (i) one has degree* **3** *but still the same* $O(n^7)$ *runtime as Cohen et al. (2011), rather than* $O(n^{3\cdot3+1})$*; and (ii) another has degree 2 but* better *runtime than Cohen et al.'s (2011) system.*

Here, we first sketch the existing exact inference algorithm,[3] and then present our variants.

3.1 Cohen et al.'s (2011) Exact Inference

The main idea of the algorithm is to group transition sequences into equivalence classes and construct longer sequences from shorter ones. Formally, for $m \geqslant 1$, Cohen et al. (2011) define a length-m *computation* as a sequence of m applications of transitions to configurations: $c_0 \xrightarrow{t_1} c_1 \cdots \xrightarrow{t_m} c_m$, where $t_i \in T$ and $t_i(c_{i-1}) = c_i$ for $i \in 1..m$. As depicted in Fig. 2, a length-m *I-computation* $[h_1, i, h_2, h_3, j]$ is any length-m computation where (1) $c_0 = (\sigma|h_1, i|\beta, A)$ and $c_m = (\sigma|h_2|h_3, j|\beta', A')$ for some σ, β, β', A, and A'; and (2) for all $k \in 1..m$, c_k's stack has σ as base and length at least $|\sigma| + 2$. Only condition (1) is relevant to this paper:[4] it states that the net effect of an I-computation is to replace the rightmost item h_1 on the stack with items h_2 and h_3, while advancing the buffer-start from i to j.

The dynamic programming algorithm is specified as a deduction system, where each transition corresponds to a deduction rule. The shift rule is:

$$\mathsf{sh}: \frac{[h_1, i, h_2, h_3, j]}{[h_3, j, h_3, j, j+1]}.$$

Each reduce rule combines two I-computations into a larger I-computation, e.g. (see Fig. 3):

$$\mathsf{re}_{s_0,s_1}: \frac{[h_1, i, h_2, h_3, k] \quad [h_3, k, h_4, h_5, j]}{[h_1, i, h_2, h_5, j]},$$

[2]While $O(n^7)$ or $O(n^{10})$ is not practical, the result is still impressive, since the search space is exponential. Cohen et al. (2011) were inspired by Huang and Sagae's (2010) and Kuhlmann et al.'s (2011) dynamic-programming approach for projective systems.

[3]See Cohen et al. (2011) for full description.

[4]Condition (2) is used for proving completeness of the deduction system (Cohen et al., 2011).

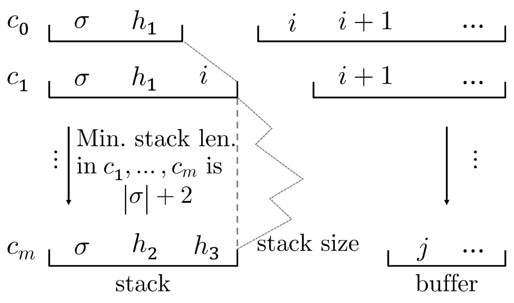


Figure 2: From Cohen et al. (2011, Fig. 2): schematic of I-computation $[h_1, i, h_2, h_3, j]$.

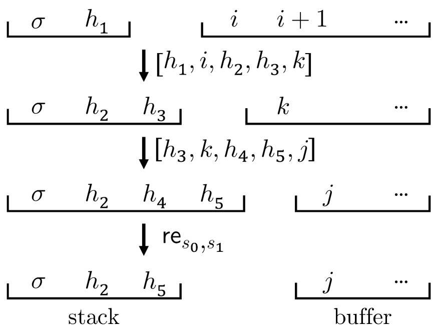


Figure 3: Illustration of deduction rule re_{s_0,s_1}.

with the side condition that h_4 modifies h_5.[5] Other reduce transitions have similar deduction rules, with the same two premises, but a different conclusion depending on the reduced stack item. As an illustration:

$$\mathsf{re}_{s_2,s_0}: \frac{[h_1, i, h_2, h_3, k] \quad [h_3, k, h_4, h_5, j]}{[h_1, i, h_2, h_4, j]}.$$

The goal of deduction is to produce the I-computation $[\epsilon, 0, \epsilon, 0, \epsilon]$, using the shift and reduce deduction rules starting from the axiom $[\epsilon, 0, \epsilon, 0, 1]$, corresponding to the first and mandatory shift transition moving the root node from buffer to stack. ϵ stands for an empty stack or buffer. As analyzed by Cohen et al. (2011), direct tabularization for this deduction system takes $O(n^5)$ space and $O(n^8)$ time. With adaptation of the "hook trick" described in Eisner and Satta (1999), we can reduce the running time to $O(n^7)$.

3.2 Our New Variants

In this section, we modify Cohen et al.'s (2011) set of reduce deduction rules to improve coverage or

[5]This side condition can be interpreted as a grammar rule (for a recognizer) or as an edge to be scored and added to the parse tree (for a parser).

time complexity. Since each such deduction rule corresponds to a reduce transition, each revision to the deduction system yields a variant of Attardi's (2006) parser. In other words, generalization of the deduction system gives rise to a family of non-projective transition-based dependency parsers.

We first explain why there are exactly nine reduce transitions $\mathcal{R} = \{\mathsf{re}_{s_0,s_1}, \mathsf{re}_{s_1,s_0}, \mathsf{re}_{s_0,s_2}, \mathsf{re}_{s_2,s_0}, \mathsf{re}_{s_1,s_2}, \mathsf{re}_{s_2,s_1}, \mathsf{re}_{b_0,s_0}, \mathsf{re}_{b_0,s_1}, \mathsf{re}_{b_0,s_2}\}$ that can be used in Cohen et al.'s (2011) exact inference algorithm, without allowing a reduction with head b_i for $i \geqslant 1$.[6] (Note that Cohen et al.'s (2011) reduce rules are precisely the first four elements of $\mathcal{R}$.) From Fig. 3 we infer that the concatenation of I-computations $[h_1, i, h_2, h_3, k]$ and $[h_3, k, h_4, h_5, j]$ yields a configuration of the form $(\sigma|h_2|h_4|h_5, j|\beta, A)$. For the application of a reduce rule to yield a valid I-computation, by condition (1) of the I-computation definition, first, the head and modifier must be selected from the "exposed" elements h_2, h_4, h_5, and j, corresponding to s_2, s_1, s_0, b_0, respectively; and second, the modifier can only come from the stack. $\mathcal{R}$ is precisely the set of rules satisfying these criteria. Further, every reduce transition from $\mathcal{R}$ is compatible with Eisner and Satta's (1999) "hook trick". This gives us the satisfactory result that the $O(n^7)$ running time upper bound still holds for transitions in $\mathcal{R}$, even though one of them has degree 3.

Next, we consider three notable variants within the family of $\mathcal{R}$-based non-projective transition-based dependency parsers. They are given in Fig. 1, along with their time complexities and empirical coverage statistics. The latter is computed using static oracles (Cohen et al., 2012) on the UD 2.1 dataset (Nivre et al., 2017).[7] We report the global coverage over the 76,084 non-projective sentences from all the training treebanks.

One might assume that adding more degree-1 transitions wouldn't improve coverage of trees with non-crossing edges. On the other hand, since their addition doesn't affect the asymptotic run-time, we define **ALLDEG1** to include all five degree-1 transitions from $\mathcal{R}$ into the Attardi (2006) system. Surprisingly, using ALLDEG1 improves non-projective coverage from 87.24% to 93.32%.

Furthermore, recall that we argued above that, by construction, using any of the transitions in $\mathcal{R}$ still preserves the original $O(n^7)$ run-time upper bound for Cohen et al.'s (2011) exact inference algorithm. We therefore define **ALL** to include all 9 reduce transitions in $\mathcal{R}$; it runs in time $O(n^7)$ despite the fact that $\mathsf{re}_{b_0,s_2} \in \mathcal{R}$ has degree 3, a significant improvement over the best previously-known bound for degree-3 systems of $O(n^{10})$. Moreover, as shown in Fig. 1, this variant improves non-projective coverage to 95.54%.

Now, if our goal is to reduce run-time, we can start with an Attardi (2006) system of degree 1 instead of 2, which, as previously mentioned, can only handle projective sentences, but which has runtime $O(n^{(3\cdot1)+1}) = O(n^4)$. Reasoning about the analog of $\mathcal{R}$ with respect to Kuhlmann et al.'s (2011) exact inference algorithm — the projective predecessor of Cohen et al. (2011) — brings us to the degree-2 set of reduce rules $\{\mathsf{re}_{s_0,s_1}, \mathsf{re}_{s_1,s_0}, \mathsf{re}_{b_0,s_1}\}$. This system, however, can only handle leftward non-projective arcs.

Instead, we return to ALL, but discard transitions reducing s_2, thus deriving **ALL**s_0s_1, which still produces both left and right non-projective arcs, but has a run-time lower than $O(n^7)$, which we show as follows. Since s_2 cannot be reduced, when concatenating $[h_1, i, h_2, h_3, k]$ and $[h_3, k, h_4, h_5, j]$, the larger I-computation we deduce will be either $[h_1, i, h_2, h_4, j]$ or $[h_1, i, h_2, h_5, j]$, so that the first three indices of the conclusion item remain the same as those of the first premise. In addition, the only remaining deduction rule, a shift, produces deduction items of the form $[h_1, j, h_1, j, j+1]$. Hence, all derivable items will be of the form $[h_1, i, h_1, h_3, j]$, with only four unique indices, instead of five. It follows that the exact inference algorithm for this variant runs in $O(n^6)$ time, improving from $O(n^7)$. The tabularization takes $O(n^4)$ space, a reduction from the original $O(n^5)$ as well. In terms of empirical coverage, this new system can handle 93.16% of the non-projective sentences in UD 2.1, more than Attardi's (2006) system, but fewer than our other two variants.

Generally, for a *degree-D* Attardi (2006)-based system, one may apply our first two variants to improve its non-projective coverage while maintaining the previously-analyzed $O(n^{3D+1})$ time complexity, or the third variant to reduce its time complexity down to $O(n^{3D})$, and space complexity from $O(n^{2D+1})$ to $O(n^{2D})$.

[6]Such reductions might prove interesting in the future.

[7]We also compare results from symbolic execution of the dynamic programming algorithms on short sentences as a double check.

4 Conclusion

We have introduced a family of variants of Cohen et al.'s (2011) Attardi-based transition system and its associated dynamic programming algorithm. Among these, we have highlighted novel algorithms that (1) increase non-projective coverage without affecting computational complexity for exact inference, and (2) improve the time and space complexity for exact inference, even while providing better coverage than the original parser. Specifically, our $\textsc{All}s_0s_1$ runs in $O(n^6)$ time and $O(n^4)$ space (improving from $O(n^7)$ and $O(n^5)$, respectively) while providing coverage of 93.16% of the non-projective sentences in UD 2.1.

Exact inference for transition-based parsers has recently achieved state-of-the-art results in projective parsing (Shi et al., 2017). The complexity improvements achieved in this paper are a step towards making their exact-inference, projective approach extensible to practical, wide-coverage non-projective parsing.

Acknowledgments

We thank the reviewers for their time and careful reading, and reviewer 2 for suggestions that improved our presentation. TS and LL were supported in part by a Google Focused Research Grant to Cornell University. 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). 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

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.

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.

Shay B. Cohen, Carlos Gómez-Rodríguez, and Giorgio Satta. 2012. Elimination of spurious ambiguity in transition-based dependency parsing. *ArXiv e-prints*, 1206.6735.

Jason Eisner and Giorgio Satta. 1999. Efficient parsing for bilexical context-free grammars and head automaton grammars. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 457–464, College Park, Maryland, USA.

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

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.

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.

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.

Joakim Nivre, Željko Agić, Lars Ahrenberg, Lene Antonsen, Maria Jesus Aranzabe, Masayuki Asahara, Luma Ateyah, Mohammed Attia, Aitziber Atutxa, Liesbeth Augustinus, Elena Badmaeva, Miguel Ballesteros, Esha Banerjee, Sebastian Bank, Verginica Barbu Mititelu, John Bauer, Kepa Bengoetxea, Riyaz Ahmad Bhat, Eckhard Bick, Victoria Bobicev, Carl Börstell, Cristina Bosco, Gosse Bouma, Sam Bowman, Aljoscha Burchardt, Marie Candito, Gauthier Caron, Gülşen Cebiroğlu Eryiğit, Giuseppe G. A. Celano, Savas Cetin, Fabricio Chalub, Jinho Choi, Silvie Cinková, Çağrı Çöltekin, Miriam Connor, Elizabeth Davidson, Marie-Catherine de Marneffe, Valeria de Paiva, Arantza Diaz de Ilarraza, Peter Dirix, Kaja Dobrovoljc, Timothy Dozat, Kira Droganova, Puneet Dwivedi, Marhaba Eli, Ali Elkahky, Tomaž Erjavec, Richárd Farkas, Hector Fernandez Alcalde, Jennifer Foster, Cláudia Freitas, Katarína Gajdošová, Daniel Galbraith, Marcos Garcia, Moa Gärdenfors, Kim Gerdes, Filip Ginter, Iakes Goenaga, Koldo Gojenola, Memduh Gökırmak, Yoav Goldberg, Xavier

Gómez Guinovart, Berta Gonzáles Saavedra, Matias Grioni, Normunds Grūzītis, Bruno Guillaume, Nizar Habash, Jan Hajič, Jan Hajič jr., Linh Hà Mỹ, Kim Harris, Dag Haug, Barbora Hladká, Jaroslava Hlaváčová, Florinel Hociung, Petter Hohle, Radu Ion, Elena Irimia, Tomáš Jelínek, Anders Johannsen, Fredrik Jørgensen, Hüner Kaşıkara, Hiroshi Kanayama, Jenna Kanerva, Tolga Kayadelen, Václava Kettnerová, Jesse Kirchner, Natalia Kotsyba, Simon Krek, Veronika Laippala, Lorenzo Lambertino, Tatiana Lando, John Lee, Phương Lê Hồng, Alessandro Lenci, Saran Lertpradit, Herman Leung, Cheuk Ying Li, Josie Li, Keying Li, Nikola Ljubešić, Olga Loginova, Olga Lyashevskaya, Teresa Lynn, Vivien Macketanz, Aibek Makazhanov, Michael Mandl, Christopher Manning, Cătălina Mărănduc, David Mareček, Katrin Marheinecke, Héctor Martínez Alonso, André Martins, Jan Mašek, Yuji Matsumoto, Ryan McDonald, Gustavo Mendonça, Niko Miekka, Anna Missilä, Cătălin Mititelu, Yusuke Miyao, Simonetta Montemagni, Amir More, Laura Moreno Romero, Shinsuke Mori, Bohdan Moskalevskyi, Kadri Muischnek, Kaili Müürisep, Pinkey Nainwani, Anna Nedoluzhko, Gunta Nešpore-Bērzkalne, Lương Nguyễn Thị, Huyền Nguyễn Thị Minh, Vitaly Nikolaev, Hanna Nurmi, Stina Ojala, Petya Osenova, Robert Östling, Lilja Øvrelid, Elena Pascual, Marco Passarotti, Cenel-Augusto Perez, Guy Perrier, Slav Petrov, Jussi Piitulainen, Emily Pitler, Barbara Plank, Martin Popel, Lauma Pretkalniņa, Prokopis Prokopidis, Tiina Puolakainen, Sampo Pyysalo, Alexandre Rademaker, Loganathan Ramasamy, Taraka Rama, Vinit Ravishankar, Livy Real, Siva Reddy, Georg Rehm, Larissa Rinaldi, Laura Rituma, Mykhailo Romanenko, Rudolf Rosa, Davide Rovati, Benoît Sagot, Shadi Saleh, Tanja Samardžić, Manuela Sanguinetti, Baiba Saulīte, Sebastian Schuster, Djamé Seddah, Wolfgang Seeker, Mojgan Seraji, Mo Shen, Atsuko Shimada, Dmitry Sichinava, Natalia Silveira, Maria Simi, Radu Simionescu, Katalin Simkó, Mária Šimková, Kiril Simov, Aaron Smith, Antonio Stella, Milan Straka, Jana Strnadová, Alane Suhr, Umut Sulubacak, Zsolt Szántó, Dima Taji, Takaaki Tanaka, Trond Trosterud, Anna Trukhina, Reut Tsarfaty, Francis Tyers, Sumire Uematsu, Zdeňka Urešová, Larraitz Uria, Hans Uszkoreit, Sowmya Vajjala, Daniel van Niekerk, Gertjan van Noord, Viktor Varga, Eric Villemonte de la Clergerie, Veronika Vincze, Lars Wallin, Jonathan North Washington, Mats Wirén, Tak-sum Wong, Zhuoran Yu, Zdeněk Žabokrtský, Amir Zeldes, Daniel Zeman, and Hanzhi Zhu. 2017. Universal Dependencies 2.1. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

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

Tianze Shi, Liang Huang, and Lillian Lee. 2017. 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.