Treebank Embedding Vectors for Out-of-Domain Dependency Parsing

Joachim Wagner and **James Barry** and **Jennifer Foster**
ADAPT Centre
School of Computing, Dublin City University, Ireland
firstname.lastname@adaptcentre.ie

Abstract

A recent advance in monolingual dependency parsing is the idea of a treebank embedding vector, which allows all treebanks for a particular language to be used as training data while at the same time allowing the model to prefer training data from one treebank over others and to select the preferred treebank at test time. We build on this idea by 1) introducing a method to predict a treebank vector for sentences that do not come from a treebank used in training, and 2) exploring what happens when we move away from predefined treebank embedding vectors during test time and instead devise tailored interpolations. We show that 1) there are interpolated vectors that are superior to the predefined ones, and 2) treebank vectors can be predicted with sufficient accuracy, for nine out of ten test languages, to match the performance of an oracle approach that knows the most suitable predefined treebank embedding for the test set.

1 Introduction

The Universal Dependencies project (Nivre et al., 2016) has made available multiple treebanks for the same language annotated according to the same scheme, leading to a new wave of research which explores ways to use multiple treebanks in monolingual parsing (Shi et al., 2017; Sato et al., 2017; Che et al., 2017; Stymne et al., 2018).

Stymne et al. (2018) introduced a *treebank embedding*. A single model is trained on the concatenation of the available treebanks for a language, and the input vector for each training token includes the treebank embedding which encodes the treebank the token comes from. At test time, all input vectors in the test set of the same treebank are also assigned this treebank embedding vector. Stymne et al. (2018) show that this approach is superior to mono-treebank training and to plain treebank concatenation. Treebank embeddings perform at about the same level as training on multiple treebanks and tuning on one, but they argue that a treebank embedding approach is preferable since it results in just one model per language.

What happens, however, when the input sentence does not come from a treebank? Stymne et al. (2018) simulate this scenario with the Parallel Universal Dependency (PUD) test sets. They define the notion of a *proxy* treebank which is the treebank to be used for a treebank embedding when parsing sentences that do not come from any of the training treebanks. They empirically determine the best proxy treebank for each PUD test set by testing with each treebank embedding. However, the question remains what to do with sentences for which no gold parse is available, and for which we do not know the best proxy.

We investigate the problem of choosing treebank embedding vectors for new, possibly out-of-domain, sentences. In doing so, we explore the usefulness of *interpolated* treebank vectors which are computed via a weighted combination of the predefined fixed ones. In experiments with Czech, English and French, we establish that useful interpolated treebank vectors exist. We then develop a simple k-NN method based on sentence similarity to choose a treebank vector, either fixed or interpolated, for sentences or entire test sets, which, for 9 of our 10 test languages matches the performance of the best (oracle) proxy treebank.

2 Interpolated Treebank Vectors

Following recent work in neural dependency parsing (Chen and Manning, 2014; Ballesteros et al., 2015; Kiperwasser and Goldberg, 2016; Zeman et al., 2017, 2018), we represent an input token by concatenating various vectors. In our experiments, each word w_i in a sentence $S = (w_1,...,w_n)$ is a

concatenation of 1) a dynamically learned word vector, 2) a word vector obtained by passing the k_i characters of w_i through a BiLSTM and 3), following Stymne et al. (2018), a treebank embedding to distinguish the m training treebanks:

$$\begin{aligned}\mathbf{e}(i) = &\mathbf{e_1}(w_i) \\ &\circ \mathbf{biLSTM}(\mathbf{e_2}(ch_{i,1}), ..., \mathbf{e_2}(ch_{i,k_i})) \\ &\circ \mathbf{f}\end{aligned} \tag{1}$$

Stymne et al. (2018) use

$$\mathbf{f} = e_3(t^\star) \tag{2}$$

where $t^\star \in 1, ..., m$ is the source treebank for sentence S or if S does not come from one of the m treebanks, a choice of one of these (the proxy treebank). We change $\mathbf{f}$ during test time to

$$\mathbf{f} = \sum_{t=1}^{m} \alpha_t e_3(t) \tag{3}$$

where there are m treebanks for the language in question and $\sum_{t=1}^{m} \alpha_t = 1$.

3 Data and Resources

For all experiments, we use UD v2.3 (Nivre et al., 2018). We choose Czech, English and French as our development languages because they each have four treebanks (excluding PUD), allowing us to train on three treebanks and test on a fourth. For testing, we use the PUD test sets for languages for which there are at least two other treebanks with training data: Czech, English, Finnish, French, Italian, Korean, Portuguese, Russian, Spanish and Swedish. Following Stymne et al. (2018), we use the transition-based parser of de Lhoneux et al. (2017) with the token input representations as Eq. 1 above. Source code of our modified parser and helper scripts to carry out the experiments are available online.[1]

4 Are Interpolated Treebank Vectors Useful?

We attempt to ascertain how useful interpolated treebank embedding vectors are by examining the labelled attachment score (LAS) of trees parsed with different interpolated treebank vectors. For each of our three development languages, we train multi-treebank parsing models on the four combinations of three of the four available treebanks and we test each model on the development sets of all four treebanks, i. e. three in-domain parsing settings and one out-of-domain setting.[2]

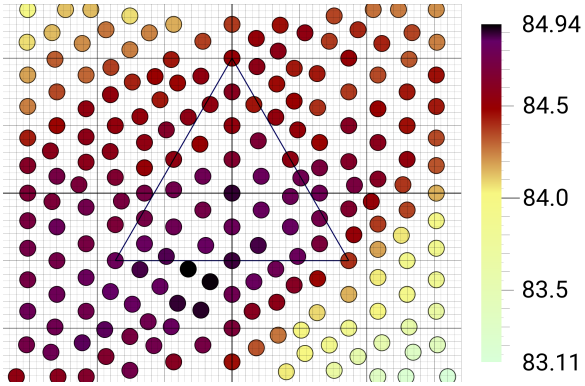

Figure 1: LAS in the treebank vector weight space ($m = 3$) for `cs_cltt+fictree+pdt` on `cs_cac-dev` with the second seed.

Since $m = 3$ and $\sum_{t=1}^{m} \alpha_t = 1$, all treebank vectors lie in a plane and we can visualise LAS results in colour plots. As the treebank vectors can have arbitrary distances, we plot (and sample) in the weight space $\mathbf{R}^m$. We include the equilateral triangle spanned by the three fixed treebank embedding vectors in our plots. Points outside the triangle can be reached by allowing negative weights $\alpha_t < 0$.

We obtain treebank LAS and sentence-level LAS for 200 weight vectors sampled from the weight space, including the corners of the triangle, and repeat with different seeds for parameter initialisation and training data shuffling. Rather than sampling at random, points are chosen so that they are somewhat symmetrical and evenly distributed.

Figure 1 shows the development set LAS on `cs_cac-dev` for a model trained on `cs_cltt+fictree+pdt` with the second seed. We create 432 such plots for nine seeds, four training configurations, four development sets and three languages. The patterns vary with each seed and configuration. The smallest LAS range within a plot is 87.8 to 88.3 (`cs_cac+cltt+pdt` on `cs_pdt` with the seventh seed). The biggest LAS range is 59.7 to 76.8 (`fr_gsd+sequoia+spoken` on `fr_spoken` with the fifth seed).

The location of the fixed treebank vectors $e_3(t)$ are at the corners of the triangle in each graph. For in-domain settings one or two corners usually have LAS close to the highest LAS in the plot. The

[1] https://github.com/jowagner/tbev-prediction

[2] An in-domain example is testing a model trained on `cs_cac+cltt+fictree` on `cs_cac`, and an out-of-domain example is testing the same model on `cs_pdt`.

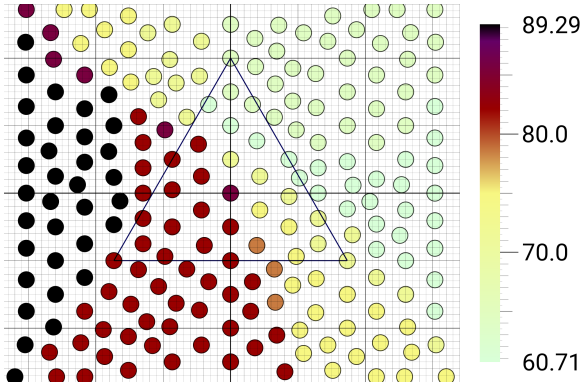

Figure 2: LAS in the treebank vector weight space ($m = 3$) for sentence 2 of `en_partut-dev` (28 tokens) with `en_ewt+gum+lines` and our first seed.

best LAS scores (black circles), however, are often located outside the triangle, i. e. negative weights are needed to reach it.

Turning to sentence-level LAS, Figure 2 shows the LAS for an individual example sentence rather than an entire development set. This sentence is taken from `en_partut-dev` and is parsed with a model trained on `en_ewt+gum+lines`. For this 28-token sentence, LAS can only change in steps of 1/28 and 34 of the 200 treebank embedding weight points share the top score. Negative weights are needed to reach these points outside the triangle.

Over all development sentences and parsing models, an interpolated treebank vector achieves highest LAS for 99.99% of sentences: In 78.07% of cases, one of the corner vectors also achieves the highest LAS and in the remaining 21.92%, interpolated vectors are needed. It is also worth noting that, for 39% of sentences, LAS does not depend on the treebank vectors at all, at least not in the weight range explored.

Often, LAS changes from one side to another side of the graph. The borders have different orientation and sharpness. The fraction of points with highest LAS varies from few to many. The same is true for the fraction of points with lowest LAS. Noise seems to be low. Most data points match the performance of their neighbours, i. e. the scores are not sensitive to small changes of the treebank weights, suggesting that the observed differences are not just random numerical effects.

This preliminary analysis suggests that useful interpolated treebank vectors do exist. Our next step is to try to predict them. In all subsequent experiments, we focus on the out-of-domain setting, i. e. each multi-treebank model is tested on a treebank not included in training.

5 Predicting Treebank Vectors

We use k-nearest neighbour (k-NN) classification to predict treebank embedding vectors for an individual sentence or a set of sentences at test time. We experiment with 1) allocating the treebank vector for an input *sentence* using the k most similar training *sentences* (`se-se`), and 2) allocating the treebank vector for a *set of input sentences* using the most similar training *treebank* (`tr-tr`).

We will first explain the `se-se` case. For each input sentence, we retrieve from the training data the k most similar sentences and then identify the treebank vectors from the candidate samples that have the highest LAS. To compute similarity, we represent sentences either as tf-idf vectors computed over character n-grams, or as vectors produced by max-pooling over a sentence's ELMo vectors (Peters et al., 2018) produced by averaging all ELMo biLM layers.[3]

We experiment with $k = 1, 3, 9$. For many sentences, several treebank vectors yield the optimal LAS for the most similar retrieved sentence(s), and so we try several tie-breaking strategies, including choosing the vector closest to the uniform weight vector (i. e. each of the three treebanks is equally weighted), re-ranking the list of vectors in the tie according to the LAS of the next most similar sentence, and using the average LAS of the k sentences retrieved to choose the treebank vector. Three treebank vector sample sizes were tried:

1. `fixed`: Only the three fixed treebank vectors, i. e. the corners of the triangle in Fig. 1.
2. $\alpha_t \geq 0$: Negative weights are not used in the interpolation, i. e. only the 32 points inside or on the triangle in Fig. 1.
3. `any`: All 200 weight points shown in Fig. 1.

When retrieving treebanks (`tr-tr`), we use the average of the treebank's sentence representation vectors as the treebank representation and we normalise the vectors to the unit sphere as otherwise the size of the treebank would dominate the location in vector space.

We include oracle versions of each k-NN model in our experiments. The k-NN oracle method is different from the normal k-NN method in that the test data is added to the training data so that the test data itself will be retrieved. This means that a

[3] We use *ELMoForManyLangs* (Che et al., 2018).

Model (se-se)		Lang Avg LAS		
Learning	**Weights**	**Cs**	**En**	**Fr**
random	fixed	82.5	73.4	72.1
random	$\alpha_t \geq 0$	**82.6**	73.9	72.5
random	any	82.4	73.3	72.1
k-NN	fixed	82.6	74.6	**73.8**
k-NN	$\alpha_t \geq 0$	82.6	**74.7**	73.8
k-NN	any	82.6	74.4	73.7
oracle k-NN	fixed	84.1	77.8	77.1
oracle k-NN	$\alpha_t \geq 0$	84.2	79.3	78.6
oracle k-NN	any	85.5	81.0	80.2

Table 1: Development set LAS with per sentence treebank vectors

Model (tr-tr)		Lang Avg LAS		
Learning	**Weights**	**Cs**	**En**	**Fr**
proxy-best	fixed	82.7	74.7	73.8
proxy-worst	fixed	82.3	72.4	70.7
k-NN	fixed	**82.7**	74.6	73.8
k-NN	$\alpha_t \geq 0$	82.7	74.6	**73.8**
k-NN	any	82.7	74.5	73.8
oracle k-NN	fixed	82.7	74.7	73.8
oracle k-NN	$\alpha_t \geq 0$	82.8	75.1	74.2
oracle k-NN	any	82.9	75.1	74.3
equal	n/a	82.7	**74.8**	72.9

Table 2: Development set LAS with one treebank vector for all input sentences

k-NN oracle with $k = 1$ knows exactly what treebank vector is best for each test item while a basic k-NN model has to predict the best vector based on the training data. In the tr-tr setting, our k-NN classifier is selecting one of three treebanks for the fourth test treebank. In the oracle k-NN setting, it selects the test treebank itself and parses the sentences in that treebank with its best-performing treebank vector. When the treebank vector sample space is limited to the vectors for the three training treebanks (fixed), this method is the same as the best-proxy method of Stymne et al. (2018).

6 Results

The development results, averaged over the four development sets for each language, are shown in Tables 1 and 2.[4] As discussed above, upper bounds for k-NN prediction are calculated by including an oracle setting in which the query item is added to the set of items to be retrieved, and k restricted to 1. We are also curious to see what happens when an equal combination of the three fixed vectors (uniform weight vector) is used (equal), and when treebank vectors are selected at random.

Table 1 shows the se-se results. The top section shows the results of randomly selecting a sentence's treebank vector, the middle section shows the k-NN results and the bottom section the oracle k-NN results. The k-NN predictor clearly outperforms the random predictor for English and French, but not for Czech, suggesting that the treebank vector itself plays less of a role for Czech, perhaps due to high domain overlap between the treebanks. The oracle k-NN results indicate not only the substantial room for improvement for the predictor, but also the potential of interpolated vectors since the results improve as the sample space is increased beyond the three fixed vectors.

Table 2 shows the tr-tr results. The first section is the proxy treebank embedding of Stymne et al. (2018) where one of the fixed treebank vectors is used for parsing the development set. We report the best- and worst-performing of the three (proxy-best and proxy-worst). The k-NN methods are shown in the second section of Table 2. The first row of this section (fixed weights) can be directly compared with the proxy-best. For Czech and French, the k-NN method matches the performance of proxy-best. For English, it comes close. Examining the per-treebank English results, k-NN predicts the best proxy treebank for all but en_partut, where it picks the second best (en_gum) instead of the best (en_ewt).

The oracle k-NN results are shown in the third section of Table 2.[5] Although less pronounced than for the more difficult se-se task, they indicate that there is still some room for improving the vector predictor at the document level if interpolated vectors are considered.

Our equal method, that uses the weights (⅓, ⅓, ⅓), is shown in the last row of Table 2. It is the overall best English model. Our best model for Czech is a tr-tr model which just selects from the three fixed treebank vectors. For French, the best is a tr-tr model which selects from interpolated vectors with positive weights. For the PUD languages not used in development, we se-

[4] To reduce noise from random initialisation, we parse each development set nine times with nine different seeds and use the median LAS.

[5] Recall that the first method in this section, oracle fixed, is the same method as proxy-best.

language	m	proxy worst	proxy best	generic	language-specific
cs	4	81.6	**82.5**	**82.5**	**82.5**
en	4	76.4	**82.9**	80.7†	81.7†
es	2	76.1	**80.3**	**80.3**	–
fi	2	52.5	**80.6**	80.5	–
fr	4	74.9	**78.6**	**78.6**	**78.6**
it	3	84.4	**85.5**	**85.5**	–
ko	2	35.5	43.9	**44.0**	–
pt	2	74.6	77.4	**77.6**	–
ru	3	82.6	**83.7**	82.9	–
sv	2	73.7	**74.7**	**74.7**	–

Table 3: PUD Test Set Results: Statistically significant differences between `proxy-best` and our best method are marked with †

lect the hyper-parameters based on average LAS on all 12 development sets. The resulting generic hyper-parameters are the same as those for the best French model: `tr-tr` with interpolated vectors and positive weights.[6]

The PUD test set results are shown in Table 3. For nine out of ten languages we match the oracle method `proxy-best` within a 95% confidence interval.[7] For Russian, the treebank vector of the second-best proxy treebank is chosen, falling 0.8 LAS points behind. Still, this difference is not significant (p=0.055). For English, the generic model also picks the second-best proxy treebank.[8]

7 Conclusion

In experiments with Czech, English and French, we investigated treebank embedding vectors, exploring the ideas of interpolated vectors and vector weight prediction. Our attempts to predict good vector weights using a simple regression model yielded encouraging results. Testing on PUD languages, we match the performance of using the best fixed treebank embedding vector in nine of ten cases within the bounds of statistical significance and in five cases exactly match it.

On the whole, it seems that our predictor is not yet good enough to find interpolated treebank vectors that are clearly superior to the basic, fixed vectors and that we know to exist from the oracle runs. Still, we think it is encouraging that performance did not drop substantially when the set of candidate vectors was widened ($\alpha_t \geq 0$ and 'any'). We do not think the superior treebank vectors found by the oracle runs are simply noise, i. e. model fluctuations due to varied inputs, because the LAS landscape in the weight vector space is not noisy. For individual sentences, LAS is usually constant in large areas and there are clear, sharp steps to the next LAS level. Therefore, we think that there is room for improvement for the predictor to find interpolated vectors which are better than the fixed ones. We plan to explore other methods to predict treebank vectors, e. g. neural sequence modelling, and to apply our ideas to the related task of language embedding prediction for zero-shot learning.

Another area for future work is to explore what information treebank vectors encode. The previous work on the use of treebank vectors in mono- and multi-lingual parsing suggests that treebank vectors encode information that enables the parser to select treebank-specific information where needed while also taking advantage of treebank-independent information available in the training data. The type of information will depend on the selection of treebanks, e. g. in a polyglot setting the vector may simply encode the language, and in a monolingual setting such as ours it may encode annotation or domain differences between the treebanks.

Interpolating treebank vectors adds a layer of opacity, and, in future work, it would be interesting to carry out experiments with synthetic data, e. g. varying the number of unknown words, to get a better understanding of what they may be capturing.

Future work should also test even simpler strategies which do not use the LAS of previous parses to gauge the best treebank vector, e. g. always picking the largest treebank.

Acknowledgments

This research is supported by Science Foundation Ireland through the ADAPT Centre for Digital Content Technology, which is funded under the SFI Research Centres Programme (Grant 13/RC/2106) and is co-funded under the European Regional Development Fund. We thank the reviewers for their inspiring questions and detailed feedback.

[6] While the k-NN models selected for final testing use char-n-gram-based sentence representations, ELMo representations are competitive.

[7] Statistical significance is tested with udapi-python (https://github.com/udapi/udapi-python).

[8] For Korean PUD, LAS scores are surprisingly low given that development results on `ko_gsd` and `ko_kaist` are above 76.5 for all seeds. A run with a mono-treebank model confirms low performance on Korean PUD. According to a reviewer, there are known differences in the annotation between the Korean UD treebanks.

References

Miguel Ballesteros, Chris Dyer, and Noah A. Smith. 2015. Improved transition-based parsing by modeling characters instead of words with lstms. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 349–359, Lisbon, Portugal. Association for Computational Linguistics.

Wanxiang Che, Jiang Guo, Yuxuan Wang, Bo Zheng, Huaipeng Zhao, Yang Liu, Dechuan Teng, and Ting Liu. 2017. The hit-scir system for end-to-end parsing of universal dependencies. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 52–62. Association for Computational Linguistics.

Wanxiang Che, Yijia Liu, Yuxuan Wang, Bo Zheng, and Ting Liu. 2018. Towards better UD parsing: Deep contextualized word embeddings, ensemble, and treebank concatenation. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 55–64, Brussels, Belgium. Association for Computational Linguistics.

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

Miryam de Lhoneux, Sara Stymne, and Joakim Nivre. 2017. Arc-hybrid non-projective dependency parsing with a static-dynamic oracle. In *Proceedings of the 15th International Conference on Parsing Technologies*, pages 99–104, Pisa, Italy. Association for Computational Linguistics.

Joakim Nivre, Mitchell Abrams, Željko Agić, Lars Ahrenberg, Lene Antonsen, Katya Aplonova, Maria Jesus Aranzabe, Gashaw Arutie, Masayuki Asahara, Luma Ateyah, Mohammed Attia, Aitziber Atutxa, Liesbeth Augustinus, Elena Badmaeva, Miguel Ballesteros, Esha Banerjee, Sebastian Bank, Verginica Barbu Mititelu, Victoria Basmov, John Bauer, Sandra Bellato, Kepa Bengoetxea, Yevgeni Berzak, Irshad Ahmad Bhat, Riyaz Ahmad Bhat, Erica Biagetti, Eckhard Bick, Rogier Blokland, Victoria Bobicev, Carl Börstell, Cristina Bosco, Gosse Bouma, Sam Bowman, Adriane Boyd, Aljoscha Burchardt, Marie Candito, Bernard Caron, Gauthier Caron, Gülşen Cebiroğlu Eryiğit, Flavio Massimiliano Cecchini, Giuseppe G. A. Celano, Slavomír Čéplö, Savas Cetin, Fabricio Chalub, Jinho Choi, Yongseok Cho, Jayeol Chun, Silvie Cinková, Aurélie Collomb, Çağrı Çöltekin, Miriam Connor, Marine Courtin, Elizabeth Davidson, Marie-Catherine de Marneffe, Valeria de Paiva, Arantza Diaz de Ilarraza, Carly Dickerson, Peter Dirix, Kaja Dobrovoljc, Timothy Dozat, Kira Droganova, Puneet Dwivedi, Marhaba Eli, Ali Elkahky, Binyam Ephrem, Tomaž Erjavec, Aline Etienne, Richárd Farkas, Hector Fernandez Alcalde, Jennifer Foster, Cláudia Freitas, Katarína Gajdošová, Daniel Galbraith, Marcos Garcia, Moa Gärdenfors, Sebastian Garza, Kim Gerdes, Filip Ginter, Iakes Goenaga, Koldo Gojenola, Memduh Gökırmak, Yoav Goldberg, Xavier Gómez Guinovart, Berta Gonzáles Saavedra, Matias Grioni, Normunds Grūzītis, Bruno Guillaume, Céline Guillot-Barbance, Nizar Habash, Jan Hajič, Jan Hajič jr., Linh Hà Mỹ, Na-Rae Han, Kim Harris, Dag Haug, Barbora Hladká, Jaroslava Hlaváčová, Florinel Hociung, Petter Hohle, Jena Hwang, Radu Ion, Elena Irimia, Ọlájídé Ishola, Tomáš Jelínek, Anders Johannsen, Fredrik Jørgensen, Hüner Kaşıkara, Sylvain Kahane, Hiroshi Kanayama, Jenna Kanerva, Boris Katz, Tolga Kayadelen, Jessica Kenney, Václava Kettnerová, Jesse Kirchner, Kamil Kopacewicz, Natalia Kotsyba, Simon Krek, Sookyoung Kwak, Veronika Laippala, Lorenzo Lambertino, Lucia Lam, Tatiana Lando, Septina Dian Larasati, Alexei Lavrentiev, John Lee, Phuong Lê H`ông, Alessandro Lenci, Saran Lertpradit, Herman Leung, Cheuk Ying Li, Josie Li, Keying Li, KyungTae Lim, Nikola Ljubešić, Olga Loginova, Olga Lyashevskaya, Teresa Lynn, Vivien Macketanz, Aibek Makazhanov, Michael Mandl, Christopher Manning, Ruli Manurung, Cătălina Mărănduc, David Mareček, Katrin Marheinecke, Héctor Martínez Alonso, André Martins, Jan Mašek, Yuji Matsumoto, Ryan McDonald, Gustavo Mendonça, Niko Miekka, Margarita Misirpashayeva, Anna Missilä, Cătălin Mititelu, Yusuke Miyao, Simonetta Montemagni, Amir More, Laura Moreno Romero, Keiko Sophie Mori, Shinsuke Mori, Bjartur Mortensen, Bohdan Moskalevskyi, Kadri Muischnek, Yugo Murawaki, Kaili Müürisep, Pinkey Nainwani, Juan Ignacio Navarro Horñiacek, Anna Nedoluzhko, Gunta Nešpore-Bērzkalne, Luong Nguy˜ên Thị, Huy`ên Nguy˜ên Thị Minh, Vitaly Nikolaev, Rattima Nitisaroj, Hanna Nurmi, Stina Ojala, Adédayọ̀ Olúòkun, Mai Omura, Petya Osenova, Robert Östling, Lilja Øvrelid, Niko Partanen, Elena Pascual, Marco Passarotti, Agnieszka Patejuk, Guilherme Paulino-Passos, Siyao Peng, Cenel-Augusto Perez, Guy Perrier, Slav Petrov, Jussi Piitulainen, Emily Pitler, Barbara Plank, Thierry Poibeau, Martin Popel, Lauma Pretkalniņa, Sophie Prévost, Prokopis Prokopidis, Adam Przepiórkowski, Tiina Puolakainen, Sampo Pyysalo, Andriela Rääbis, Alexandre Rademaker, Loganathan Ramasamy, Taraka Rama, Carlos Ramisch, Vinit Ravishankar, Livy Real, Siva Reddy, Georg Rehm, Michael Rießler, Larissa Rinaldi, Laura Rituma, Luisa Rocha, Mykhailo Romanenko, Rudolf Rosa, Davide Rovati, Valentin Roșca, Olga Rudina, Jack Rueter, Shoval Sadde, Benoît Sagot, Shadi Saleh, Tanja Samardžić, Stephanie Samson, Manuela Sanguinetti,

Baiba Saulīte, Yanin Sawanakunanon, Nathan Schneider, Sebastian Schuster, Djamé Seddah, Wolfgang Seeker, Mojgan Seraji, Mo Shen, Atsuko Shimada, Muh Shohibussirri, Dmitry Sichinava, Natalia Silveira, Maria Simi, Radu Simionescu, Katalin Simkó, Mária Šimková, Kiril Simov, Aaron Smith, Isabela Soares-Bastos, Carolyn Spadine, Antonio Stella, Milan Straka, Jana Strnadová, Alane Suhr, Umut Sulubacak, Zsolt Szántó, Dima Taji, Yuta Takahashi, Takaaki Tanaka, Isabelle Tellier, 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, Jing Xian Wang, Jonathan North Washington, Seyi Williams, Mats Wirén, Tsegay Woldemariam, Tak-sum Wong, Chunxiao Yan, Marat M. Yavrumyan, Zhuoran Yu, Zdeněk Žabokrtský, Amir Zeldes, Daniel Zeman, Manying Zhang, and Hanzhi Zhu. 2018. Universal dependencies 2.3. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. 2016. Universal dependencies v1: A multilingual treebank collection. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, pages 1659–1666, Paris, France. European Language Resources Association (ELRA).

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Motoki Sato, Hitoshi Manabe, Hiroshi Noji, and Yuji Matsumoto. 2017. Adversarial training for cross-domain universal dependency parsing. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 71–79. Association for Computational Linguistics.

Tianze Shi, Felix G. Wu, Xilun Chen, and Yao Cheng. 2017. 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. Association for Computational Linguistics.

Sara Stymne, Miryam de Lhoneux, Aaron Smith, and Joakim Nivre. 2018. Parser training with heterogeneous treebanks. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 619–625. Association for Computational Linguistics.

Daniel Zeman, Jan Hajič, Martin Popel, Martin Potthast, Milan Straka, Filip Ginter, Joakim Nivre, and Slav Petrov. 2018. CoNLL 2018 shared task: Multilingual parsing from raw text to universal dependencies. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 1–21, Brussels, Belgium. Association for Computational Linguistics.

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, Çağr 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.