Arc Representation for Graph-based Dependency Parsing

Anonymous ACL submission

Abstract

In this paper, we address the explicit representation of arcs in syntactic dependency parsing, diverging from conventional approaches where parsing algorithms directly manipulate dependency arc scores derived from input token representations. We propose augmenting the parser with an intermediate arc representation, arguing for two main advantages. Firstly, arc vectors encapsulate richer information, enhancing the capabilities of subsequent scoring functions. Secondly, by introducing refinement layers, we enable interactions among vector representations, facilitating the consideration of global long-range dependencies. We demonstrate the efficacy of this approach through empirical evaluations on PTB and UD dependency treebanks.

1 Introduction

Recent graph-based dependency model with powerful neural extractors pioneered in (Kiperwasser and Goldberg, 2016; Dozat and Manning, 2017) make the assumption that the plausibility of a lexical arc or its labelling, as expressed by a score, can be computed directly from the vector representation of the two words linked by this arc. This approach has led to tremendous improvements in parsing accuracy, and consequently this assumption has rarely been questioned with the exception of (Ji et al., 2019) where the structure of the parse forest is exploited to rescore arcs, similarly to forest rerankers for statistical parsers (Huang, 2008). In this paper, we want to challenge this assumption, but with a more pragmatic approach through the lens of deep learning. We propose to learn how to *represent* lexical arcs by vectors, and derive scores from these vectors. This method allows to manipulate these vectors through deep archictectures building the parse forest, and to test this hypothesis we use Transformers to refine arc representations.

2 Model

Before we introduce the arc-centric biaffine architecture for dependency parsing, we first review the standard biaffine parser. Then we highlight the key differences of the proposed approach and their consequences in both arc and label scoring.

Prior to parsing, in all systems, from an input sentence $x_0x_1 \dots x_n$, where x_0 is the dummy root and $\forall 1 \leq i \leq n$, x_i corresponds to the i^{th} token of the sentence, models start by computing contextual representations of tokens $e_0e_1 \dots e_n$. This can be implemented in various ways, for instance with pretrained static word embeddings followed by layers of Bi-LSTMs, or by averaging a subsets of layers from pretrained dynamic word embeddings. These contextual embeddings are further specialized for head and modifier roles. This is implemented as two feed-forward transformations, resulting in two sets of word representations, $h_0h_1 \dots h_n$ for heads and $m_0m_1 \dots m_n$ for modifiers. In the remainder, given a vector v of size d we note v' the vector of size $d + 1$ where $v[i] = v'[i], \forall 1 \leq i \leq d$ and $v'[d + 1]$ is set to 1.

2.1 Biaffine Model

We present the first-order model as introduced in (Dozat and Manning, 2017) and refer readers to (Zhang et al., 2020) for higher-order extensions. The first-order scoring function decomposes the score of a parse tree as the sum of the scores of its arcs, if they form a valid tree (*i.e.* rooted in x_0, connected and acyclic) and can be implemented as a CRF where arc variables are independently scored but connected to a global factor asserting well-formedness constraints. This CRF can be trained efficiently and finding the most probable parse can be performed with well-known algorithms. Still, learning imposes to compute for each sentence *its partition*, the sum of the scores of all parse candidates. While being tractable, this is an overhead compared to computing arc scores independently without tree-shape constraints. Hence, several recent parsers, *e.g.* (Dozat and Manning, 2017), simplify learning by casting it as a head-selection task for each token but the root, *i.e.* arc score predictors are trained without tree constraints. In all cases, tree-constrained CRF or head selection, evaluation is performed by computing the highest-scoring parse (Eisner, 1997; Tarjan, 1977), where arc scores may be replaced by marginal log-probabilities for better performance (Goel and Byrne, 2000).

Arc scores are computed by a biaffine[1] function: for arc $x_i \rightarrow x_j$, Dozat and Manning (2017) define arc score as $s_{ij} = h_i^\top M m'_j$ with trainable matrix M.[2] For embeddings of size d, M has dimensions $d \times (d+1)$.

Arc Labelling is considered a distinct task: at training time arc labelling has its own loss and at prediction time most systems use a pipeline approach where first a tree is predicted, and second each predicted arc is labelled.[3] Scoring is also implemented with a biaffine model: for arc $x_i \rightarrow x_j$, the label logit vector is $l_{ij} = h'^\top_i L m'_j$, with trainable L. For word vectors of size d and for a system with k possible arc labels, L has dimension $(d+1) \times k \times (d+1)$. While we used h and m notations, these specialized word embeddings are given by feed-forward networks different from the ones used for arc scores.

This model relies on two biaffine functions, the first for arc scores returning a scalar per arc, and a second for arc labels scores returning for each arc a vector of label scores. Parameter sharing between arc score and arc labelling computations is limited to contextual word embeddings e.

2.2 Arc Models

Our models differ architecturally in two ways: *(i)* an intermediate vector representation is computed for each arc and *(ii)* both arc and labelling scores are derived from this single arc representation.

For arc $x_i \rightarrow x_j$ we compute vector representation v_{ij}. Again, we use a biaffine function outputting a vector similarly to arc labelling in standard models: $v_{ij} = h_i^\top R m'_j$ for a trainable tensor R with dimensions $d \times r \times d$, where r is the size of the arc vector representation v_{ij}, and is a hyperparameter to be fixed, as is the word embedding size. We recover arc score s_{ij} and arc labelling l_{ij} from v_{ij} by feed-forward transformations: $s_{ij} = F_s(v_{ij})$ and $l_{ij} = F_l(v_{ij})$. Note that there is only one biaffine transformation, and one specialization for head and modifier roles. Finally, remark that this change does not impact the learning objective: parsers are trained the same way.

2.3 Refining with Attention

Arc vectors obtained as above can read information from sentence tokens via contextual embeddings. But we can go further and use Transformers (Vaswani et al., 2017) to leverage attention in order to make arc representations aware of other arc candidates in the parse forest and adjust accordingly, effectively refining representations and realizing a sort of forest reranking. We call v_{ij}^0 the vector computed by the biaffine function over word embeddings described above. Then we successively feed vectors of the form v_{ij}^{p-1} to Transformer encoder layer T^p in order to obtain v_{ij}^p and eventually get the final representation v_{ij}^P. This representation is the one used to compute scores with F_s and F_l. Remark again that this change in the vector representation is compatible with any previously used learning objectives.

The main issue with this model is the space complexity. The softmax operation in Transformers requires multiplying all query/key pairs, the result being stored as a $t \times t$ matrix, where t is the number of elements to consider. In our context, the number of arc candidates is quadratic in the number of tokens in the sentence, so we conclude that memory complexity is $O(n^4)$ where n is the number of tokens. To tackle this issue, we could take advantage of efficient architectures proposed recently *e.g.* Linear Transformers (Qin et al., 2022). Preliminary experiments showed training to be unstable so we resort to a simpler mechanism.

Filtered Attention One way to tackle the softmax memory consumption is to filter input elements. If the number of queries and keys fed to the transformer is linear, we recover a quadratic space complexity. To this end we implement a simple filter F_f to retrieve the best k head candidates per word, reminiscent of some higher-order models prior to deep learning, *e.g.* Koo and Collins (2010)

[1] We ignore bias terms for simplicity.

[2] This additional 1 on the modifier side intuitively makes the expression for s_{ij} mimic the conditional probability of the presence of arc $i \rightarrow j$ given i is classified as a head word, see (Dozat and Manning, 2017) for a detailed discussion.

[3] We remark that Zhang et al. (2021) learn the two separately and merge them at prediction time.

which used arc marginal probabilities to perform filtering. We keep the k highest-scoring $F_f(v^0_{ij})$ for each position j, where k typically equals 10. Kept vectors v^0_{ij} are passed through the transformer as described above, while unkept vectors are considered final. This means that the transformer only processes arcs whose filter scores are among the highest-scoring ones, the intuition being that transformers are only used on ambiguous cases where more context is required to further refine arc or label scores. The filter is trained to predict the arc presence with binary cross entropy, along with arc and labelling losses.

3 Experiments

Data We conduct experiments on the English Penn Treebank (PTB) with Stanford dependencies (de Marneffe and Manning, 2008), as well as the Universal Dependencies 2.2 Treebanks (UD; Nivre et al. 2018), from which we select 12 languages, which we have pseudo-projectivized using (Nivre and Nilsson, 2005). We follow the usual train/dev/test split on all datasets. Contextual word embeddings are obtained from fine-tuned pretrained LLMs. We use RoBERTa$_{\text{large}}$ (Liu et al., 2019) for the PTB and XLM-RoBERTa$_{\text{large}}$ (Conneau et al., 2020) for UD.

Evaluation metrics We use unlabeled and labeled attachment scores (UAS/LAS respectively), with the latter to select best models on development sets. Results are averaged over 8 runs initialized with random seeds. Following Zhang et al. (2020) and others, we omit punctuations during evaluation on PTB but keep them on UD. Finally, we use gold POS tags on UD but omit them for PTB.

Models LOC is the local model from (Zhang et al., 2020), trained with arc cross-entropy and evaluated with the Eisner algorithm (Eisner, 1997). ARCLOC is our model with arc vectors, filtering and one transformer layer.

3.1 Main Results

We first evaluate our model on PTB and compare it with other systems trained with RoBERTa. Results in Table 1 show that our approach with arcs represented by their own vector gives a slight performance improvement on both evaluation metrics over LOC a very strong baseline compared to other state-of-the-art parsers.

	UAS	LAS
Wang and Tu (2020)⋆	96.94	95.37
Gan et al. (2022) Proj⋆	97.24	95.49
Yang and Tu (2022)⋆⋆	97.4	95.8
Amini et al. (2023) w/o POS ⋆⋆	97.4	95.8
LOC	97.32	95.86
ARCLOC (ours)	97.38	**95.91**

Table 1: Results on PTB test with RoBERTa, except for ⋆⋆. For Loc and ours, we average 8 runs with random init. ⋆: from (Gan et al., 2022). ⋆⋆: from (Amini et al., 2023), using XLNet.

The same observation can be carried out on 12 diverse languages from UD, as we can see in Table 2 where we report results from our parsers and others using XLM-RoBERTa for word embeddings. We see that on average, our model with arc representations achieves state-of-the-art results. On ten languages out of twelve our parser gives the best results.

3.2 Ablation study and analysis

Arc representation As we see in Table 3, arc representations can achieve better performance than the base model, and we notice an increase in the UAS/LAS correlated with an increase in arc size up to a plateau.

Size of Arc Vector	UAS	LAS
N/A (LOC)	96.78	95.10
32	96.80	94.97
64	96.85	95.14
128	95.83	95.14
256	96.85	**95.19**
512	**96.86**	95.18

Table 3: PTB dev scores w.r.t. arc vector sizes from 32 to 512. (average of 8 runs, no transformer)

Role of Attention We run an ablation experiment to measure the impact of the Transformer module in our architecture. Table 4 shows that our arc representation is the main factor of improvement of the baseline LOC. Still, Transformers can add a small but consistent improvement. In preliminary experiment, we did not observe any performance gains from adding Trasnformer layers, henceforth all our testing with transformers use one layer.

	bg	ca	cs	de	en	es	fr	it	nl	no	ro	ru	Avg
projective%	99.8	99.6	99.2	97.7	99.6	99.6	99.7	99.8	99.4	99.3	99.4	99.2	99.4
(Wang and Tu, 2020)	91.42	93.75	92.15	82.20	90.91	92.60	89.51	93.79	91.45	91.95	86.50	92.81	90.75
(Gan et al., 2022) Proj	93.61	94.04	93.10	84.97	91.92	92.32	91.69	94.86	92.51	94.07	88.76	94.66	92.21
(Gan et al., 2022) NProj	93.76	94.38	93.72	**85.23**	91.95	92.62	**91.76**	94.79	92.97	94.50	88.67	95.00	92.45
Loc	94.58	94.49	93.99	84.14	92.16	93.86	91.69	94.96	93.86	95.31	90.10	95.60	92.90
ArcLoc	**94.65**	**94.60**	**94.23**	84.26	**92.31**	**93.97**	91.67	**95.21**	**93.91**	**95.52**	**90.25**	**95.69**	**93.02**

Table 2: Test LAS for 12 languages in UD2.2. We use ISO 639-1 codes to represent languages. The projectivity percentage is taken from (Gan et al., 2022)

	UAS	LAS
Loc	96.79	95.11
ArcLoc no transformer	96.86	95.22
ArcLoc	96.86	95.22

Table 4: Impact of arc vectors and Transformers on PTB dev data.

Arc Filtering Table 5 compares parsing UAS with the filter oracle UAS (percentage of correct heads in the set returned by filter). The filter's easily learns which head candidates to keep, even from a limited pool of candidates, we choose to keep 10 potential heads per word to get the highest oracle score.[4]

#Heads	1	2	3	5	10
Oracle	73.6	98.9	99.5	99.7	99.9
Parser	74.4	96.8	96.8	96.8	96.8

Table 5: PTB Dev UAS scores for ArcLoc and its filter's Oracle with different filter sizes (number of kept heads per word).

4 Related Work

In addition to the encoder, attention is widely utilized in syntactic analysis (Mrini et al., 2020; Tian et al., 2020). For instance, Kitaev and Klein (2018) examine the correlation between attention on lexical and positional contents, while Le Roux et al. (2019) employ specialized cross-attention for transition-based parsing. Representing spans has been shown to be beneficial for NLP (Li et al., 2021; Yan et al., 2023; Yang and Tu, 2022) as well as using transformers to enhance them (Zaratiana et al., 2022). Our method is closely related to the use of global attention in Edge Transformers (Bergen et al., 2021). Besides the difference in formalisms of analysis, we do not use any particular attention mask while they use a *triangular* attention. Our use of a learned filter to select arcs that are allowed to interact, is more flexible. Other novel forms of graph attention have been proposed in NodeFormer (Wu et al., 2022). Our use of transformers over arcs is a part of a growing literature on generalizing transformers to relational graph-structured data, as called for by Battaglia et al. (2018). This includes approaches that encode graphs as sets and input them to standard transformers similar to TokenGT (Kim et al., 2022) and Graphormer (Ying et al., 2021). However, we restrict the transformer input to arc vectors excluding node. We differ from approaches that modify standard self-attention either to model structural dependencies (Kim et al., 2017) or implement relative positional encodings (Cai and Lam, 2019) and (Hellendoorn et al., 2020). They do not maintain arc vectors but instead use them to improve node vectors. Lastly, we note that our model bears resemblances to earlier work on reranking for parsing (Collins and Koo, 2005; Charniak and Johnson, 2005; Le and Zuidema, 2014; Hayashi et al., 2013).

5 Conclusion

We presented a new model for graph-based dependency parsing where lexical arcs have their own representation in a high-dimensional vector space, from which their lexical scores are computed. This model demonstrates a clear improvement on parsing metrics over a strong baseline and achieves state-of-the-art performance on PTB and 12 UD corpora. Moreover we show that this architecture is amenable to further processing where arc vectors are refined through transformer encoders. This method could be extended to other tasks, such as constituent parsing or relation extraction. Future research will address this extension and study different architectural choices besides Transformers.

[4]Note that there is no discrepancy in the first column, we can have a UAS score higher than filter's oracle, as an arc can be filtered out and still end up in the parse, our filter only chooses arcs to be processed by the transformer.

6 Limitations

Our system with Transformers relies on the attention mechanism which is quadratic in space and time in the number of elements to consider. Since the number of elements (arcs in our context) is itself quadratic in the number of word tokens, this means that naively the proposed transformer extension is of quadratic complexity. In practice we showed that adding a filtering mechanism is sufficient to revert complexity back to $O(n^2)$, but we leave using efficient transformers, with linear attention mechanism, to future work

7 Ethical Considerations

We do not believe the work presented here further amplifies biases already present in the datasets. Therefore, we foresee no ethical concerns in this work.

References

Afra Amini, Tianyu Liu, and Ryan Cotterell. 2023. Hexatagging: Projective dependency parsing as tagging. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 1453–1464, Toronto, Canada. Association for Computational Linguistics.

Peter W. Battaglia, Jessica B. Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, Caglar Gulcehre, Francis Song, Andrew Ballard, Justin Gilmer, George Dahl, Ashish Vaswani, Kelsey Allen, Charles Nash, Victoria Langston, Chris Dyer, Nicolas Heess, Daan Wierstra, Pushmeet Kohli, Matt Botvinick, Oriol Vinyals, Yujia Li, and Razvan Pascanu. 2018. Relational inductive biases, deep learning, and graph networks.

Leon Bergen, Timothy J. O'Donnell, and Dzmitry Bahdanau. 2021. Systematic generalization with edge transformers. *CoRR*, abs/2112.00578.

Deng Cai and Wai Lam. 2019. Graph transformer for graph-to-sequence learning.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and MaxEnt discriminative reranking. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 173–180, Ann Arbor, Michigan. Association for Computational Linguistics.

Michael Collins and Terry Koo. 2005. Discriminative reranking for natural language parsing. *Computational Linguistics*, 31(1):25–70.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.

Marie-Catherine de Marneffe and Christopher D. Manning. 2008. The Stanford typed dependencies representation. In *Coling 2008: Proceedings of the workshop on Cross-Framework and Cross-Domain Parser Evaluation*, pages 1–8, Manchester, UK. Coling 2008 Organizing Committee.

Timothy Dozat and Christopher D. Manning. 2017. Deep biaffine attention for neural dependency parsing. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Jason Eisner. 1997. Bilexical grammars and a cubic-time probabilistic parser. In *Proceedings of the Fifth International Workshop on Parsing Technologies*, pages 54–65, Boston/Cambridge, Massachusetts, USA. Association for Computational Linguistics.

Leilei Gan, Yuxian Meng, Kun Kuang, Xiaofei Sun, Chun Fan, Fei Wu, and Jiwei Li. 2022. Dependency parsing as MRC-based span-span prediction. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2427–2437, Dublin, Ireland. Association for Computational Linguistics.

Vaibhava Goel and William J. Byrne. 2000. Minimum bayes-risk automatic speech recognition. *Comput. Speech Lang.*, 14(2):115–135.

Katsuhiko Hayashi, Shuhei Kondo, and Yuji Matsumoto. 2013. Efficient stacked dependency parsing by forest reranking. *Transactions of the Association for Computational Linguistics*, 1:139–150.

Vincent J. Hellendoorn, Charles Sutton, Rishabh Singh, Petros Maniatis, and David Bieber. 2020. Global relational models of source code. In *International Conference on Learning Representations*.

Liang Huang. 2008. Forest reranking: Discriminative parsing with non-local features. In *Proceedings of ACL-08: HLT*, pages 586–594. Association for Computational Linguistics.

Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. 2018. Averaging weights leads to wider optima and better generalization. In *34th Conference on Uncertainty in Artificial Intelligence 2018, UAI 2018*, 34th Conference on Uncertainty in Artificial Intelligence 2018, UAI 2018, pages 876–885. Association For Uncertainty in Artificial Intelligence (AUAI). Funding Information: Acknowledgements. This work was supported by NSF IIS-1563887, Samsung Research, Samsung Electronics and Russian Science Foundation grant 17-11-01027. We also thank Vadim Bereznyuk for helpful comments. Funding Information: This work was supported by NSF IIS-1563887, Samsung Research, Samsung Electronics and Russian Science Foundation grant 17-11-01027. We also thank Vadim Bereznyuk for helpful comments. Publisher Copyright: © 34th Conference on Uncertainty in Artificial Intelligence 2018. All rights reserved.; 34th Conference on Uncertainty in Artificial Intelligence 2018, UAI 2018 ; Conference date: 06-08-2018 Through 10-08-2018.

Tao Ji, Yuanbin Wu, and Man Lan. 2019. Graph-based dependency parsing with graph neural networks. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2475–2485, Florence, Italy. Association for Computational Linguistics.

Jinwoo Kim, Dat Tien Nguyen, Seonwoo Min, Sungjun Cho, Moontae Lee, Honglak Lee, and Seunghoon Hong. 2022. Pure transformers are powerful graph learners. In *Advances in Neural Information Processing Systems*.

Yoon Kim, Carl Denton, Luong Hoang, and Alexander M. Rush. 2017. Structured attention networks. In *International Conference on Learning Representations*.

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.

Nikita Kitaev and Dan Klein. 2018. Constituency parsing with a self-attentive encoder. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2676–2686, Melbourne, Australia. Association for Computational Linguistics.

Terry Koo and Michael Collins. 2010. Efficient third-order dependency parsers. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1–11, Uppsala, Sweden. Association for Computational Linguistics.

Phong Le and Willem Zuidema. 2014. The inside-outside recursive neural network model for dependency parsing. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 729–739, Doha, Qatar. Association for Computational Linguistics.

Joseph Le Roux, Antoine Rozenknop, and Mathieu Lacroix. 2019. Representation learning and dynamic programming for arc-hybrid parsing. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, pages 238–248, Hong Kong, China. Association for Computational Linguistics.

Fei Li, ZhiChao Lin, Meishan Zhang, and Donghong Ji. 2021. A span-based model for joint overlapped and discontinuous named entity recognition. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4814–4828, Online. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Khalil Mrini, Franck Dernoncourt, Quan Hung Tran, Trung Bui, Walter Chang, and Ndapa Nakashole. 2020. Rethinking self-attention: Towards interpretability in neural parsing. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 731–742, Online. Association for Computational Linguistics.

Joakim Nivre, Mitchell Abrams, Željko Agić, Lars Ahrenberg, Lene Antonsen, 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, John Bauer, Sandra Bellato, Kepa Bengoetxea, 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, Giuseppe G. A. Celano, 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, 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, Tomáš Jelínek, Anders Johannsen, Fredrik Jørgensen, Hüner Kaşıkara, Sylvain Kahane, Hiroshi Kanayama, Jenna Kanerva, Tolga Kayadelen, Václava Kettnerová, Jesse Kirchner, Natalia Kotsyba, Simon Krek, Sookyoung Kwak, Veronika Laippala, Lorenzo Lambertino, Tatiana Lando, Septina Dian Larasati, Alexei Lavrentiev, John Lee, Phương

Lê Hồng, 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, Anna Missilä, Cătălin Mititelu, Yusuke Miyao, Simonetta Montemagni, Amir More, Laura Moreno Romero, Shinsuke Mori, Bjartur Mortensen, Bohdan Moskalevskyi, Kadri Muischnek, Yugo Murawaki, Kaili Müürisep, Pinkey Nainwani, Juan Ignacio Navarro Horñiacek, Anna Nedoluzhko, Gunta Nešpore-Bērzkalne, Lương Nguyễn Thị, Huyền Nguyễn 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, 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, Shoval Sadde, 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, 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, Veronika Vincze, Lars Wallin, 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 2.2. LINDAT/CLARIAH-CZ digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Joakim Nivre and Jens Nilsson. 2005. Pseudo-projective dependency parsing. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 99–106, Ann Arbor, Michigan. Association for Computational Linguistics.

Zhen Qin, Xiaodong Han, Weixuan Sun, Dongxu Li, Lingpeng Kong, Nick Barnes, and Yiran Zhong. 2022. The devil in linear transformer. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 7025–7041, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

R. E. Tarjan. 1977. Finding optimum branchings. *Networks*, 7(1):25–35.

Yuanhe Tian, Yan Song, Fei Xia, and Tong Zhang. 2020. Improving constituency parsing with span attention. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1691–1703, Online. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Xinyu Wang and Kewei Tu. 2020. Second-order neural dependency parsing with message passing and end-to-end training. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, pages 93–99, Suzhou, China. Association for Computational Linguistics.

Qitian Wu, Wentao Zhao, Zenan Li, David P Wipf, and Junchi Yan. 2022. Nodeformer: A scalable graph structure learning transformer for node classification. In *Advances in Neural Information Processing Systems*, volume 35, pages 27387–27401. Curran Associates, Inc.

Zhaohui Yan, Songlin Yang, Wei Liu, and Kewei Tu. 2023. Joint entity and relation extraction with span pruning and hypergraph neural networks. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7512–7526, Singapore. Association for Computational Linguistics.

Songlin Yang and Kewei Tu. 2022. Headed-span-based projective dependency parsing. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2188–2200, Dublin, Ireland. Association for Computational Linguistics.

Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. 2021. Do transformers really perform badly for graph representation? In *Advances in Neural Information Processing Systems*.

Urchade Zaratiana, Nadi Tomeh, Pierre Holat, and Thierry Charnois. 2022. GNNer: Reducing overlapping in span-based NER using graph neural networks. In *Proceedings of the 60th Annual Meeting* *of the Association for Computational Linguistics: Student Research Workshop*, pages 97–103, Dublin, Ireland. Association for Computational Linguistics.

Xudong Zhang, Joseph Le Roux, and Thierry Charnois. 2021. Strength in numbers: Averaging and clustering effects in mixture of experts for graph-based dependency parsing. In *Proceedings of the 17th International Conference on Parsing Technologies and the IWPT 2021 Shared Task on Parsing into Enhanced Universal Dependencies (IWPT 2021)*, pages 106–118, Online. Association for Computational Linguistics.

Yu Zhang, Zhenghua Li, and Min Zhang. 2020. Efficient second-order TreeCRF for neural dependency parsing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3295–3305, Online. Association for Computational Linguistics.

A Hyperparameters

We mostly use the same hyperparameter settings as Zhang et al. (2020) which are found in their released code. Specifically we adopt the approach they use when training models using BERT, using the average of the 4 last layers to compute our word embeddings. The batch size is 5000, the dimension of the arc MLP is 500, the dropout rate for our MLPs is 0.33, we train our model for 10 epochs and save the one with the best LAS score on the dev data. The learning rates are 1.1e-4, 7e-5 and 7.8e-5 for Loc, ArcLoc without a transformer and ArcLoc respectively if not using stochastic weight averaging (SWA). With SWA, they are 8.3e-5, 3.5e-5, 3.7e-5 before the weight averaging and 5e-6, 2.3e-6 and 3.7e-6 from the fifth epoch onward when we use SWA. The transformer in ArcLoc benefits from its own hyperparameters, while the model warms up for one epoch, the transformer does so for three and has a base learning rate of 3e-3, which becomes 6e-5 when using SWA.

A.1 Efficiency

We trained the bulk of our models on Nvidia v100 GPUs with 32GB of memory but conducted our efficiency comparison on Nvidia a40 GPUs with 48GB of GPU memory. Our models' memory footprint and speed directly depend on the size of the arcs, whether we use a transformer, and whether we filter the arcs.

With the use of our filter, we reduce memory consumption to manageable levels allowing us to use a softmax transformer, however we do notice a slower parsing speed due to the biaffine function's cost when computing the arc vectors.

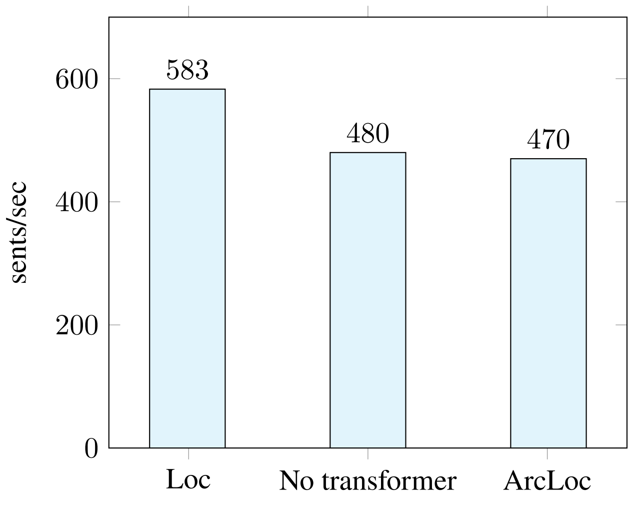

Figure 1: Parsing speed on PTB in sent/s.

Stochastic weight averaging We implement stochastic weight averaging (SWA) introduced in Izmailov et al. (2018) after 4 epochs, which we found lead to consistent improvements in all models after finetuning.