Scaling Graph-Based Dependency Parsing with Arc Vectorization and Attention-Based Refinement

Nicolas Floquet, Joseph Le Roux, Nadi Tomeh, Thierry Charnois
Université Sorbonne Paris Nord, CNRS,
Laboratoire d'Informatique de Paris Nord,
LIPN, F-93430 Villetaneuse, France
{floquet, leroux, tomeh, charnois}@lipn.fr

Abstract

We propose a novel architecture for graph-based dependency parsing that explicitly constructs vectors, from which both arcs and labels are scored. Our method addresses key limitations of the standard two-pipeline approach by unifying arc scoring and labeling into a single network, reducing scalability issues caused by the information bottleneck and lack of parameter sharing. Arc vectors encapsulate richer information, improving the capabilities of scoring functions, additionally, our architecture overcomes limited arc interactions with transformer layers to efficiently simulate higher-order dependencies. Experiments on PTB and UD show that our model outperforms state-of-the-art parsers in both accuracy and efficiency.

1 Introduction

Recent graph-based dependency parsers have adopted a standard architecture (Kiperwasser and Goldberg, 2016; Dozat and Manning, 2017) extended by Zhang et al. (2020). These models consist of two pipelines: one pipeline scores arcs while the other scores their labels. Each pipeline uses independent models to generate specialized head and dependent representations from word embeddings, followed by a biaffine scoring model.

We investigate the *scalability* of this widely-used architecture. Our motivation stems from the observation that not all model architectures scale efficiently with increased parameters. For example, transformer-based language models exhibit predictable scaling laws, where performance consistently improves with more parameters (Kaplan et al., 2020). In contrast, other architectures, *e.g.* CNNs, require careful scaling across multiple dimensions (Tan and Le, 2019). Similar observations have been made in computer vision (Dosovitskiy et al., 2021). Our empirical results show that simply increasing the number of parameters in the standard parsing model does not improve performance. We hypothesize that the core issue lies in the indirect representation of arcs. The model encodes the entire space of possible arcs through word vectors and biaffine scoring, which limits its ability to handle increased complexity. Furthermore, using two scoring networks restricts information flow between arc selection and labeling tasks.

We propose a novel architecture [1] that explicitly constructs vector representations for each arc. By unifying arc scoring and labeling tasks within a single network, our approach allows more parameter sharing and enhances scalability. Finally, we add transformer layers over a selection of arc representations to promote interactions, inspired by higher-order models. The selection is performed by a differential filtering mechanism. This design captures dependencies between arcs while maintaining computational and memory efficiency.

2 Model

We review the standard biaffine parser (Figure 1, left) and then highlight the key differences of our arc-centric approach (Figure 1, right). Prior to parsing, from an input sentence $x_0 x_1 \dots x_n$, 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 embeddings $\boldsymbol{e}_0, \boldsymbol{e}_1, \dots, \boldsymbol{e}_n$. This can be implemented in various ways, *e.g.* with averaged layers from pretrained dynamic word embeddings. These contextual embeddings are further specialized for head and modifier roles using two feed-forward (FFN) transformations. This results in two sets of word representations, $\boldsymbol{h}_0, \boldsymbol{h}_1, \dots, \boldsymbol{h}_n$ for heads and $\boldsymbol{m}_1, \dots, \boldsymbol{m}_n$ for modifiers.

[1] Our code is available at https://github.com/NicolasFlo/ArcLoc

Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 2: Short Papers), pages 722–734
April 29 - May 4, 2025 ©2025 Association for Computational Linguistics

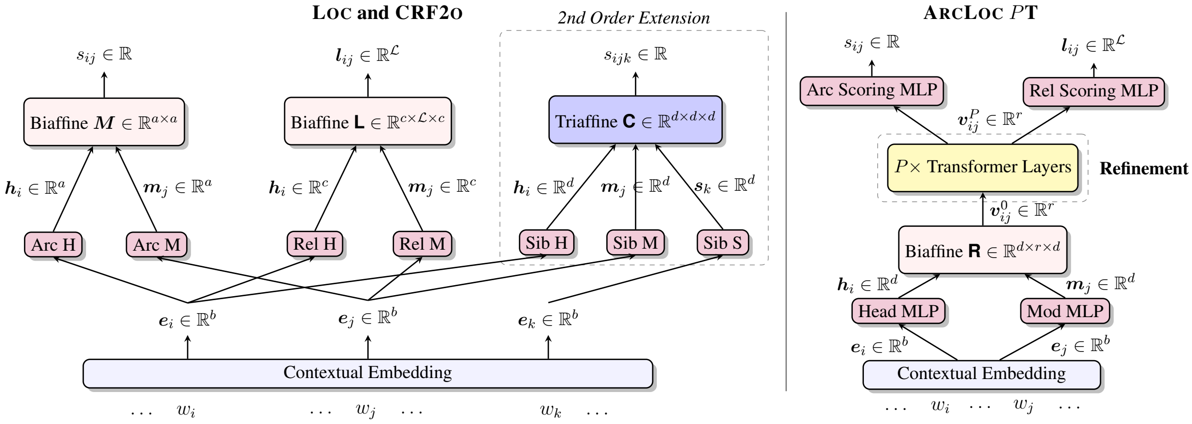

Figure 1: Illustration of both models. LEFT: standard model with 2 (resp. 3) pipelines for LOC (resp. CRF2O) with shared word embeddings. RIGHT: our proposal with a single pipeline and optionally P transformers.

2.1 Standard Model

We present the local and first-order models 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 as the sum of the scores of its arcs, if they form a valid tree, 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 inference is performed with polynomial-time algorithms. Still, learning imposes to compute for each sentence *its partition*, the sum of the (exponentiated) scores of all parse candidates, *i.e.* valid trees. 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) which called this model *local*, simplify learning by casting it as a head-selection task for each word, *i.e.* arc score predictors are trained without tree constraints. In all cases, tree CRF or head selection, evaluation is performed by computing the optimal parse (Eisner, 1997; Tarjan, 1977).

Arc Scores are computed by a biaffine function:[2] for arc $x_i \to x_j$, Dozat and Manning (2017) set arc score to $s_{ij} = \boldsymbol{h}_i^\top \boldsymbol{M} \boldsymbol{m}_j$ with trainable $\boldsymbol{M}$. For embeddings of size d, $\boldsymbol{M}$ has dimensions $d \times d$.

Arc Labeling is considered a distinct task: at training time arc labeling 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 labeled.[3] Labeling is also implemented with a biaffine: for arc $x_i \to x_j$, the label logit vector is $\boldsymbol{l}_{ij} = \boldsymbol{h_i}^\top \mathbf{L} \boldsymbol{m}_j$, with trainable $\mathbf{L}$. For word vectors of size d and for a system with arc label set $\mathcal{L}$, $\mathbf{L}$ has dimension $d \times |\mathcal{L}| \times d$. While we noted them $\boldsymbol{h}$ and $\boldsymbol{m}$, these specialized word embeddings are given by FFNs different from the ones used for arc scores. This model relies on two biaffine functions, one for arc scores returning a scalar per arc, and one for labelings returning for each arc a vector of label scores. Parameter sharing between them is limited to word embeddings $\boldsymbol{e}$.

2.2 Single Pipeline Model

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

For arc $x_i \to x_j$ we compute vector representation $\boldsymbol{v}_{ij}$. Again, we use a biaffine function outputting a vector similarly to arc labeling in standard models: $\boldsymbol{v}_{ij} = \boldsymbol{h}_i^\top \mathbf{R} \boldsymbol{m}_j$ for a trainable tensor $\mathbf{R}$ with dimensions $d \times r \times d$, where r is the size of the arc vector representation $\boldsymbol{v}_{ij}$, and is a hyperparameter as is the word embedding size. We recover arc score s_{ij} and arc labeling $\boldsymbol{l}_{ij}$ from $\boldsymbol{v}_{ij}$ by FFNs: $s_{ij} = F_s(\boldsymbol{v}_{ij})$ and $\boldsymbol{l}_{ij} = F_l(\boldsymbol{v}_{ij})$. Note that there is only one biaffine function, and one specialization for head and modifiers. Finally, remark that this change does not impact the learning objective: parsers are trained the same way.

[2] We ignore bias for the sake of notation.

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

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 $\boldsymbol{v}_{ij}^0$ the vector computed by the biaffine function over word embeddings described above. Then we successively feed vectors of the form $\boldsymbol{v}_{ij}^{p-1}$ to Transformer encoder layer T^p in order to obtain $\boldsymbol{v}_{ij}^p$ and eventually get the final representation $\boldsymbol{v}_{ij}^P$. 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 objective.

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 filtering 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) which used arc marginal probabilities to perform filtering. We keep the k highest-scoring $F_f(\boldsymbol{v}_{ij}^0)$ for each position j, where k typically equals 10. Kept vectors $\boldsymbol{v}_{ij}^0$ are passed through the transformer as described above, while discarded ones are considered final. This means that the transformer only sees arcs whose filter scores are among the highest-scoring ones, the intuition being that transformers are only needed on cases where more context is required to further refine arc or label scores.

Our approach is inspired by the straight-through estimator (Bengio et al., 2013) and is implemented as follows. For each token m we compute the filter scores of all arcs $h \to m$, from their vector representations v_{hm}. Then we add some Gumbel noise (at training time only) and normalize scores via softmax: we obtain probabilities $p(h \to m)$ that we use to sort arcs from most to least probable: $h_1 \to m \dots h_n \to m$.

Finally the k^{th} arc vector returned by the filter for modifier m is computed as:

$$
\begin{aligned}
v_k(m) = \text{argsort}(v_{h_1 m} \dots v_{h_n m})[k] - \\
\text{detach}(\mathbb{E}_{p(\cdot \to m)}[v_{hm}]) + \mathbb{E}_{p(\cdot \to m)}[v_{hm}]
\end{aligned}
$$

During the forward pass the two last terms cancel each other out and $v_k(m)$ is the vector of the k^{th} most probable arc for m, $h_k \to m$. During the backward pass, the first two terms have zero gradient, and the third one amounts to a weighted average of the vectors of arcs $h_1 \to m \dots h_n \to m$, with weights given by their probabilities.

Table 1 compares parsing UAS and the filter's oracle UAS (percentage of correct heads in the set returned by the filter). We keep 10 potential heads per word to get the highest oracle score with a reasonably small sequence of arcs.[4]

#Heads	1	2	3	5	10
Oracle	37.65	75.88	92.48	99.10	99.88
Parser	48.79	78.06	89.69	94.74	96.88

Table 1: PTB Dev UAS scores for ArcLoc 1T and its filter's Oracle with different filter sizes (number of kept heads per word).

3 Experiments

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

[4]Note that there is no discrepancy in the first or second 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.

	Speed	Dev		Test	
		UAS	LAS	UAS	LAS
Wang and Tu (2020)⋆	-	-	-	96.94	95.37
Gan et al. (2022) Proj⋆	-	-	-	97.24	95.49
Yang and Tu (2022a)⋆⋆	-	-	-	97.4	95.8
Amini et al. (2023) ⋆⋆	-	-	-	97.4	95.8
4 million parameters					
Loc	353	96.85	95.16	97.36	**95.90**
CRF2o	144	**96.87**	**95.18**	97.33	95.89
ArcLoc 0T	**356**	96.85	95.16	**97.37**	95.86
ArcLoc 1T	337	96.84	95.13	97.36	95.81
ArcLoc 2T	329	96.81	95.12	97.35	95.82
50 million parameters					
Loc	**333**	96.83	95.16	97.36	95.91
CRF2o	140	96.89	95.19	97.31	95.88
ArcLoc 0T	**333**	**96.91**	**95.26**	**97.37**	95.90
ArcLoc 1T	316	96.90	95.22	97.36	95.87
ArcLoc 2T	308	96.87	95.20	**97.37**	**95.91**
100 million parameters					
Loc	301	96.79	95.12	97.35	95.87
CRF2o	135	96.88	95.18	97.34	95.88
ArcLoc 0T	**319**	**96.92**	**95.29**	**97.38**	**95.92**
ArcLoc 1T	292	96.91	95.23	97.35	95.86
ArcLoc 2T	283	96.90	95.22	97.34	95.85

Table 2: Results on PTB test with RoBERTa, except for ⋆⋆. ⋆: from (Gan et al., 2022). ⋆⋆: from (Amini et al., 2023), using XLNet and no POS tags.

Evaluation We report unlabeled and labeled attachment scores (UAS/LAS), with the latter to select best models on validation. Results are averaged over 8 randomly initialized runs. Following Zhang et al. (2020) and others, we omit punctuations when evaluating on PTB but keep them on UD. Finally, we use gold POS on UD but omit them for PTB.

Models Loc is the local model from (Zhang et al., 2020) trained with arc cross-entropy while CRF2o is their second-order CRF. Vi is the non-projective second-order CRF implementing mean-field variational inference (Wang and Tu, 2020). ArcLoc is our model with arc vectors trained with arc cross-entropy. All models[5] are evaluated with the Eisner algorithm (Eisner, 1997) extended to higher-order for CRF2o on PTB. For UD, we use the MST algorithm (McDonald et al., 2005) for all parsers but CRF2o for which we report deprojectized results. We tested 3 parameter regimes: small (4M), big (50M) and large (100M). Hyperparameter details are given in Appendix A. We include recently published results for comparison.

Main Results Our results on PTB (Table 2) show that our approach is slightly faster and improves LAS on the dev set over Loc and other state-of-the-art parsers. Increasing the number of parameters is beneficial for our model, detrimental for Loc, and has no significant effect for CRF2o. We also remark that on PTB, arc interactions through higher-order scoring or transformer layers have no beneficial impact.

For the 12 tested UD languages Table 3 reports results where we can see that on 11 languages out of 12 a configuration of our parser achieves better performance than Loc, Vi[6] and CRF2o. We notice that on UD the use of transformers allows for better results. By increasing the number of parameters in ArcLoc we manage to achieve state-of-the-art performances at little cost in parsing speed.

Detailed results on dev sets are given in Appendix C and an error analysis in Appendix D.

4 Related Work

Our model, assigning vectors to arcs, *i.e.* the objects to be scored, draws inspiration from the auto-regressive neural approach to parsing (Dyer et al., 2015), as well as from span-based parsers such as (Stern et al., 2017; Zhou and Zhao, 2019) and arc-hybrid parsing in (Le Roux et al., 2019). Recently (Yang and Tu, 2022b) proposed arc vectorization for semantic higher-order dependency parsing based on GNNs.

Refining initial arc representations has also been explored (Strubell and McCallum, 2017; Mohammadshahi and Henderson, 2021). Our model with transformers bears a resemblance to earlier work on forest reranking for parsing (Collins and Koo, 2005; Le and Zuidema, 2014), as we use transformers to promote or demote arcs before scoring and parsing, and to (Ji et al., 2019) where the parse forest is exploited to recompute vectors for words, as opposed to our work where we recompute arc vectors.

Attention is widely utilized in parsing (Mrini et al., 2020; Tian et al., 2020), possibly with ad-hoc constraints on attention (Kitaev and Klein, 2018). Representing spans has been shown to be beneficial for NLP (Li et al., 2021; Yan et al., 2023; Yang and Tu, 2022a) while in (Zaratiana et al., 2022) transformers have also been used to enhance span representations. Our method uses standard softmax attention with a differentiable filter as opposed to rigid constrained masking (Bergen et al., 2021)

[5]Models are based on `https://github.com/yzhangcs/parser` and will be publicly available upon publication.

[6]We only report 4M for Vi since we found training to be unstable otherwise, leading to performance collapse.

Model	#Param (10^6)	Speed	bg	ca	cs	de	en	es	fr	it	nl	no	ro	ru	Avg
(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
VI	4	328	94.31	94.33	94.18	84.08	91.65	93.72	91.48	94.63	93.50	95.10	90.24	95.82	92.75
Loc	4	497	94.54	94.60	94.15	85.54	92.36	93.96	91.70	95.18	94.14	95.34	90.27	95.79	93.13
Loc	50	463	94.41	94.53	94.15	85.28	92.19	93.88	91.72	95.11	94.06	95.19	90.16	95.80	93.04
Loc	100	426	94.37	94.49	94.11	85.25	92.21	93.81	91.75	95.09	93.96	95.18	90.21	95.80	93.02
CRF2o	4	161	94.54	94.32	93.62	85.34	92.30	93.71	91.80	95.24	93.67	95.33	90.10	95.40	92.95
CRF2o	50	158	94.28	94.29	92.84	85.24	92.30	93.73	91.78	95.23	93.48	95.21	90.08	95.42	92.82
CRF2o	100	155	94.28	94.27	93.57	85.19	92.17	93.70	**91.87**	95.26	93.41	95.16	90.18	95.39	92.87
ArcLoc 0T	4	484	94.09	94.22	94.14	84.97	92.10	93.56	91.40	94.87	93.71	94.98	90.01	95.75	92.82
ArcLoc 0T	50	459	94.33	94.50	94.28	85.35	92.35	93.94	91.78	95.06	94.03	95.27	90.32	95.83	93.09
ArcLoc 0T	100	420	94.46	94.61	**94.30**	85.50	92.38	93.94	91.83	95.20	94.17	95.37	90.28	95.88	93.16
ArcLoc 1T	4	451	94.24	94.41	94.15	85.24	92.20	93.71	91.56	94.99	93.95	95.42	90.18	95.74	92.98
ArcLoc 1T	50	421	94.47	94.72	**94.30**	85.52	92.43	94.01	91.71	95.30	**94.22**	95.63	90.34	**95.89**	93.21
ArcLoc 1T	100	393	**94.56**	94.76	94.29	85.62	92.44	**94.07**	91.80	95.29	94.18	**95.71**	**90.38**	**95.89**	**93.25**
ArcLoc 2T	4	449	94.24	94.41	94.13	85.22	92.19	93.73	91.52	95.09	93.88	95.45	90.05	95.75	92.97
ArcLoc 2T	50	419	94.53	94.72	**94.30**	85.60	92.41	94.02	91.75	**95.34**	**94.22**	95.65	90.32	**95.89**	93.23
ArcLoc 2T	100	387	94.55	**94.79**	**94.30**	**85.68**	**92.46**	**94.07**	91.78	95.26	94.11	95.64	90.32	**95.89**	93.24

Table 3: Test LAS for 12 languages in UD2.2. PT is the number of transformer layers.

and other forms of attention (Wu et al., 2022; Kim et al., 2017; Cai and Lam, 2019; Hellendoorn et al., 2020). Our model is part of the literature on generalizing transformers to relational graph-structured data (Battaglia et al., 2018; Kim et al., 2022; Ying et al., 2021).

5 Conclusion

We presented a change in the main graph-based dependency parsing architecture, where arcs have their own vector representation, from which scores are computed. Our model improves parsing metrics and achieves state-of-the-art results on PTB and 11 UD corpora. We also demonstrated that transformer-based refinement simulates higher-order interactions and enhances parameter scalability. Our model can be extended to many other tasks in NLP, such as constituent parsing or relation extraction.

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.

Our model requires more parameters than previously proposed architecture to achieve the same level of performance. This might be an issue for memory limited systems.

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.

8 Acknowledgments

This work was granted access to the HPC resources of IDRIS under the allocation 2023-AD011013732R1 made by GENCI. This work was supported by the Labex EFL (Empirical Foundations of Linguistics, ANR-10-LABX-0083), operated by the French National Research Agency (ANR). This work is supported by the SEMI-AMOR project grant (CE23-2023-0005) given by the French National Research Agency (ANR).

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. *Preprint*, arXiv:1806.01261.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. 2013. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*.

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. *Preprint*, arXiv:1911.07470.

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.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. 2021. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*.

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.

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

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.

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*.

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.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *Preprint*, arXiv:2001.08361.

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.

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

Alireza Mohammadshahi and James Henderson. 2021. Recursive non-autoregressive graph-to-graph transformer for dependency parsing with iterative refinement. *Transactions of the Association for Computational Linguistics*, 9:120–138.

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, Phng 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, 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, Lng 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, 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.

Mitchell Stern, Jacob Andreas, and Dan Klein. 2017. A minimal span-based neural constituency parser. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 818–827, Vancouver, Canada. Association for Computational Linguistics.

Emma Strubell and Andrew McCallum. 2017. Dependency parsing with dilated iterated graph CNNs. In *Proceedings of the 2nd Workshop on Structured Prediction for Natural Language Processing*, pages 1–6, Copenhagen, Denmark. Association for Computational Linguistics.

Mingxing Tan and Quoc Le. 2019. EfficientNet: Rethinking model scaling for convolutional neural networks. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 6105–6114. PMLR.

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. 2022a. 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.

Songlin Yang and Kewei Tu. 2022b. Semantic dependency parsing with edge GNNs. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 6096–6102, Abu Dhabi, United Arab Emirates. 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.

Junru Zhou and Hai Zhao. 2019. Head-Driven Phrase Structure Grammar parsing on Penn Treebank. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2396–2408, Florence, Italy. 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.[7] 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, and also using a batch size of 5000, the dropout rate for all of 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.

[7] `https://github.com/yzhangcs/parser`

Loc We use arc MLP output sizes of 900, 3750, 5500 and label MLP output sizes of 150, 750, 1100 for the small (4×10^6 parameters), big (50×10^6 parameters) and large (100×10^6 parameters) models respectively.

ArcLoc In the small model, the dimension of the arc MLP is 155 without any attention layers, and 150 when using 1 or 2 layers, the arc sizes are 160 when using 0 or 1 layer of attention and 155 when using 2. In the big model, the arc MLP dimension is 500 and the arc size is 192 no matter the number of attention layers we use and for the large model, we increase these sizes to 625 and 256 respectively.

Transformer Our transformer uses a number of attention heads as close to one sixteenth of the arc size as we can get while following the rule that the arc size must be a multiple of the number of attention heads. 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 2.5e-3, which becomes 1.35e-4 when using SWA.

Miscellaneous The learning rates are 8.3e-5 and 3.7e-5 for Loc and ArcLoc respectively before the stochastic weight averaging (SWA) and 5e-6 and 3.7e-6 also respectively from the fifth epoch onward when we use SWA.

Other Parsers For CRF2o, we start from the parameters as Zhang et al. (2020) with a few changes, the learning rates which are the same as Loc, and we have 3 different MLP sizes for the 3 model sizes, for the small model, the sizes are 560, 112 and 112 for the arc, rel, and sib MLPs respectively, for the big model, they are 1675, 335, 335, respectively and for the large model, 2150, 430, and 430, respectively. For Vi, we start with the released code of the implementation by Zhang et al. (2020), and apply the exact same changes we applied to CRF2o.

Parameter Count We use RoBERTa's and XLM-RoBERTa's contextual embeddings of size 1024. Single layer MLPs to obtain $\boldsymbol{h}, \boldsymbol{m}$ vectors of size o (ignoring bias term) contain $1024o$ parameters. Biaffine layers (without bias) of input size i and output size o have i^2o parameters.

Accordingly, we use the following formula to determine the parameter count for Loc with 2 arc MLPs, 2 label MLPs, and 2 biaffine modules, one for the arcs and one for the labels:

$$
\begin{aligned}
&2 \times 1024x + 2 \times 1024y + x^2 + y^2\mathcal{L} \\
=&2048(x + y) + x^2 + y^2\mathcal{L}
\end{aligned}
$$

where x, y are the arc and label MLP output dimensions respectively and $\mathcal{L}$ is the number of labels in the dataset.

For ArcLoc, we use 2 single-layer MLPs for $\boldsymbol{h}, \boldsymbol{m}$ with output size d and one biaffine layer of input size d and output size r.

We also use 2 MLPs with a hidden layer to compute arc scores and labeling scores. These MLPs with input size r, hidden size $\frac{r}{2}$ for arcs and $2\mathcal{L}$ for labels, and output size either 1 for scores and $\mathcal{L}$ for labels respectively contain $r \times \frac{r}{2} + \frac{r}{2}$ and $r \times 2\mathcal{L} + 2\mathcal{L} \times \mathcal{L}$ parameters.

$$
\begin{aligned}
&2 \times 1024d + d^2r + r\frac{r}{2} + \frac{r}{2} + 2\mathcal{L} \times (r + \mathcal{L}) \\
=&2048d + d^2r + \frac{r}{2}(1 + r) + 2L(r + L)
\end{aligned}
$$

Additionally, each layer of Transformer adds (attention + MLP with hidden layer):

$$
r^2 + r \times (4r) + (4r) \times r = r^2 + 8r^2 = 9r^2
$$

CRF2o and VI require to add 3 single-layer MLPs with output size z and a triaffine layer for sibling scores with output size 1, on top of the Loc parameters:

$$
3072z + z^3
$$

B 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 (Loc, ArcLoc, CRF2o) after fine-tuning.

C UD Development Results

We report UD dev set results using gold POS in Table 4. In this case, we see that ArcLoc struggles to improve over Loc in the 4M regime, and that adding more allows parameters ArcLoc to recover the performance gap, while it has a detrimental effect on Loc. Adding transformer layers for arc representation refinement is useful in this setting, especially in big and large settings.

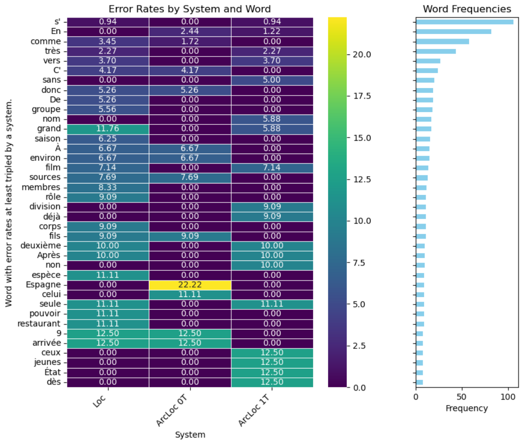

Figure 2: French error rates for words where one system has at least three times the error rate of another.

D Error Analysis: French and English UD Treebanks

This section provides a comparative analysis of the error rates across the French and English Universal Dependencies (UD) treebanks for the three parsing systems: Loc, ArcLoc 0T, and ArcLoc 1T. We analyze errors based on attachment distance, depth in the tree, part-of-speech (POS) tags, specific words, and dependency relations. The error trends and insights are discussed for both languages.

D.1 Error Rates for Words with Different Error Rates Across Systems

In this subsection, we analyze the words where one parsing system has error rates that are at least three times higher than another system. This comparison highlights significant performance differences between the systems when parsing certain words, emphasizing areas where certain models underperform.

Figure 2 shows the error rates for French words where one system has at least three times the error rate of another system. In the French dataset, words such as *Espagne* and *grand* exhibit large disparities between systems. For example, ArcLoc 0T struggles significantly more with the word *Espagne*, recording an error rate of 22.22%, whereas both Loc and ArcLoc 1T make no errors. Similarly, the word *grand* shows high error rates for Loc, with an error rate of 11.76%, while ArcLoc 0T and ArcLoc 1T have much lower error rates.

Figure 3 provides a similar comparison for the English dataset. Words like *form* and *Department*

	# Param (10^6)	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
VI	4	92.93	94.09	94.51	88.44	92.43	93.91	92.86	94.04	94.78	95.56	90.19	95.27	93.25
LOC	4	93.10	94.35	94.52	89.61	93.04	94.17	93.04	94.59	95.18	95.83	90.07	95.31	93.57
LOC	50	92.75	94.25	94.51	89.40	92.92	94.10	92.98	94.48	94.94	95.75	89.99	95.26	93.44
LOC	100	92.66	94.23	94.47	89.37	92.92	94.04	93.06	94.45	94.92	95.70	90.03	95.22	93.43
CRF2O	4	93.46	94.07	93.97	89.43	93.03	93.97	93.08	94.72	94.82	95.49	90.19	94.94	93.43
CRF2O	50	93.17	94.05	93.19	89.35	93.06	93.93	93.08	94.67	94.65	95.47	90.13	94.89	93.30
CRF2O	100	93.03	94.00	93.91	89.39	92.92	93.91	93.08	94.63	94.65	95.47	90.13	94.88	93.33
ARCLOC 0T	4	92.64	93.98	94.51	88.66	92.70	93.78	92.98	94.33	94.74	95.60	89.86	95.19	93.25
ARCLOC 0T	50	93.14	94.28	94.62	89.18	92.96	94.11	93.12	94.59	95.03	95.83	90.15	95.34	93.53
ARCLOC 0T	100	93.21	94.34	**94.65**	89.34	93.03	94.20	93.17	94.61	94.97	95.79	90.20	95.36	93.57
ARCLOC 1T	4	93.19	94.18	94.51	88.82	92.87	93.94	93.11	94.40	94.88	95.72	90.03	95.19	93.40
ARCLOC 1T	50	93.51	94.48	94.63	89.42	93.09	94.23	**93.23**	94.63	95.13	95.94	90.22	95.34	93.66
ARCLOC 1T	100	**93.67**	**94.51**	94.60	**89.49**	**93.15**	94.32	**93.23**	**94.79**	**95.14**	**95.99**	**90.30**	**95.38**	**93.71**
ARCLOC 2T	4	93.06	94.19	94.49	88.86	92.88	93.98	93.05	94.47	94.84	95.82	89.99	95.20	93.40
ARCLOC 2T	50	93.53	94.49	94.62	89.40	**93.15**	94.28	93.19	94.63	95.06	95.94	90.26	95.35	93.66
ARCLOC 2T	100	**93.67**	**94.51**	94.63	89.46	93.14	**94.36**	93.21	94.72	**95.14**	95.98	90.27	95.36	93.70

Table 4: Dev LAS for 12 languages in UD2.2 for different numbers of parameters per model and different numbers of layers for ARCLOC

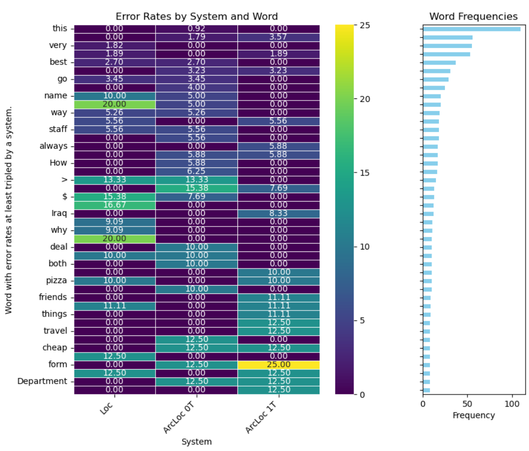

Figure 3: English error rates for words where one system has at least three times the error rate of another.

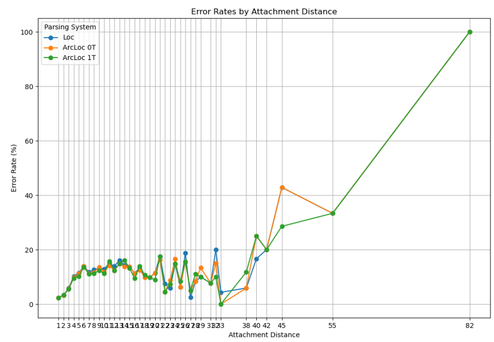

Figure 4: French error rates by attachment distance.

show stark differences in performance.

These discrepancies are likely due to challenges in handling certain lexical or syntactic constructions.

D.2 Error Rates by Attachment Distance

Figures 4 and 5 show the error rates as a function of attachment distance for French and English, respectively. For both languages, the systems perform well on short attachment distances (below 20), with error rates staying below 20%. However, as the attachment distance increases, the performance diverges. In French, ARCLOC 1T shows a steep increase in error rates beyond distance 30, while in English, ARCLOC 0T exhibits a sharp rise at distances above 40. These findings suggest that handling long-distance dependencies remains a challenge for all systems, particularly in French, where the errors rise more rapidly at shorter distances.

D.3 Error Rates by POS Tags

Figures 6 and 7 display the error rates across different POS tags for French and English. Both languages exhibit similar trends, with the highest error rates found for punctuation (PUNCT) and unknown symbols (X). For content words like nouns (NOUN) and verbs (VERB), the systems show relatively low error rates (below 10%). However, function words like pronouns (PRON), symbols (SYM), and conjunctions (CCONJ) are prone to higher error rates. The systems show higher sensitivity to these categories in English, particularly for SYM

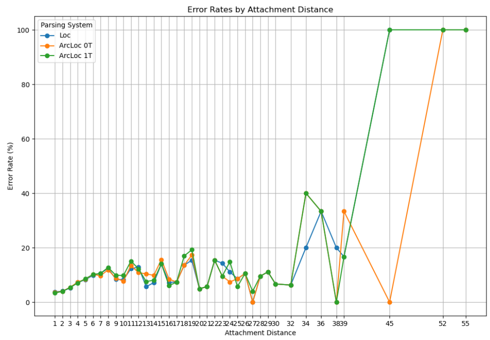

Figure 5: English error rates by attachment distance.

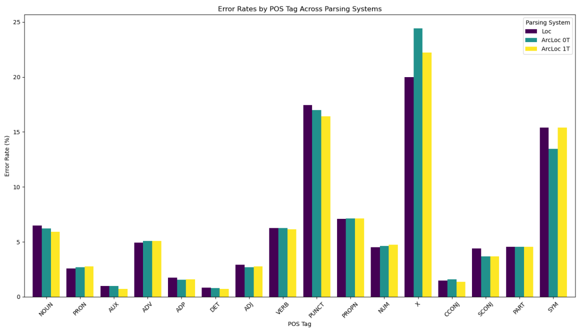

Figure 6: French error rates by POS tags.

and INTJ, where errors exceed 20%.

D.4 Error Rates by Depth in the Tree

Figures 8 and 9 present the error rates by depth of the dependent in the tree. For both languages, error rates are relatively low for shallow dependencies (depths 0 to 4). However, as depth increases, so do the error rates. In both French and English, LOC performs slightly worse at deeper levels, with error rates reaching up to 13.79% for depth 9 in French, and around 16% for depth 7 in English. In general, the deeper the dependency, the harder it is for all systems to maintain accuracy, with ARCLOC 0T performing somewhat better at deeper levels in English compared to French.

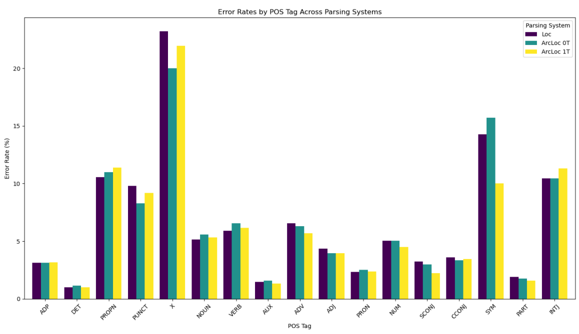

Figure 7: English error rates by POS tags.

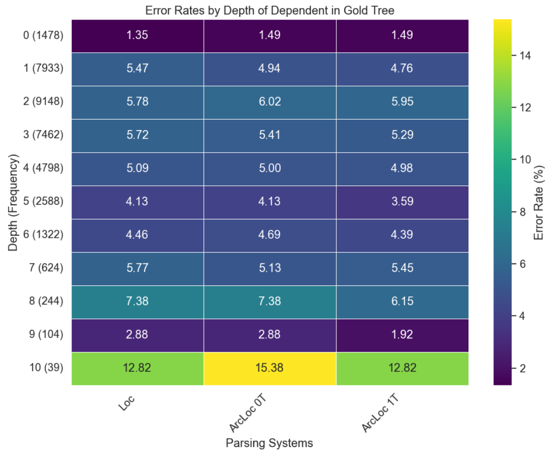

Figure 8: French error rates by depth of dependent in the tree.

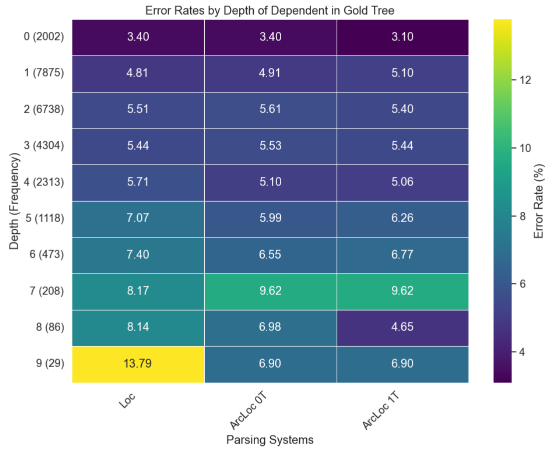

Figure 9: English error rates by depth of dependent in the tree.

D.5 Error Rates by Dependency Relations

Figures 10 and 11 present heatmaps of error rates across different dependency relations for French and English. In both languages, complex relations like `parataxis-root` and `nmod:obl` exhibit the highest error rates. While ARCLOC 0T shows higher errors for French in these challenging relations, it performs better on average for English, especially in long-distance relations such as `flat:foreign-compound` and `fixed-case`. This indicates that while certain syntactic structures are universally challenging, language-specific factors also contribute to system performance differences.

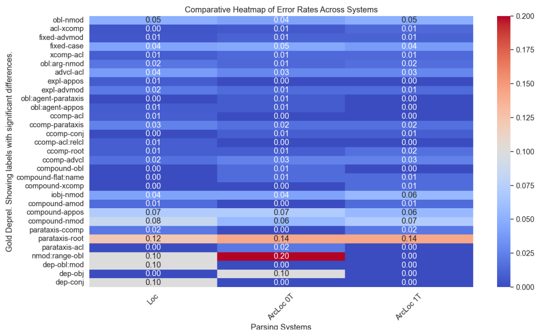

Figure 10: French heatmap of error rates by dependency relations.

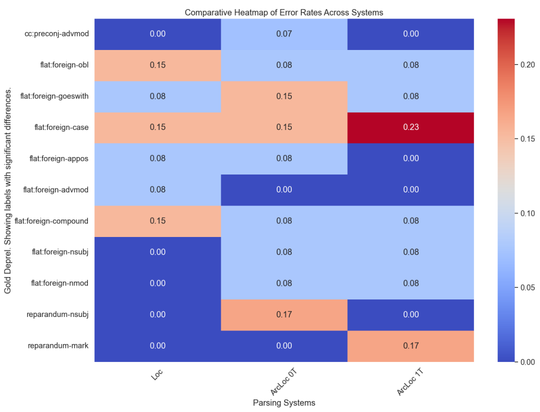

Figure 11: English heatmap of error rates by dependency relations.

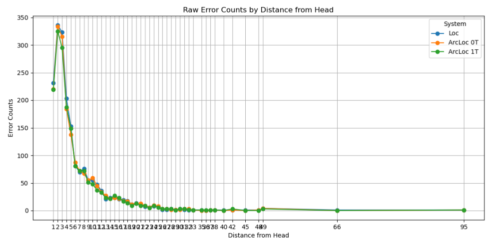

Figure 12: French raw error counts by distance from head.

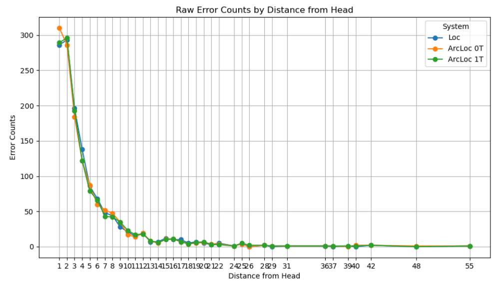

Figure 13: English raw error counts by distance from head.

D.6 Raw Error Counts by Distance from Head

Figures 12 and 13 present the raw error counts as a function of distance from the head. For both languages, the majority of errors occur at short distances (1 to 5 words), where dependency relations are the most frequent. The error count decreases as the distance increases, but significant spikes in errors occur beyond distance 30, particularly in French. This confirms that handling long-range dependencies remains a common challenge across both languages and all parsing systems.