Parsing linearizations appreciate PoS tags - but some are fussy about errors

Alberto Muñoz-Ortiz[1], **Mark Anderson**[2], **David Vilares**[1], **Carlos Gómez-Rodríguez**[1]
[1]Universidade da Coruña, CITIC, Spain
[2] PIN Caerdydd, Prifysgol Caerdydd, United Kingdom
alberto.munoz.ortiz@udc.es, andersonm8@caerdydd.ac.uk,
david.vilares@udc.es, carlos.gomez@udc.es

Abstract

PoS tags, once taken for granted as a useful resource for syntactic parsing, have become more situational with the popularization of deep learning. Recent work on the impact of PoS tags on graph- and transition-based parsers suggests that they are only useful when tagging accuracy is prohibitively high, or in low-resource scenarios. However, such an analysis is lacking for the emerging sequence labeling parsing paradigm, where it is especially relevant as some models explicitly use PoS tags for encoding and decoding. We undertake a study and uncover some trends. Among them, PoS tags are generally more useful for sequence labeling parsers than for other paradigms, but the impact of their accuracy is highly encoding-dependent, with the PoS-based head-selection encoding being best only when both tagging accuracy and resource availability are high.

1 Introduction

PoS tags have long been considered a useful feature for parsers, especially prior to the prevalence of neural networks (Voutilainen, 1998; Dalrymple, 2006; Alfared and Béchet, 2012). For neural parsers, it is less clear if they are useful or not. Work has shown that when using word and character embeddings, PoS tags become much less useful (Ballesteros et al., 2015; de Lhoneux et al., 2017). However, Dozat et al. (2017) found using universal PoS (UPoS) tags to be somewhat helpful, but improvements are typically quite small (Smith et al., 2018). Similarly, for multi-task systems, small improvements have been observed for both UPoS and finer-grained tags (Zhang et al., 2020).

A limiting factor when using predicted PoS tags is the apparent need for very high accuracy from taggers (Anderson and Gómez-Rodríguez, 2020). This is particularly problematic in a low-resource setting where using gold tags gives unreasonably high performance (Tiedemann, 2015) and high accuracy taggers are difficult to obtain (Kann et al., 2020). However, some work has suggested that in a low-resource setting even low accuracy taggers can be beneficial for parsing performance, especially when there is more PoS tag annotations than dependency tree annotations (Anderson et al., 2021).

These findings relate to transition-based (TB) and graph-based (GB) parsers, but recently several encodings have been proposed to frame dependency parsing as a sequence labeling task (Strzyz et al., 2019; Lacroix, 2019; Gómez-Rodríguez et al., 2020), providing an alternative to GB and TB models when efficiency is a priority (Anderson and Gómez-Rodríguez, 2021). Muñoz-Ortiz et al. (2021) found that the amount of data required for different encodings varied and that some were impacted by predicted PoS tag use more than others.

Here, we evaluate the impact of PoS tagging accuracy on different encodings and also the interplay of this potential relation and the amount of available data (using low-, mid-, high-, and very-high-resource treebanks). This is done by artificially controlling the accuracy of PoS taggers by using the nature of errors generated by robust taggers.[1]

2 Sequence labeling parsing

In dependency parsing as sequence labeling, the goal is to assign a single label of the form (x_i, l_i) to every input token w_i of a sequence, where x_i encodes a subset of the arcs related to w_i and l_i is the dependency type. Below, we review the existing families of linearizations used in this work.

Head-selection (Spoustová and Spousta, 2010), where x_i encodes the head of w_i using an absolute index or a relative offset, that can be based on some word property (usually PoS tags, which is also the property we use in this work due to its strong performance in previous work). So for instance, if x_i = (+n, X), this would indicate that the head of w_i is the nth word to the right of w_i with the word

[1]All source code available at https://www.grupolys.org/software/aacl2022/.

property X. Some desirable properties of this encoding family are a direct correspondence between words and arcs and the capacity to encode any non-projective tree. However, a major weakness is its dependency on the chosen property (in our case, PoS tags) to decode trees.

Bracketing-based x_i represents the dependency arcs using a string of brackets, with each arc represented by a bracket pair. Its main advantage is that it is independent of external features, but regarding projectivity it cannot represent arcs that cross in the same direction. To alleviate this, we use the encoding proposed by Strzyz et al. (2020), that adds a second independent plane of brackets ($2\mathrm{p}^{\mathrm{b}}$), inspired by multiplanarity (Yli-Jyrä, 2003).

Transition-based (Gómez-Rodríguez et al., 2020), where given a sequence of transitions generated by a left-to-right transition-based parser, it splits it in labels based on read transitions (e.g. SHIFT), such that each word receives a label x_i with a subset of transition actions. For this work, we consider mappings from a projective algorithm, arc-hybrid ($\mathrm{ah}^{\mathrm{tb}}$; Kuhlmann et al., 2011) and a non projective algorithm, Covington (c^{tb}; Covington, 2001).

2.1 Parser systems

We use a 2-layer bidirectional long short-term memory (biLSTM) network with a feed-forward network to predict the labels using softmaxes. We use hard-sharing multi-task learning to predict x_i and l_i.[2] The inputs to the network are randomly initialized word embeddings and LSTM character embeddings and *optionally* (see §4), PoS tag embeddings. The appendix specifies the hyperparameters. For a homogeneous comparison against work on the usefulness of PoS tags for transition and graph-based models, and focused on efficiency, we do not use large language models.

3 Controlling PoS tag accuracy

We purposefully change the accuracy of the PoS tags in a treebank, effectively treating this accuracy as the independent variable in a controlled experiment and LAS as the dependent variable, i.e. $LAS = f(Acc_{PoS})$ where f is some function. Rather than randomly altering the gold label of PoS tags, we alter them based on the actual errors that PoS taggers make for a given treebank. This means PoS tags that are more likely to be incorrect for a given treebank will be more likely to be altered when changing the overall PoS accuracy of that treebank. We refer to this as the *error rate* for PoS tags. The incorrect label is also based on the most likely incorrect label for the PoS tag error for that treebank based on the incorrect labeling from the tagger. We refer to this as the *error type*, e.g. NOUN→VERB.

We trained BiLSTM taggers for each of the treebanks to get the error rates for each PoS tag type and rate of each error type for each tag. Their generally high performances, even for the smaller treebanks, are shown in Table 5 in the Appendix.

From the errors of these taggers, we first need the estimated probability that a given PoS tag t is tagged erroneously:

$$p(error|t) = \frac{E_t}{C_t} \tag{1}$$

where E_t is the error count for tag t and C_t is the total count for tag t. Then we need the probability of applying an erroneous tag e to a ground-truth tag t:

$$p(e|t, error) = \frac{E_{t\to e}}{E_t} \tag{2}$$

where $E_{t\to e}$ is the error count when labeling t as e. This estimated probability remains fixed, whereas $p(error|t)$ is adjusted to vary the overall accuracy.

We adjust these values by applying a weight, γ:

$$\gamma = \frac{E_A}{E} \tag{3}$$

where E is the global error count and E_A is the adjusted global error count such that the resulting tagging error is A. $p(error|t)$ is then adjusted:

$$p(error|t) = \frac{\gamma E_t}{C_t} \tag{4}$$

It is possible that $\gamma E_t > C_t$. When this occurs to tag t we cap γE_t at C_t and then recalculate γ, removing the counts associated with this tag:

$$\gamma = \frac{E_A - C_t}{E - C_t} \tag{5}$$

This is then done iteratively for each tag where $E_t \geq C_t$ until we obtain an error count for each tag such that the total error count reaches E_A.

These are all derived and applied as such to the test set of treebanks as this is where we evaluate the impact of PoS tag errors. To further echo the erroneous nature of these taggers, when $E_A \leq$

[2]We use a 2-task setup for all encodings, except $2\mathrm{p}^{\mathrm{b}}$ for which we use 3 tasks, as each plane is predicted independently.

	Treebank	Family	# Trees	# Tokens
Low	Skolt Sami$_{\text{Giellagas}}$	Uralic (Sami)	200	2 461
	Guajajara$_{\text{TuDeT}}$	Tupian (Tupi-Guarani)	284	2 052
	Ligurian$_{\text{GLT}}$	IE (Romance)	316	6 928
	Bhojpuri$_{\text{BHTB}}$	IE (Indic)	357	6 665
Mid	Kiche$_{\text{IU}}$	Mayan	1 435	10 013
	Welsh$_{\text{CCG}}$	IE (Celtic)	2 111	41 208
	Armenian$_{\text{ArmTDP}}$	IE (Armenian)	2 502	52 630
	Vietnamese$_{\text{VTB}}$	Austro-Asiatic (Viet-Muong)	3 000	43 754
High	Basque$_{\text{BDT}}$	Basque	8 993	121 443
	Turkish$_{\text{BOUN}}$	Turkic (Southwestern)	9 761	122 383
	Bulgarian$_{\text{BTB}}$	IE (Slavic)	11 138	146 159
	Ancient Greek$_{\text{Perseus}}$	IE (Greek)	13 919	202 989
V. High	Norwegian$_{\text{Bokmål}}$	IE (Germanic)	20 044	310 221
	Korean$_{\text{Kaist}}$	Korean	27 363	350 090
	Persian$_{\text{PerDT}}$	IE (Iranian)	29 107	501 776
	Estonian$_{\text{EDT}}$	Uralic (Finnic)	30 972	437 769

Table 1: Details of the treebanks used in this work.

E only the subset of real errors are used when generating errors. When $E_A > E$ this subset of real errors is maintained and subtracted such that:

$$p(error|t) = \frac{(\gamma - 1)E_t}{C_t - E_t} \tag{6}$$

and this is only applied on the tokens which were not erroneously tagged by the taggers.

For every eligible token, based on its tag t an error is generated based on $p(error|t)$ and if an error is to be generated, the erroneous tag is selected based on the distribution over $p(e|t, error)$.

This is also applied to the training and dev set as it seems better to use predicted tags when training (Anderson and Gómez-Rodríguez, 2020). There are differences in the distribution of PoS tags and as the algorithm is based on the test data, at times it isn't possible to get exactly E_A. We therefore allow a small variation of $\pm$0.05 on E_A.

We then selected a set of PoS tag accuracies to test a range of values (75, 80, 85, 95, 97.5, 100). We included the 97.5% accuracy to evaluate the findings of Anderson and Gómez-Rodríguez (2020), where they observed a severe increase in performance between high scoring taggers and gold tags, otherwise we use increments of 5%.

4 Experiments

We now present the experimental setup to determine how parsing scores evolve for the chosen linearizations when the tagging accuracy degrades. As evaluation metrics, we use Labeled (LAS) and Unlabeled Attachment Scores (UAS).

Data Treebanks from Table 1 were selected using a number of criteria. We chose treebanks that were all from different language families and therefore

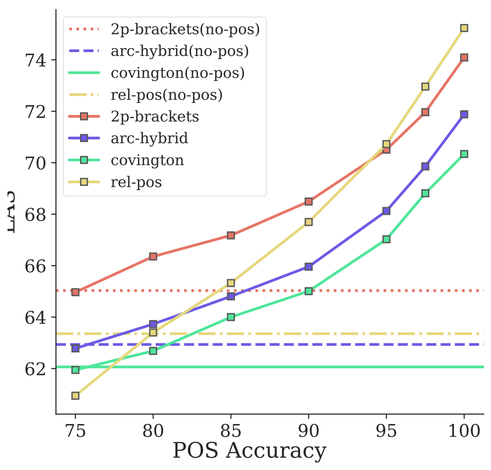


Figure 1: Average LAS across all treebanks against PoS tagging accuracies for different linearizations, compared to the no-tags baselines.

o

exhibit a range of linguistic behaviors. We also selected treebanks such that we used 4 low-resource, 4 mid-resource, 4 high-resource and 4 very high-resource treebanks. Within each of those categories, we also selected treebanks with slightly different amounts of data, so as to obtain an incremental range of treebank sizes across low, mid, high and very high boundaries. Moreover, we ensured the quality of the treebanks by selecting treebanks that were either manually annotated in the UD framework or manually checked after automatic conversions. When a treebank did not contain a development set, we re-split the data by collecting the data across the training and test data and split the full data such that 60% was allocated to the training set, 10% to the development, and 30% to the test.

Setup We train and test parsers on sets of predicted tags, as explained in §3. We consider two baselines: (i) parsers trained without PoS tags[3] (`base-no-tags`), (ii) parsers trained with gold tags on a multi-task setup (`base-mtl`).

4.1 Results

Table 2 shows the average LAS scores across all treebank setups for all encodings and tagging accuracies, together with both baselines. To better interpret the results and tendencies, we will also visualize the results in different figures.[4] Note that we don't include `base-mtl` as they performed very similar to `base-no-tags`. We include the

[3]Forced setup for rp^{h}, as PoS tags are needed to decode.
[4]UAS results are shown in Figures 3 and 4 in the Appendix.

Setup	Low-resource				Mid-resource				High-resource				V. high-resource				All			
	$2p^{b}$	ah^{tb}	c^{tb}	rp^{h}	$2p^{b}$	ah^{tb}	c^{tb}	rp^{h}	$2p^{b}$	ah^{tb}	c^{tb}	rp^{h}	$2p^{b}$	ah^{tb}	c^{tb}	rp^{h}	$2p^{b}$	ah^{tb}	c^{tb}	rp^{h}
75	**50.65**	49.33	48.43	47.72	**63.26**	60.18	60.23	58.64	**66.34**	64.18	63.87	64.09	**79.63**	77.44	75.26	73.32	**64.97**	62.78	61.98	60.94
80	**53.84**	50.58	48.78	50.94	**64.00**	61.52	61.34	60.87	**67.53**	64.88	64.88	64.70	**80.06**	77.93	75.74	77.09	**66.36**	63.73	62.69	63.40
85	**54.17**	52.48	51.27	52.62	**65.25**	62.34	62.06	63.36	**68.11**	65.38	65.33	66.56	**81.18**	79.02	77.34	78.76	**67.18**	64.81	64.00	65.32
90	**56.03**	53.55	52.78	55.34	**67.30**	64.05	63.35	66.18	69.31	66.86	66.61	**69.47**	**81.33**	79.39	77.05	79.80	**68.49**	65.96	65.01	67.70
95	**59.30**	56.88	55.75	58.90	69.84	67.34	66.20	**70.30**	70.28	67.66	67.32	**71.18**	**82.61**	80.62	78.83	82.52	70.51	68.12	67.02	**70.72**
97.5	60.00	58.70	57.59	**61.86**	72.63	69.47	68.99	**72.84**	71.59	69.27	68.39	**72.83**	83.91	82.00	80.27	**84.31**	71.96	69.86	68.81	**72.96**
100	62.16	60.97	58.64	**64.23**	74.28	71.19	70.02	**75.20**	73.40	70.60	70.05	**74.50**	86.52	84.77	82.65	**87.20**	74.09	71.88	70.34	**75.24**
MTL	47.78	46.83	45.60	**48.08**	**64.15**	62.15	60.68	63.17	**67.97**	64.94	65.26	67.47	**81.52**	79.46	76.85	80.95	**65.35**	63.34	62.10	64.92
No PoS tags	47.36	46.18	45.79	**49.26**	**63.94**	61.58	60.73	57.52	**67.67**	64.76	64.75	66.58	**81.15**	79.22	76.98	80.06	**65.03**	62.94	62.06	63.35

Table 2: Average LAS for different setups and PoS tag accuracies for the groups of treebanks studied.

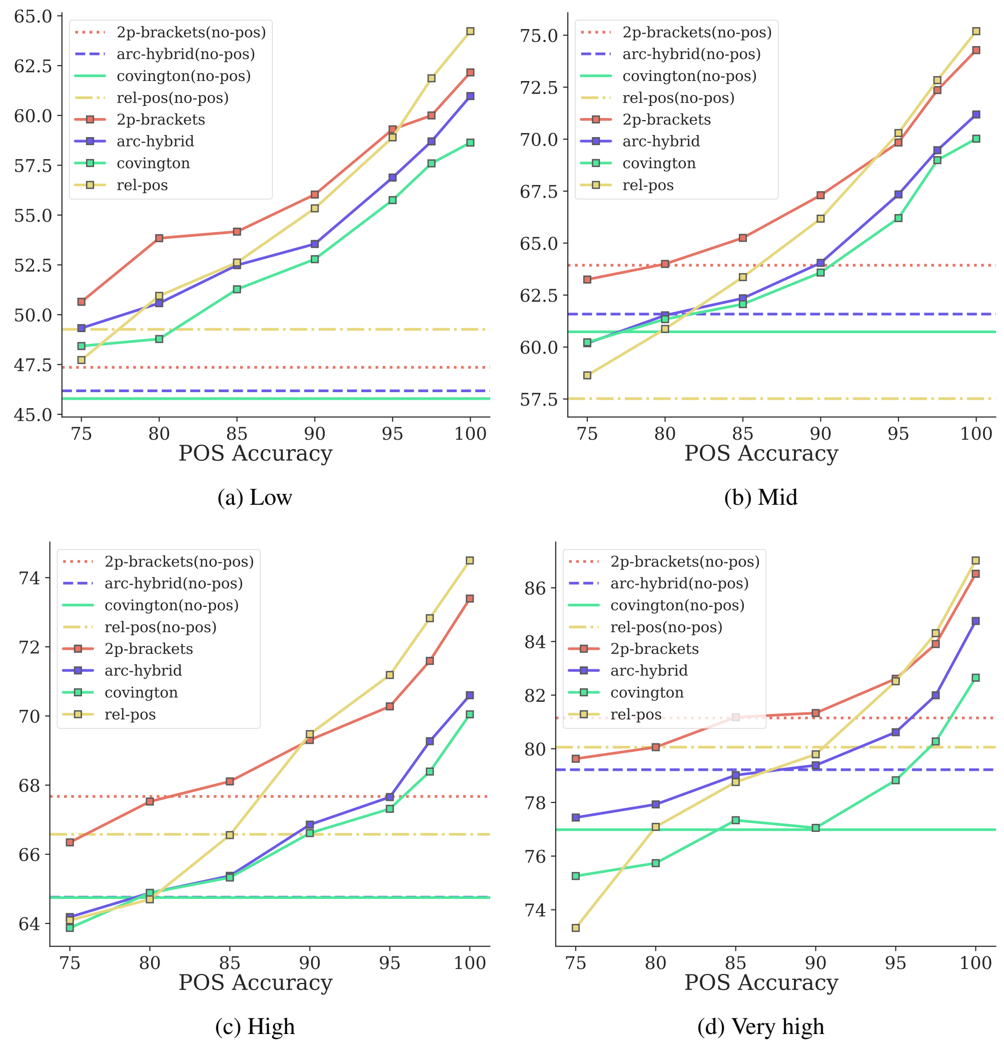


Figure 2: Average LAS for the (a) low-, (b) mid-, (c) high- and (d) very high-resource subsets of treebanks for different PoS tagging accuracies and linearizations, compared to the no-tags baselines.

results with a state-of-the-art graph based parser (Dozat et al., 2017) in Table 3 for comparison.

All treebanks Figure 1 shows the average LAS across all treebanks for the different linearizations, using PoS tags or not. The results suggest that even using low accuracy tags is better than not using them. In detail, rp^{h} is the linearization that is affected the most by the quality of the PoS tags, as it relies directly on them in order to decode the tree, degrading from the 1st position when using gold tags to the last one when tags have an accuracy of 75%. On the other hand, $2p^{b}$ seems to be the most useful encoding for real-world situations, outperforming the other linearizations when no tags or tags with an accuracy under 95% are used, and performing on par with rp^{h} over that mark. Note

Setup	Low	Mid	High	V. High	All
75	55.61	69.79	76.66	86.00	72.01
80	56.60	70.17	76.49	85.95	72.30
85	59.12	70.76	76.90	86.33	73.28
90	60.40	71.61	77.69	86.62	74.08
95	62.12	74.63	78.22	87.13	75.52
97.5	65.05	76.42	79.44	88.16	77.27
100	66.65	78.52	80.96	90.74	79.22
No PoS tags	58.40	71.71	77.66	87.72	73.74

Table 3: Average LAS for different setups and PoS tag accuracies for the groups of treebanks studied using the graph-based parser.

that while Strzyz et al. (2019) chose rp^{h} as their best model for evaluation, the choice was biased by using English, a language with atypically high tagging accuracy.

Results for different resourced sets of treebanks Figure 2 shows the results for the low-resource, mid-resource, high-resource and very high-resource treebanks, respectively. Interestingly, we observe trends regarding the *cutoff points* (the points where a model surpasses another), depending on the quality of PoS tags and quantity of available data. In particular, the cutoff points between the parsers that use PoS tags and the base-no-tags models are found at higher tagging accuracies when the data resources are larger too. Also, the cutoff point between rp^{h} and 2p^{b} is at a lower PoS tagging accuracy when we have more data, although the results for the very high-resource treebanks break this trend. Finally, the low performance of the transition-based encodings is more pronounced for high-resource treebanks, with the exception the ah^{tb} for the very high-resource treebanks.

5 Discussion

The obtained results offer some valuable information about how PoS tag quality affects performance for different encodings and quantities of data. In most situations using PoS tags as features is better than not using them, in contrast with results for other parser architectures as described above.

In addition, the less resources, the harder it is for rp^{h} to beat brackets: cutoffs are at 97.5%, 95%, 90% for low-, mid-, and high-resource treebanks, respectively. However, for very high-resource treebanks, the cutoff is back at 95%. Compounded with the low tagging accuracy expected in low-resource setups, this highlights that rp^{h} is less suited for them. 2p^{b}, which generally outperforms the other encodings below 90% tagging accuracy, is the best low-resource option.

The more resources available, the harder it is for the models using PoS tags to outperform base-no-tags, both for bracketing- and transition-based linearizations; i.e. experiments suggest that the benefits provided by the PoS tags decline when more training data is available. For brackets, the cutoffs occur at <75%, 80%, 85% and 90% for the low-, mid-, high- and very high-resource set, and for transition encodings, they are at <75% for the low-resource set and at ∼80% for mid- and high-resource sets. For the very-high resource set, cutoff points are at 85% for c^{tb} and 90% for ah^{tb}.

6 Conclusion

We connected the impact that the quality of PoS tags and quantity of available data has on several dependency parsing linearizations. We tested this by controlling PoS tagging performance on a range of UD treebanks, diverse in terms of both amount of resources and typology. The results showed that for sequence labeling parsing, which prioritizes efficiency, PoS tags are still welcome, contrary to more mature parsing paradigms such as transition-based and graph-based ones. The experiments also showed that parsing linearizations benefit from PoS tagging accuracy differently, and in particular linearizations that represent arcs as bracket strings are a better choice for most realistic scenarios.

Aknowledgements

Mark was supported by a UKRI Future Leaders Fellowship (MR/T042001/1). This paper has received funding from ERDF/MICINN-AEI (SCANNER-UDC, PID2020-113230RB-C21), Xunta de Galicia (ED431C 2020/11), and Centro de Investigación de Galicia "CITIC", funded by Xunta de Galicia and the European Union (ERDF - Galicia 2014-2020 Program), by grant ED431G 2019/01

References

Ramadan Alfared and Denis Béchet. 2012. POS taggers and dependency parsing. *International Journal of Computational Linguistics and Applications*, 3(2):107–122.

Mark Anderson, Mathieu Dehouck, and Carlos Gómez-Rodríguez. 2021. A falta de pan, buenas son tortas: The efficacy of predicted UPOS tags for low resource UD 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 78–83, Online. Association for Computational Linguistics.

Mark Anderson and Carlos Gómez-Rodríguez. 2020. On the frailty of universal POS tags for neural UD parsers. In *Proceedings of the 24th Conference on Computational Natural Language Learning*, pages 69–96, Online. Association for Computational Linguistics.

Mark Anderson and Carlos Gómez-Rodríguez. 2021. A modest Pareto optimisation analysis of dependency parsers in 2021. 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 119–130, Online. Association for Computational Linguistics.

Miguel Ballesteros, Chris Dyer, and Noah A Smith. 2015. Improved transition-based parsing by modeling characters instead of words with LSTMs. *arXiv preprint arXiv:1508.00657*.

Michael A. Covington. 2001. A fundamental algorithm for dependency parsing. In *Proceedings of the 39th annual ACM southeast conference*, volume 1. Citeseer.

Mary Dalrymple. 2006. How much can part-of-speech tagging help parsing? *Natural Language Engineering*, 12(4):373–389.

Miryam de Lhoneux, Yan Shao, Ali Basirat, Eliyahu Kiperwasser, Sara Stymne, Yoav Goldberg, and Joakim Nivre. 2017. From raw text to universal dependencies-look, no tags! In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 207–217.

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

Carlos Gómez-Rodríguez, Michalina Strzyz, and David Vilares. 2020. A unifying theory of transition-based and sequence labeling parsing. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 3776–3793, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Katharina Kann, Ophélie Lacroix, and Anders Søgaard. 2020. Weakly supervised POS taggers perform poorly on truly low-resource languages. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(5):8066–8073.

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

Ophélie Lacroix. 2019. Dependency parsing as sequence labeling with head-based encoding and multitask learning. In *Proceedings of the Fifth International Conference on Dependency Linguistics (Depling, SyntaxFest 2019)*, pages 136–143, Paris, France. Association for Computational Linguistics.

Alberto Muñoz-Ortiz, Michalina Strzyz, and David Vilares. 2021. Not all linearizations are equally data-hungry in sequence labeling parsing. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2021)*, pages 978–988, Held Online. INCOMA Ltd.

Aaron Smith, Miryam de Lhoneux, Sara Stymne, and Joakim Nivre. 2018. An investigation of the interactions between pre-trained word embeddings, character models and POS tags in dependency parsing. *arXiv preprint arXiv:1808.09060*.

Drahomíra Johanka Spoustová and Miroslav Spousta. 2010. Dependency parsing as a sequence labeling task. *The Prague Bulletin of Mathematical Linguistics*, 94(2010):7–14.

Michalina Strzyz, David Vilares, and Carlos Gómez-Rodríguez. 2019. Viable dependency parsing as sequence labeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 717–723, Minneapolis, Minnesota. Association for Computational Linguistics.

Michalina Strzyz, David Vilares, and Carlos Gómez-Rodríguez. 2020. Bracketing encodings for 2-planar dependency parsing. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2472–2484, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Jörg Tiedemann. 2015. Cross-lingual dependency parsing with universal dependencies and predicted pos labels. *Proceedings of the Third International Conference on Dependency Linguistics (Depling 2015)*, pages 340–349.

Atro Voutilainen. 1998. Does tagging help parsing?: a case study on finite state parsing. In *Proceedings of the International Workshop on Finite State Methods in Natural Language Processing*, pages 25–36. Association for Computational Linguistics.

Anssi Mikael Yli-Jyrä. 2003. Multiplanarity-a model for dependency structures in treebanks. In *TLT 2003, Proceedings of the Second Workshop on Treebanks and Linguistic Theories*. Växjö University Press.

Yu Zhang, Zhenghua Li, Houquan Zhou, and Min Zhang. 2020. Is POS tagging necessary or even helpful for neural dependency parsing? *arXiv preprint arXiv:2003.03204*.

A PoS tagging details

Table 4 details the hyperparameters used to train the taggers in this work.

Hyperparameter	Value
Word embedding dimensions	100
Character embedding in	32
Character embedding out	100
Embedding dropout	0.33
biLSTM layers	3
biLSTM nodes	400
biLSTM dropout	0.33
MLP dimensions	512
MLP layers	1
Epochs	200
Patience	10
training batch size	32
learning rate	0.002
β_1, β_2	0.9, 0.9
ϵ	1×10^{-12}
decay	0.75

Table 4: Hyperparameters used for the taggers.

Meanwhile, Table 5 shows the performance of the taggers that we initially used to draw the error distributions and propose PoS tags with different levels of accuracy.

	Tagger Accuracy
Ancient Greek-Perseus	90.14
Armenian-ArmTDP	92.22
Basque-BDT	94.74
Bhojpuri-BHTB	81.52
Bulgarian-BTB	98.26
Estonian-EDT	96.32
Guajajara-TuDeT	84.20
Kiche-IU	92.28
Korean-Kaist	94.34
Ligurian-GLT	81.19
Norwegian-Bokmål	97.51
Persian-PerDT	96.53
Skolt Sami-Giellagas	80.03
Turkish-BOUN	91.31
Vietnamese-VTB	87.05
Welsh-CCG	91.76

Table 5: Accuracy on test sets of biLSTM taggers trained for each treebank from which each error distribution was deduced and used to control accuracy for each treebank in experiments.

B Parsing hyperparameters

Table 6 details the hyperparameters used to train all the sequence labeling parsers evaluated in this work.

Hyperparameter	Value
Word embedding dimensions	100
Character embedding dimensions	30
Character hidden dimensions	50
Hidden dimensions	800
POS embedding dimension	25
LSTM layers	2
CNN laters	4
Dropout	0.5
Epochs	50
training batch size	8
learning rate	0.02
momentum	0.9
decay	0.05

Table 6: Hyperparameters used for the sequence labeling parsers.

C Additional results

Figures 3 and 4 shows the UAS results complementing the LAS results reported in §4 (in Figures 1 and 2, respectively). Figures from 5 to 20 show the LAS results for each treebank.

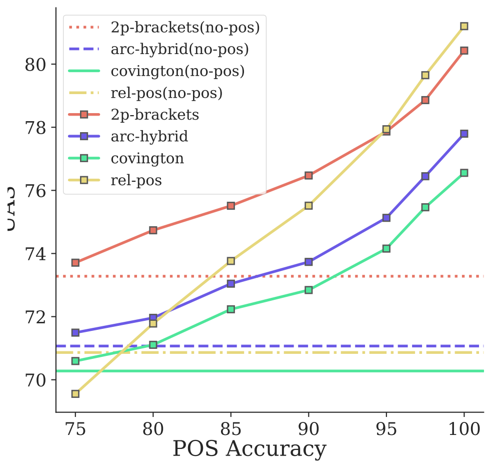
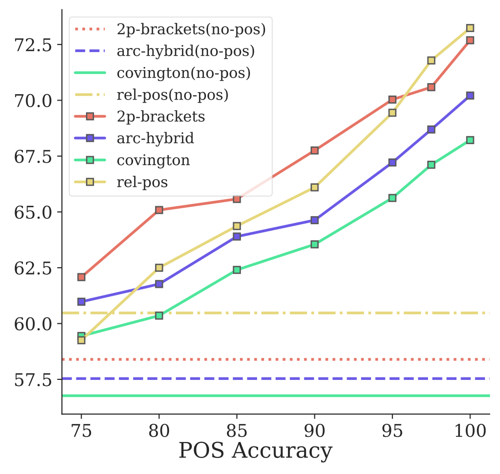
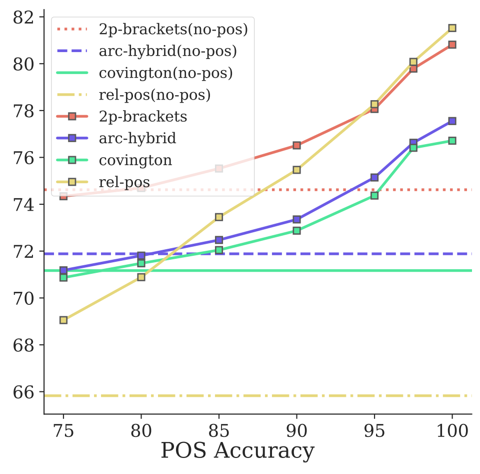


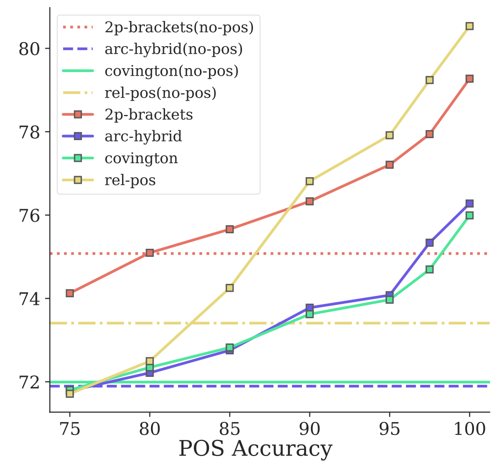
Figure 3: Average UAS across all treebanks against PoS tagging accuracies for different linearizations, compared to the no-tags baselines.



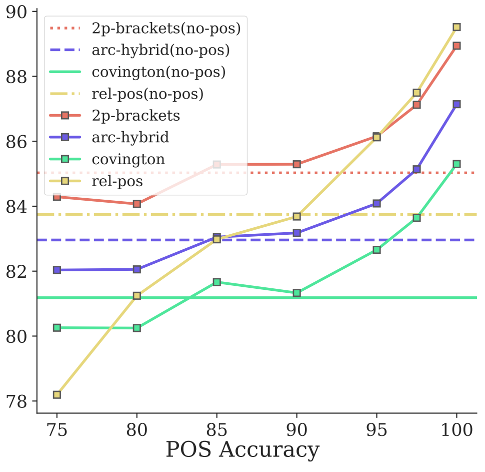
(a) Low



(b) Mid



(c) High



(d) Very high

Figure 4: Average UAS for the (a) low-, (b) mid-, (c) high and (d) very-high-resource subsets of treebanks for different PoS tagging accuracies and linearizations, compared to the no-tags baselines.

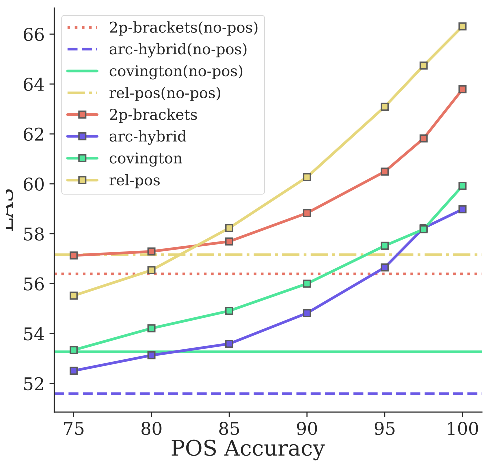


Figure 5: LAS against PoS tagging accuracies for different linearizations for the Ancient Greek$_{\text{Perseus}}$, compared to the no-tags baselines.

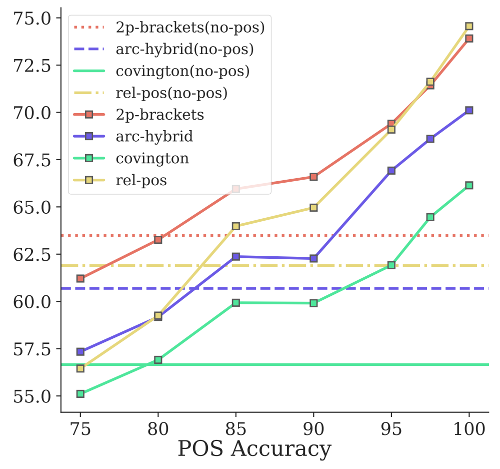


Figure 6: LAS against PoS tagging accuracies for different linearizations for the Armenian$_{\text{ArmTDP}}$, compared to the no-tags baselines.

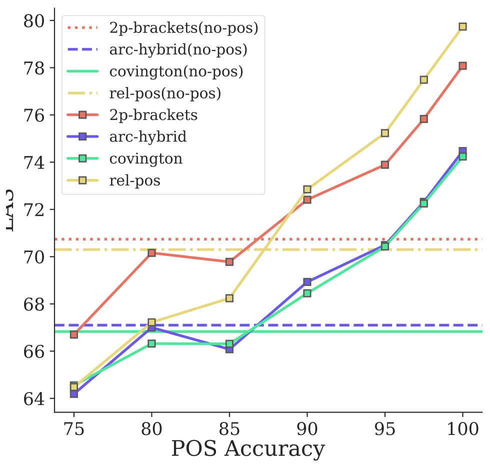


Figure 7: LAS against PoS tagging accuracies for different linearizations for the Basque$_{\text{BDT}}$, compared to the no-tags baselines.

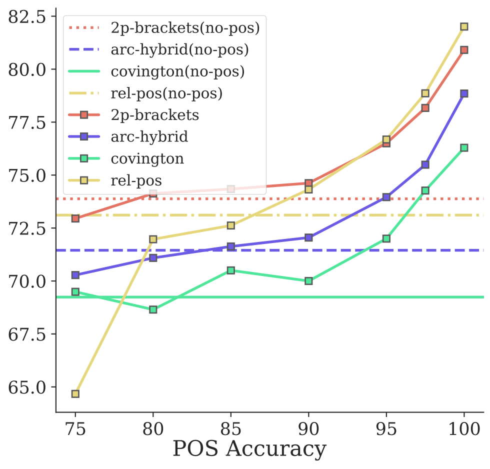


Figure 10: LAS against PoS tagging accuracies for different linearizations for the Estonian$_{\text{EDT}}$, compared to the no-tags baselines.

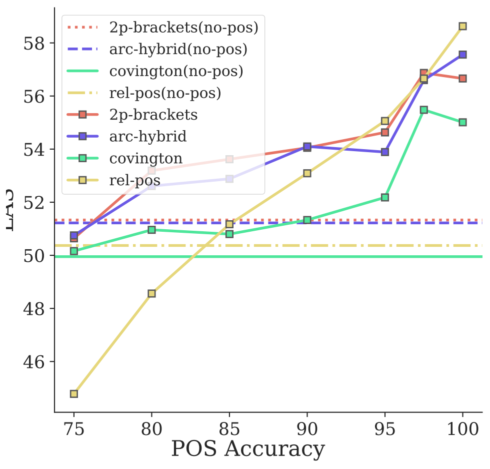


Figure 8: LAS against PoS tagging accuracies for different linearizations for the Bhojpuri$_{\text{BHTB}}$, compared to the no-tags baselines.

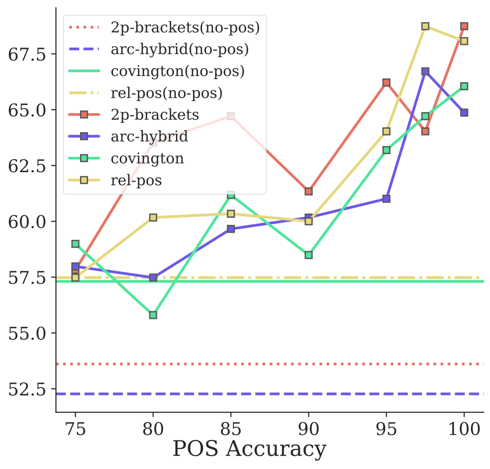


Figure 11: LAS against PoS tagging accuracies for different linearizations for the Guajajara$_{\text{TuDeT}}$, compared to the no-tags baselines.

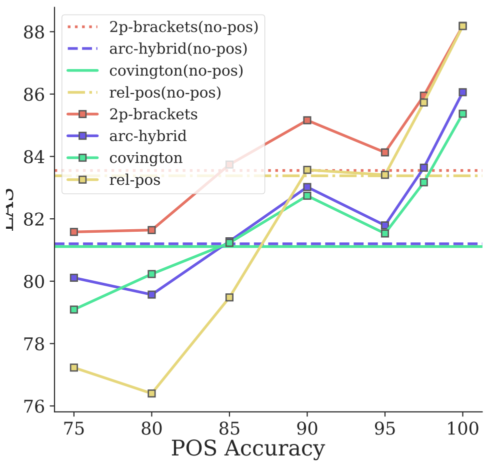


Figure 9: LAS against PoS tagging accuracies for different linearizations for the Bulgarian$_{\text{BTB}}$, compared to the no-tags baselines.

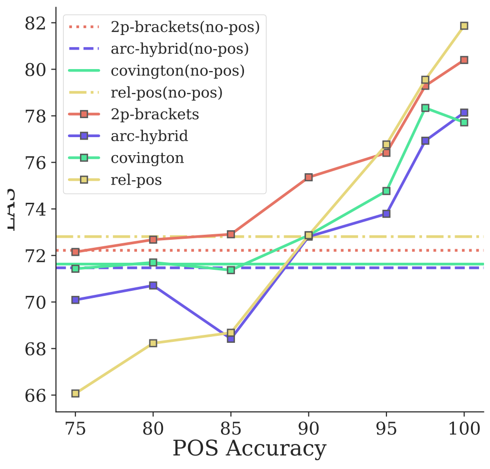


Figure 12: LAS against PoS tagging accuracies for different linearizations for the Kiche$_{\text{IU}}$, compared to the no-tags baselines.

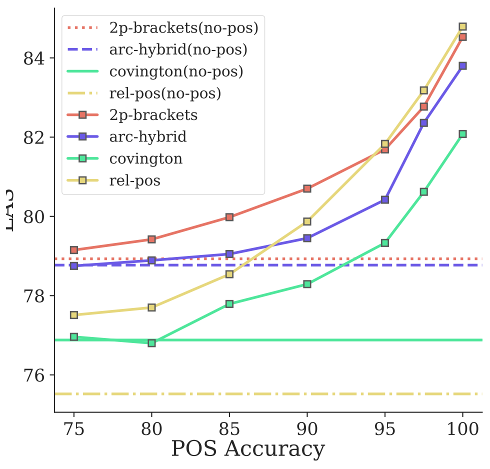


Figure 13: LAS against PoS tagging accuracies for different linearizations for the Korean$_{\text{Kaist}}$, compared to the no-tags baselines.

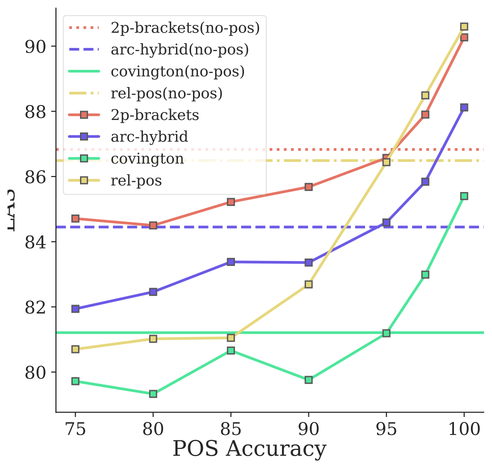


Figure 16: LAS against PoS tagging accuracies for different linearizations for the Persian$_{\text{PerDT}}$, compared to the no-tags baselines.

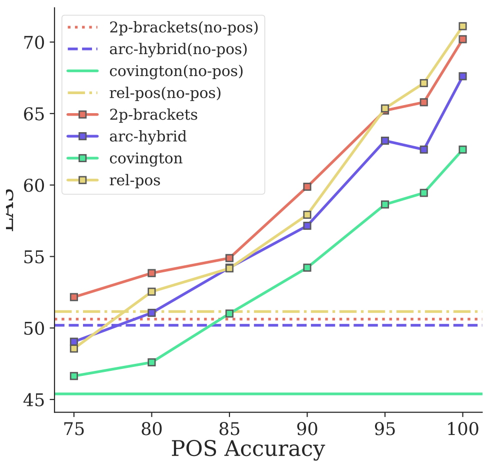


Figure 14: LAS against PoS tagging accuracies for different linearizations for the Ligurian$_{\text{GLT}}$, compared to the no-tags baselines.

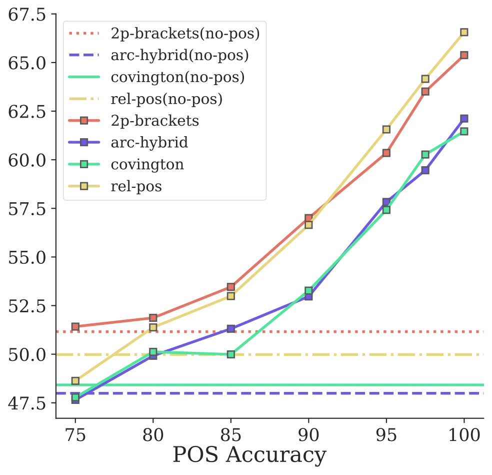


Figure 17: LAS against PoS tagging accuracies for different linearizations for the Vietnamese$_{\text{VTB}}$, compared to the no-tags baselines.

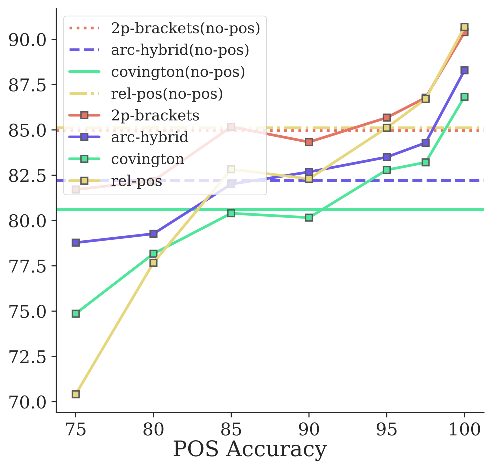


Figure 15: LAS against PoS tagging accuracies for different linearizations for the Norwegian$_{\text{Bokmål}}$, compared to the no-tags baselines.

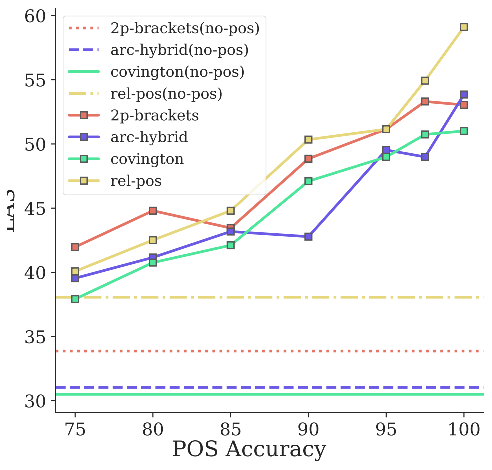


Figure 18: LAS against PoS tagging accuracies for different linearizations for the Skolt Sami$_{\text{Giellagas}}$, compared to the no-tags baselines.

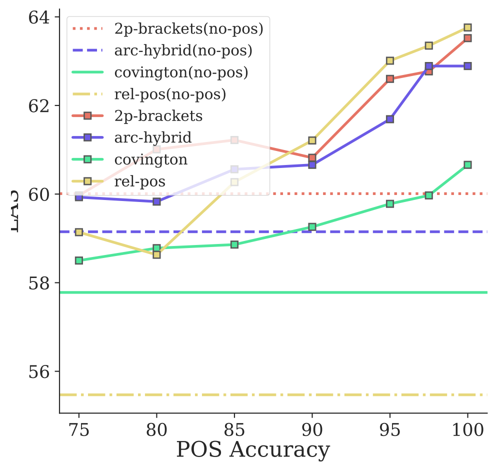


Figure 19: LAS against PoS tagging accuracies for different linearizations for the Turkish$_{\text{BOUN}}$, compared to the no-tags baselines.

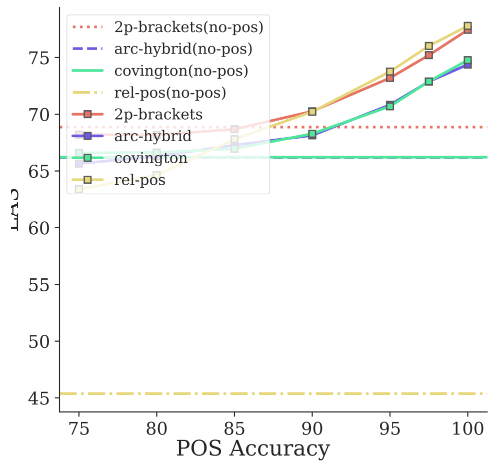


Figure 20: LAS against PoS tagging accuracies for different linearizations for the Welsh$_{\text{CCG}}$, compared to the no-tags baselines.