What Taggers Fail to Learn, Parsers Need the Most

Mark Anderson **Carlos Gómez-Rodríguez**
Universidade da Coruña, CITIC
FASTPARSE Lab, LyS Research Group,
Departamento de Ciencias de la Computación y Tecnologías de la Información
{m.anderson,carlos.gomez}@udc.es

Abstract

We present an error analysis of neural UPOS taggers to evaluate why using gold standard tags has such a large positive contribution to parsing performance while using predicted UPOS tags either harms performance or offers a negligible improvement. We evaluate what neural dependency parsers implicitly learn about word types and how this relates to the errors taggers make to explain the minimal impact using predicted tags has on parsers. We also present a short analysis on what contexts result in reductions in tagging performance. We then mask UPOS tags based on errors made by taggers to tease away the contribution of UPOS tags which taggers succeed and fail to classify correctly and the impact of tagging errors.

1 Introduction

Part-of-speech (POS) tags have commonly been used as input features for dependency parsers. They were especially useful for non-neural implementations (Voutilainen, 1998; Dalrymple, 2006; Alfared and Béchet, 2012). However, the efficacy of POS tags for neural network dependency parsers is less apparent especially when utilising character embeddings (Ballesteros et al., 2015; de Lhoneux et al., 2017). Universal POS (UPOS) tags have still been seen to improve parsing performance but only if the predicted tags come from a sufficiently accurate tagger (Dozat et al., 2017).

Typically using predicted POS tags has offered a nominal increase in performance or has had no impact at all. Smith et al. (2018) undertook a thorough systematic analysis of the interplay of UPOS tags, character embeddings, and pre-trained word embeddings for multi-lingual Universal Dependency (UD) parsing and found that tags offer a marginal improvement for their transition based parser. However, Zhang et al. (2020) found that the only way to leverage POS tags (both coarse and fine-grained) for English and Chinese dependency parsing was to utilise them as an auxiliary task in a multi-task framework. Further, Anderson and Gómez-Rodríguez (2020) investigated the impact UPOS tagging accuracy has on graph-based and transition-based parsers and found that a prohibitively high tagging accuracy was needed to utilise predicted UPOS tags. Here we investigate whether dependency parsers inherently learn similar word type information to taggers, and therefore can only benefit from the hard to predict tags that taggers fail to capture. We also investigate what makes them hard to predict.

2 Methodology

We performed two experiments. The first was an attempt to compare what biaffine parsers learn about UPOS tags by fine-tuning them with tagging information and comparing their errors with those from normally trained UPOS taggers. The second experiment attempted to evaluate the impact tagging errors have by either masking errors or using the gold standard tags for erroneously predicted tags while masking all other tags.

Data We took a subset of UD v2.6 treebanks consisting of 11 languages, all of which are from different language families (Zeman et al., 2020): Arabic PADT (ar), Basque BDT (eu), Finnish TDT (fi), Indonesian GSD (id), Irish IDT (ga), Japanese GSD (ja), Korean Kaist (ko), Tamil TTB (ta), Turkish IMST (tr), Vietnamese VTB (vi), and Wolof WTB (wo). We used pre-trained word embeddings from fastText (for Wolof we had to use the previous Wiki version) (Bojanowski et al., 2017; Grave et al., 2018). We compressed the word embeddings to 100 dimensions with PCA.

	Tagger	Tagger-FT	Parser
Arabic	96.71	96.52	93.73
Basque	95.35	95.18	88.09
Finnish	96.92	96.62	92.24
Indonesian	93.72	93.79	91.98
Irish	92.84	92.80	88.24
Japanese	97.94	97.85	92.80
Korean	95.09	94.26	86.93
Tamil	89.29	87.28	75.41
Turkey	95.10	94.98	86.14
Vietnamese	87.85	87.63	83.40
Wolof	93.85	93.79	85.81

Table 1: Tagging accuracies for tagger trained normally (Tagger), "fine-tuning" a newly initialised MLP for the trained taggers (Tagger-FT), and for parsers fine-tuned to predict tags (Parser).

Experiment 1: Error crossover We trained parsers and taggers on the subset of UD treebanks described above. We then took the parser network and replaced the biaffine structure with a multi-layer perceptron (MLP) to predict UPOS tags. We froze the network except for the MLP and fine-tuned the MLP with one epoch of learning, which is similar to the process used in Vania et al. (2019). We train for only one epoch to balance training the MLP to decode what the system already has encoded without giving it the opportunity to encode more information. We repeated this for the tagger networks (replacing their MLP with a randomly initialised MLP) to validate this fine-tuning procedure. We then compared the tagging errors of both the parsers fine-tuned for tagging and the original taggers. We also undertook an analysis of the errors from the normal taggers which included looking at the impact out-of-vocabulary, POS tag context, and a narrow syntactic context. We define the contexts in Section 3.

Experiment 2: Masked tags We then used the output from the taggers from Experiment 1 to train different parsers. We trained parsers using all the predicted tags, using only the gold standard tags the taggers failed to predict (for both the standard taggers and parsers fine-tuned for tagging), using predicted tags from the standard taggers but masking the errors, and training with all gold standard tags. Note that the respective sets of POS tags were used at both training and inference time. We also trained parsers with no tags as a baseline.

Network details Both the taggers and parsers use pre-trained word embeddings and randomly-initialised character embeddings. The parsers use UPOS tag embeddings as specified in the experimental details. The character and tag embeddings are randomly initialised. The parsers consist of the embedding layer followed by BiLSTM layers and then a biaffine mechanism (Dozat and Manning, 2017). The taggers are similar but with an MLP following the BiLSTMs instead. We ran a small hyperparameter search using fi, ga, tr, and wo and using their respective development data. This resulted in 3 BiLSTM layers with 200 nodes, 100 dimensions for each embedding type with 100 dimension input to the character LSTM. The arc MLP of the biaffine structure had 100 dimensions, 50 for the relation MLP. Dropout was 0.33 for all layers. Learning rate was 2×10^{-3}, β_1 and β_2 were 0.9, batch size was 30, and we trained both taggers and parsers for 200 epochs but with early stopping if no improvement was seen after 20 epochs. Models were selected based on the performance on the development set.

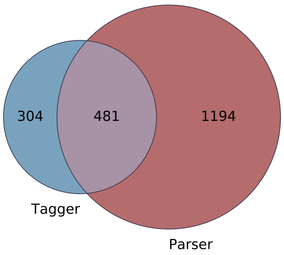

Figure 1: Average union of tagging errors for parser fine-tuned for tagging and fully-trained tagger (standard deviation: 159 for tagger error, 715 for parser, and 242 for union).

3 Results and discussion

Experiment 1: Error crossover Table 1 shows the tagging performance for the normally trained taggers, the re-fine-tuned taggers, and the fine-tuned parser taggers. The re-fine-tuned taggers achieve relatively similar performance to the original taggers, which suggests that this procedure does allow us to develop a decoder that captures

	All	Open	Closed	Other
Tagger	8,637	6,434	1,867	336
Parser	18,426	15,181	2,816	429
Total	171,373	101,965	46,362	23,046

Table 2: Error (Parser, Tagger) and total (Total) counts across all data per word class of gold tag.

	Error Types					Errors	Tokens
ar	noun→x 197	x→noun 139	noun→adj 108	adj→x 78	adj→noun 60	931	28.3K
eu	propn→noun 145	verb→aux 113	noun→adj 101	aux→verb 100	adj→noun 94	1134	24.4K
fi	propn→noun 56	noun→propn 53	noun→adj 43	adj→noun 39	noun→verb 37	649	21.1K
id	propn→noun 147	noun→propn 92	adj→noun 47	noun→adj 34	verb→noun 23	740	11.8K
ga	propn→noun 184	noun→propn 53	noun→adj 53	adj→noun 38	noun→pron 36	724	10.1K
ja	noun→adv 52	propn→noun 24	noun→adj 22	adj→noun 22	aux→verb 20	269	13.0K
ko	noun→propn 252	propn→noun 145	verb→adj 133	aux→verb 78	cconj→sconj 75	1394	28.4K
ta	noun→propn 24	aux→verb 22	propn→noun 17	noun→verb 12	adj→adp 12	213	2.0K
tr	noun→adj 54	propn→noun 52	noun→verb 37	noun→propn 35	adv→adj 31	491	10.0K
vi	noun→verb 201	verb→noun 152	noun→adj 151	verb→adj 140	verb→x 83	1452	12.0K
wo	noun→propn 71	verb→noun 57	pron→det 46	noun→verb 38	verb→aux 30	640	10.4K

Table 3: Top 5 most common errors and their number of occurrences for each treebank. Also shown are the total number of errors and token count for each treebank.

what the BiLSTM and embedding layers learn about UPOS tags without adding new information. Clearly more training would likely improve the parsers fine-tuned for tagging, but it would be less clear if that would be extracting information the parser previously learnt or adding more information via MLP weights.

Figure 1 shows the average cross-over of specific error occurrences for the two systems, where only 38% of the tagger's errors don't occur for the parser. Table 2 shows the breakdown of errors from each system by word type class for all treebanks. The ratio of the errors is substantially different for each class: 0.42 for *open*, 0.66 for *closed*, 0.78 for *other*. This perhaps suggests that the parser has a tendency to learn more syntactically fixed word types than open types. Table 4 shows the F1-score for each UPOS for both systems. For the most part the parser is pretty close to the tagger for open class tags, except for INTJ which the parser never predicts, PROPN (32.7 less for the parser), and to a lesser extent ADJ (13.0 less). Table 3 shows the top 5 most common errors per treebank for the normal taggers where PROPN appears in 15 error types and ADJ appears in 19 out of 55. This prevalence combined with the parsers' poor performance for these tags suggests that errors containing these tags are especially impactful for parsers when using predicted UPOS. However, it could also be that the parsers perform poorly on predicting PROPN tags as they occur in similar syntactic roles as NOUN tokens and as such aren't as important for syntactic analysis.

	F1-score			
	Tagger	Parser	Tokens	Class
PUNCT	99.94	99.93	19.9K	Other
SYM	97.83	0.00	0.2K	
X	76.37	54.51	2.6K	
ADJ	87.98	74.98	9.4K	Open
ADV	93.94	89.97	8.5K	
INTJ	40.91	0.00	0.1K	
NOUN	95.49	94.63	43.7K	
PROPN	90.21	57.49	9.0K	
VERB	94.80	94.05	21.5	
ADP	97.77	94.14	9.9K	Closed
AUX	96.37	93.65	6.8K	
CCONJ	96.30	94.29	7.3K	
DET	94.73	86.88	4.2K	
NUM	93.96	78.12	4.4K	
PART	90.49	76.88	1.7K	
PRON	96.31	72.46	6.0K	
SCONJ	93.15	91.22	3.2K	

Table 4: F1-score for separate tags clustered by word type class with ”Other” at the top, ”Open” in the middle, and ”Closed” at the bottom for all tokens in the collection of treebanks used. Also reported are the total number of tokens for each tag type present across all treebanks (Tokens).

For the closed class type tags, again the parser performs similarly to the tagger but obtains a few points less except for DET, NUM, PART, and PRON with drops for parser scores of 7.9, 15.8, 13.6, and 23.9, respectively. However, of these 4 tags, only PRON and DET appear in the most common errors and only twice and once, respectively. The most common tag to appear in an error is NOUN occurring 41 times, but there is less than one point in difference between the tagger's performance and the parser's for NOUN. Of these 41 appearances, 14 co-occur with ADJ and 15 with PROPN with a fairly even split of mis-tagging NOUN as either of these tags or the other way around. So generally NOUN tokens are fairly easy to tag, but the times where the tagger fails are typically where there is confusion with ADJ and PROPN tags. Figure 2 shows statistical metrics of the taggers' errors. First we show the proportion of out-of-vocabulary (OOV) word

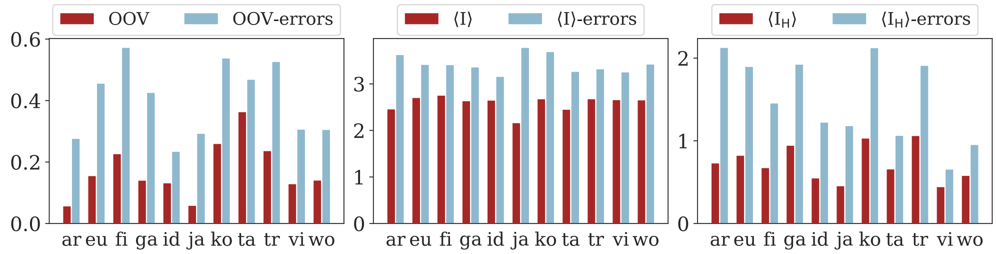

Figure 2: Measurements of all tags (red) and error (blue) tags for OOV proportion, POS bigram surprisal (⟨I⟩, ⟨I⟩-errors), and head POS and relation surprisal (⟨I$_\mathrm{H}$⟩, ⟨I$_\mathrm{H}$⟩-errors).

forms for all tokens and also the tokens where the tagger makes an error. Consistently across all treebanks the OOV proportion is considerably higher for tokens erroneously tagged. Second we report the mean UPOS surprisal. For a given UPOS tag, θ_n for token n, the surprisal of that UPOS tag in a given context, c_k is given as:

$$I(\theta_n) = -\log_2 p(\theta_n|c_k) \tag{1}$$

where we use a bigram context:

$$c_k = (\theta_{n-2}, \theta_{n-1}) \tag{2}$$

Then the mean surprisal, $\langle I \rangle$, over a sample of tokens is given as:

$$\langle I \rangle = \frac{1}{N} \sum_{n \in N} I(\theta_n) \tag{3}$$

where N is the number of tokens in the sample. Again, the mean tag surprisal is substantially different across all treebanks for the tokens where the tagger makes a mistake in comparison to the average over the entire treebank. Finally we report the mean surprisal of UPOS but with the context of its head's tag and the syntactic relation joining the two tokens, such that c_k is defined as:

$$c_k = (\theta_{head}, rel) \tag{4}$$

The difference between the error sub-sample and the whole treebank is starker for the head-relation surprisal, suggesting that the tagger struggles more when the syntactic structure is uncommon.

Experiment 2: Masked tags Table 5 shows the labelled attachment scores for parsers with varying types of UPOS input. First we use the predicted output from the normal taggers from Experiment 1 (Pred) and unlike Anderson and Gómez-Rodríguez (2020) we observe a slight increase over using no UPOS tags. However, using predicted tags isn't universally beneficial. Arabic, Indonesian, Japanese, and Tamil all perform better with no tags.

We then used gold standard tags but masking the tags that the taggers correctly predicted to test if the erroneous tags are particularly useful. We did this for the normal taggers (M¬E$_\mathrm{T}$) and also for the fine-tuned parsers (M¬E$_\mathrm{P}$). The average increase for both is about 2.5 over the no tag baseline and over 2 points better than using predicted tags. Also, the improvement is universal with at least a small increase in performance over using predicted UPOS tags. Interestingly the smaller set from the tagger outperforms the larger set from the parser by 0.15, suggesting that what both the taggers and the parsers fail to capture is more important than the errors unique to the parsers. We then masked the errors from the taggers (M∀E$_\mathrm{T}$) to test if avoiding adding errors would still be beneficial. The performance is almost 2 points better than using the

	None	Pred.	M¬E$_\mathrm{T}$	M¬E$_\mathrm{P}$	M∀E$_\mathrm{T}$	Gold
ar	83.29	82.87	84.17	84.06	84.45	84.73
eu	81.12	81.14	82.33	82.62	83.13	84.45
fi	85.96	86.04	86.88	87.09	87.61	88.80
id	79.04	78.95	82.20	82.69	81.08	82.95
ga	76.13	76.57	76.62	76.65	77.46	77.90
ja	93.15	92.72	94.41	94.38	94.39	95.30
ko	85.40	85.86	87.53	87.82	87.44	88.52
ta	65.61	64.50	70.24	66.67	66.01	71.95
tr	66.67	67.68	67.62	67.66	67.84	68.86
vi	58.43	60.09	65.42	66.75	65.18	70.87
wo	77.87	78.49	82.03	81.39	81.11	85.41
avg	77.52	77.72	79.95	79.80	79.61	81.79

Table 5: LAS parser performance with no tags (None), with predicted tags (Pred), gold standard tags but with all tags masked except those the respective taggers predicted wrong (M¬E$_\mathrm{T}$), similarly for the tagging errors from the fine-tuned parser (M¬E$_\mathrm{P}$), masking the errors from the tagger (M∀E$_\mathrm{T}$), and finally using all gold standard tags.

predicted tags and again an increase is observed for all treebanks. This could be of use, as it is easy to envisage a tagger which learns to predict tags when a prediction is clear and to predict nothing when the probability is low. Finally, using gold standard tags is nearly 2 points better on average than the best masked tag model, which suggests that to fully utilise the information in the final few percentage that taggers miss, the full set of easy to predict tags are needed.

4 Conclusion

We have presented results which suggest that parsers do learn something of word types and that what taggers fail to learn is needed to augment that knowledge. We have evaluated the nature of typical tagging errors for a diverse subset of UD treebanks and highlighted consistent error types and also what statistical features they have compared to the average measurement across all tokens in a treebank. We have shown that it would be more beneficial to implement taggers to not only predict tags but also decide when to do so, as the errors undermine anything gained from using predicted tags for dependency parsers. Note that while we only used one parser system, the original paper (Anderson and Gómez-Rodríguez, 2020) which prompted this work observed similar behaviour with regard to predicted UPOS tags for both the system used here (graph-based) and a neural transition-based parser, suggesting that the results discussed here might extend to other parsing systems. And while it is true that we have only investigated one POS tagger system, we feel we have been careful in not making egregiously grand claims of the universality of our findings: it is merely one data point to be considered amongst many.

Acknowledgments

This work has received funding from the European Research Council (ERC), under the European Union's Horizon 2020 research and innovation programme (FASTPARSE, grant agreement No 714150), from MINECO (ANSWER-ASAP, TIN2017-85160-C2-1-R), from Xunta de Galicia (ED431C 2020/11), and from 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. The authors would also like to the thank the reviewers for their suggestions and criticisms.

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 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.

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*.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

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

Timothy Dozat and Christopher D Manning. 2017. Deep biaffine attention for neural dependency parsing. *Proceedings of the 5th International Conference on Learning Representations*.

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.

Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. 2018. Learning word vectors for 157 languages. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*.

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.

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*.

Clara Vania, Yova Kementchedjhieva, Anders Søgaard, and Adam Lopez. 2019. A systematic comparison of methods for low-resource dependency parsing on genuinely low-resource languages. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1105–1116, Hong

Kong, China. Association for Computational Linguistics.

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.

Daniel Zeman, Joakim Nivre, et al. 2020. Universal Dependencies 2.6. LINDAT/CLARIAH-CZ digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

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*.