# On the Frailty of Universal POS Tags for Neural UD Parsers

**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
Campus Elviña, s/n, 15071 A Coruña, Spain
{m.anderson,carlos.gomez}@udc.es

## Abstract

We present an analysis on the effect UPOS accuracy has on parsing performance. Results suggest that leveraging UPOS tags as features for neural parsers requires a prohibitively high tagging accuracy and that the use of gold tags offers a non-linear increase in performance, suggesting some sort of exceptionality. We also investigate what aspects of predicted UPOS tags impact parsing accuracy the most, highlighting some potentially meaningful linguistic facets of the problem.

## 1 Introduction

Part-of-speech (POS) tags and dependency parsing have formed a long-standing union in NLP. But equally long-standing has been the question of its efficacy. Prior to the prevalence of deep learning in NLP, they were shown to be useful for syntactic disambiguation in certain contexts (Voutilainen, 1998; Dalrymple, 2006; Alfared and Béchet, 2012). However, for neural network implementations, especially those which utilise character embeddings, POS tags have been shown to be much less useful (Ballesteros et al., 2015; de Lhoneux et al., 2017).

Others have found that POS tags can still have a positive impact when using character representations given that the accuracy of the predicted POS tags used is sufficiently high (Dozat et al., 2017). Smith et al. (2018) undertook a systematic study of the impact of features for Universal Dependency (UD) parsing and found that using universal POS (UPOS) tags does still offer a marginal improvement for their transition-based neural parser. The use of fine-grained POS tags still seems to garner noticeable improvements (Ammar et al., 2016).

Latterly, POS tags have been commonly utilised implicitly for neural network parsers in multi-learning frameworks where they can be leveraged without the cost of error-propagation (Zhang and Weiss, 2016; Yang et al., 2017; Li et al., 2018; Nguyen and Verspoor, 2018; Zhang et al., 2020). Beyond multi-learning systems, Strzyz et al. (2019) introduced dependency parsing as sequence labelling by encoding dependencies using relative positions of UPOS tags, thus explicitly requiring them at runtime.

We follow the work of Smith et al. (2018) and evaluate the interplay of word embeddings, character embeddings, and POS tags as features for two modern parsers, one a graph-based parser, Biaffine, and the other a transition-based parser, UUParser (Dozat and Manning, 2017; Smith et al., 2018). Similar to Zhang et al. (2020), we focus on the contribution of POS tags but evaluate UPOS tags.

**Contribution** We analyse the effect UPOS accuracy has on two dependency parser systems for a number of UD treebanks. Our results suggest that in order to leverage UPOS tags as explicit features for these neural parsers, a prohibitively high tagging accuracy is needed, and that gold tag annotation seems to possess some exceptionality. We also investigate what aspects of predicted UPOS tags have the most impact on parsing accuracy.

## 2 Experimental details

We ran three experiments to measure the impact POS[1] tagging accuracy has on parsing performance when using POS tags as features. Experiment 1 considered the POS tagging accuracy as a controlled variable, set by training taggers as described below and then using the output of these taggers as features for parsers. Experiment 2 was similar, except the size of character embeddings were also changed. Experiment 3 was an extension to test the impact of taggers in an optimal setting where they achieve very high accuracies.

[1]From this point on we refer to universal POS tags as POS tags rather than UPOS tags for sake of efficiency.



*Proceedings of the 24th Conference on Computational Natural Language Learning*, pages 69–96

https://doi.org/10.18653/v1/P17

**Data** We use the same subset of UD v2.4 treebanks (Nivre et al., 2019) as Anderson and Gómez-Rodríguez (2020): Ancient Greek Perseus, Chinese GSD, English EWT, Finnish TDT, Hebrew HTB, Russian GSD, Tamil TTB, Uyghur UDT, and Wolof WTB. We used fastText word embeddings for each language except for Ancient Greek and Wolof (Grave et al., 2018). For Ancient Greek we use embeddings from Ginter et al. (2017) and for Wolof those from Heinzerling and Strube (2018). When necessary, we reduced the dimensions to 100 using the algorithm of Raunak (2017).

### 2.1 Methodology

**POS taggers** We train POS taggers for each treebank separately using the sequence-labelling framework NCRF++ (Yang and Zhang, 2018). We train taggers so as to have POS taggers with varying accuracies ranging from 60 to the maximum score the network can achieve (that fits our binning procedure). The accuracy bins we used were increments of 2.5±0.3 from 60 to 80 and increments of 1±0.3 from 80 onwards. We allowed a small window around the desired accuracy for each bin to account for the fact we might never see a model with that exact accuracy. To obtain taggers with varying accuracies, we train each tagger as normal and save models when they reach a certain accuracy. We chose to vary the accuracy of the taggers in this more *natural* way so as to better represent how the taggers would likely behave if they were trained normally but never exceeded the accuracy of a given bin, so it is more likely that easier patterns are learnt first and systematic failures are more likely than if we randomly added noise.

**Network details** We use the default parameters for both parsers, i.e. those reported in each subsequent paper. We use v2.3 of UUParser[2] and use a PyTorch implementation of Biaffine.[3] The features to the networks are the word embeddings as mentioned above, character embeddings, and POS tag embeddings, with the latter two embeddings being randomly initialised. For Experiment 1, the character embedding size was 32 and varied as specified below for Experiment 2. The BiLSTM output dimension of the character embedding layer was 100 and the embedding dimension of the word and POS embeddings were also 100. These dimensions were chosen to control the contribution from each feature, but it is obviously feasible that optimising these contributions could result in different absolute results. However, keeping these static unless purposefully changing them for controlled input means we can make relative comparisons.

**Experiment 1** We trained parsers for each treebank with gold tags and with predicted tags using a subset of the POS taggers with accuracy bins 60, 70, 80, 86, 91, and 93. The values were chosen such that we could cover a reasonable range and include as many treebanks as possible (e.g. only English, Hebrew, and Russian have taggers which achieve 93% accuracy). The parsers trained with predicted tags are run on inputs tagged by the same model, and those trained with gold tags are tested both on gold and a range of predicted tags. The goal of this experiment was to test the sensitivity of parsers to POS tagging accuracy for different treebanks. We also trained parsers without POS tags as a baseline for comparison.

**Experiment 2** We trained parsers for each treebank with gold tags and with predicted tags using a subset of the POS taggers with accuracy bins 80, 86, and the max accuracy for each treebank which was on average 91(3). Each parser is run on inputs tagged by the same model. We used varying character embedding sizes of 32, 100, 180, 325, and 500. We also train parsers with these varying character embedding sizes with no POS tags as a baseline.

**Experiment 3** We trained parsers with and without predicted POS tags for treebanks for which we obtained high-scoring POS taggers with a mean accuracy of 96(2) to evaluate the trend observed in Experiment 1. We use the settings from Experiment 1. The treebanks used were Catalan AnCora, Japanese GSD, Latin ITTB, and Polish PDB.

## 3 Results and analysis

**Experiment 1** The results of Experiment 1 are shown in Figure 1, where the average difference in attachment scores between the baseline parsers (without POS tags) and those with differing POS tag accuracies are shown. We show the differences in attachment scores rather than the absolute values, as averaging over treebanks obscures differences.

There is an unsurprising relation between parsing score and tagging performance when training with gold POS tags. What is less expected is how

[2] UUParser GitHub from Smith et al. (2018).

[3] Biaffine PyTorch GitHub based on Dozat et al. (2017).

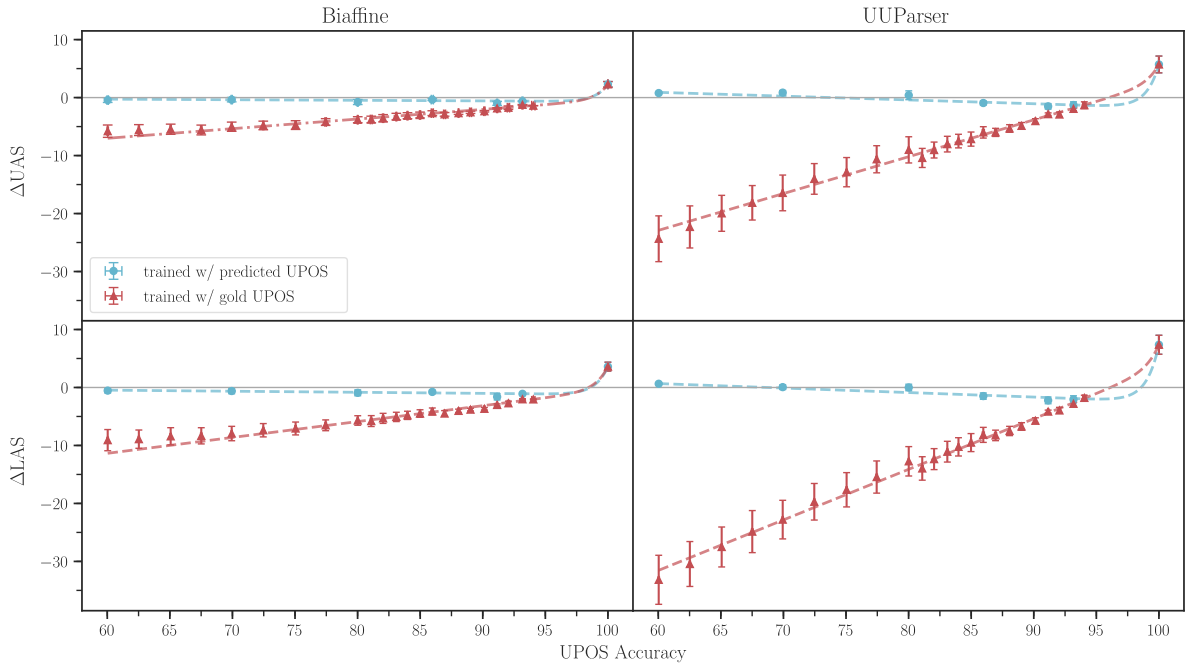


Figure 1: Average Δ attachment scores across all treebanks over the relative baseline parsers trained without POS tags, plotted with respect to POS tag accuracy for parsers trained with predicted (blue, circles) and gold (red, triangles) tags.

little of an impact is observed when using predicted tags during training, with an almost consistent performance with respect to POS tag accuracy.

The gold training trend for the graph-based parser suggests that it is less sensitive to POS tag accuracy than the transition-based parser. This is likely due to the transition-based parser being able to leverage POS tags more, so that it will see more of an impact when tagging accuracy is low. This is somewhat corroborated by the larger positive difference over the baseline when using gold tags at prediction time for UUParser compared to the increase seen for Biaffine.

Another notable phenomenon is that the results for parsing texts annotated with gold POS (rightmost point in each plot) outperform what could be expected from extrapolating the general trends. This raises the question as to whether this is due to smooth nonlinear accuracy increases in the rightmost part of the curves (where we couldn't obtain taggers) or to a sudden jump at the very end in a hockey-stick shape, indicating an exceptionality of gold POS tags and inadequacy of even very accurate but imperfect POS tags (which is relevant under the assumption that tagging accuracy can be pushed further with future model and/or training data improvements). Answering this question was the motivation for Experiment 3.

Almost exclusively, using predicted POS tags does not outperform the parser trained without any POS tags. Curiously, the only parsers that are marginally better are those trained with predicted POS tags from the least accurate POS taggers.

Figures 6–9 in the Appendix show these results for each treebank separately, and almost all treebanks follow the general trend seen for both parsers. The only exception is Tamil TTB for UUParser, which benefits from POS tags both when training with gold and predicted tags. Tamil TTB is the smallest treebank, and it has the additional difficulty for parsing and tagging of being an agglutinative language, so possibly this combination of factors lends itself well to leveraging POS tags even in less than optimal circumstances. Tamil is also the lowest performing language with respect to POS tagging and parsing accuracy, but compared to Uyghur and Ancient Greek (the next two lowest performing languages) it outperforms both when using gold tags. In fact, Tamil has the biggest difference when using gold tags over the baseline than any other language, suggesting that they might be particularly useful when there is a heightened probability of ambiguity coupled with a dearth of data.

**Experiment 2** The average attachment score differences for Experiment 2 are shown in Figure 2. This experiment was initially devised as we anticipated POS tags would have more of a positive

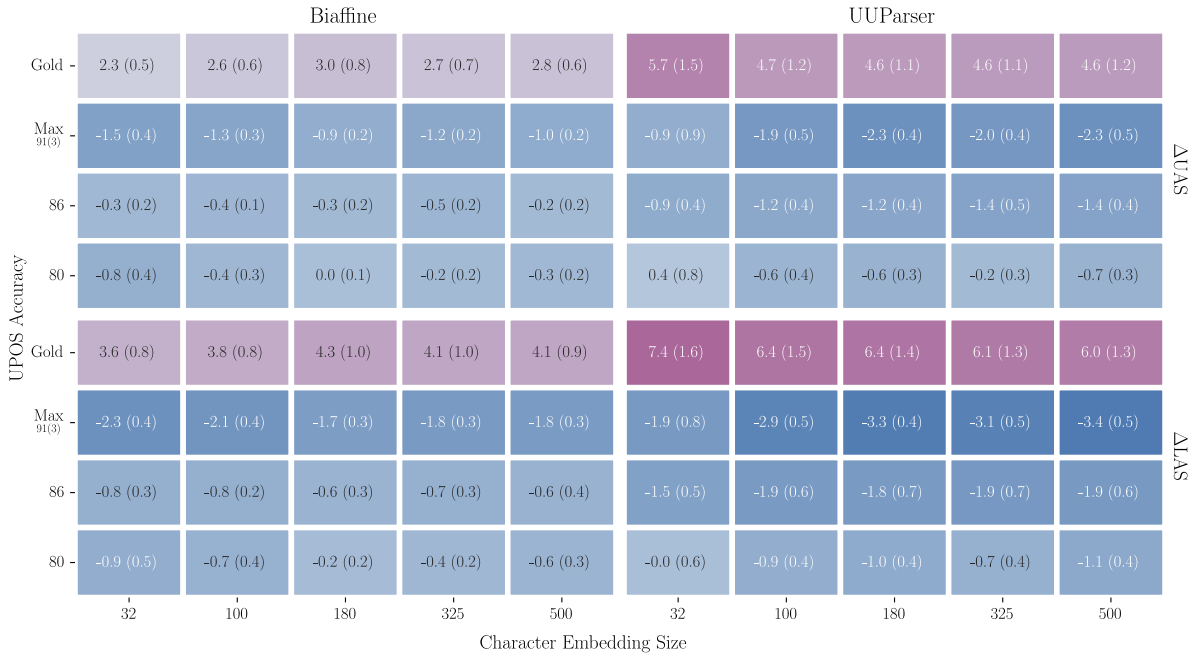


Figure 2: Average Δ attachment scores across all treebanks over the relative baseline parsers trained without POS tags for different character embedding sizes and different POS tag accuracies (80%, 86%, max (average of 91%) tagger accuracy for each treebank, and gold).

effect, especially for higher accuracy taggers, and we wanted to evaluate if having larger character embeddings would offset this. However, as the results of Experiment 1 showed no improvement over not using POS tags at all, this experiment became a verification of the inutility of predicted POS tags instead. And it is clear that in all contexts where predicted tags are used, no matter what the character embedding size is or what parser is used, predicted POS tags perform worse than not using POS tags at all. The unexpected dip in performance as tagging accuracy increases is even clearer here, as this trend is consistent across different character embedding sizes and is the case for both parsers. This decrease in performance is even more marked as the performance actually increases when increasing the character embedding size and not using POS tags at all for UUParser, as shown in Table 1. This result corroborates one of the many observations from Smith et al. (2018). For the graph-based parser there is a negligible negative impact at higher character embedding sizes. Both parser implementations use a BiLSTM to create the character vector input to the network, so this seems more likely to be a result of the transition-based decoder leveraging features more than the graph-based one. The transition-based parser's ability to leverage POS tags in optimal settings is even clearer in Figure 2, as UUParser has twice the improvement using gold tags than that of Biaffine. Also, the impact of predicted POS tags is more pronounced as character embedding sizes increase for UUParser, but for Biaffine there is only a slight tendency to decrease as the character embedding size increases. We show the breakdown for each treebank in Figures 10–13 where again Tamil is clearly an outlier for UUParser, as it is the only language where any settings with predicted POS tags result in a positive increase (80 POS tag accuracy, character embedding size of 32) and has by far and away the largest increase when using gold tags (a factor of 2 greater than the next best improving language, Wolof, for both UAS and LAS).

| | Biaffine | | UUParser | |
|---|---|---|---|---|
| **Char** | UAS | LAS | UAS | LAS |
| **32** | 84.0 (5.9) | 78.6 (8.7) | 77.9 (8.3) | 71.9 (10.0) |
| **100** | 83.9 (6.3) | 78.6 (8.9) | 79.0 (7.3) | 73.0 (9.3) |
| **180** | 83.4 (6.8) | 78.1 (9.4) | 79.1 (7.2) | 73.1 (9.2) |
| **325** | 83.6 (6.7) | 78.1 (9.5) | 79.2 (7.1) | 73.3 (8.9) |
| **500** | 83.6 (6.4) | 78.1 (9.1) | 79.3 (7.0) | 73.4 (8.8) |

Table 1: Average attachment scores for different character embedding sizes (Char) without POS tags.

**Experiment 3** In Figure 1, there is a point around 96-98 POS tag accuracy where the parsers outperform the baselines without POS tags. Due to a lack

of models in that range, this is just an extrapolation. So we trained parsers with treebanks, listed above in the description of Experiment 3, for which we could obtain high POS tagging accuracy. The results of these parsers are shown in Table 2. Only the top two treebanks with the highest tagging accuracy (Catalan AnCora and Japanese GSD) perform better than using no POS tags, and only for UUParser. However, when the performance is below the baseline the difference is marginal. These results are consistent with the extrapolations in Figure 1 and suggest a sharp increase in the Δ attachment score slopes when POS tagging accuracy is in the 98-100 range, i.e., that predicted POS tags suddenly start being useful when they are very close to gold POS tags. This suggests that there may be certain tag patterns or contexts that are particularly relevant for parsing, but especially difficult for taggers to learn.

### 3.1 Parsing difficulty of POS tags

We then delved deeper by looking at the difficulty of predicting arcs and labels for each POS tag type. The full results are shown in Figures 14 and 15 in the Appendix, where the average differences in score with respect to the baseline model (no POS tags) are given. `X` (the UPOS tag for "other") is consistently difficult across parsers and parser types, except that the loss in UAS for UUParser is much smaller than for Biaffine for both training with predicted and gold tags. However, the only time `X` is consistently better than the baseline model for LAS is when using gold tags at runtime and only with UUParser.

Another noticeable feature in these results is that for the max POS predicted accuracy and gold tag parsers for UUParser, `INTJ` (interjection) performs significantly better, both using predicted POS tags and gold tags for training, compared to the lower POS tag accuracy parsers. Beyond this, the performances echo the global scores with respect to the tagging accuracy.

Next, we evaluated the correlations (Pearson co-

| | Biaffine | | UUParser | | |
|---|---|---|---|---|---|
| | UAS | LAS | UAS | LAS | $POS_{ACC}$ |
| *Catalan-AnCora* | | | | | |
| Predicted | 92.59 | 89.57 | **90.88** | **88.03** | 98.26 |
| None | **92.89** | **90.33** | 90.82 | 87.92 | n/a |
| *Japanese-GSD* | | | | | |
| Predicted | 95.02 | 93.66 | **94.56** | **92.94** | 97.69 |
| None | **95.12** | **93.54** | 94.47 | 92.74 | n/a |
| *Polish-PDB* | | | | | |
| Predicted | 92.78 | 89.97 | 89.25 | 85.57 | 97.52 |
| None | **93.64** | **90.94** | **89.32** | **85.60** | n/a |
| *Latin-ITTB* | | | | | |
| Predicted | 90.92 | 88.47 | 86.99 | 83.99 | 97.46 |
| None | **91.09** | **88.74** | **87.25** | **84.25** | n/a |

Table 2: Performance for treebanks with high scoring POS taggers trained with predicted POS tags, compared to the performance on the same treebanks without using POS tags.

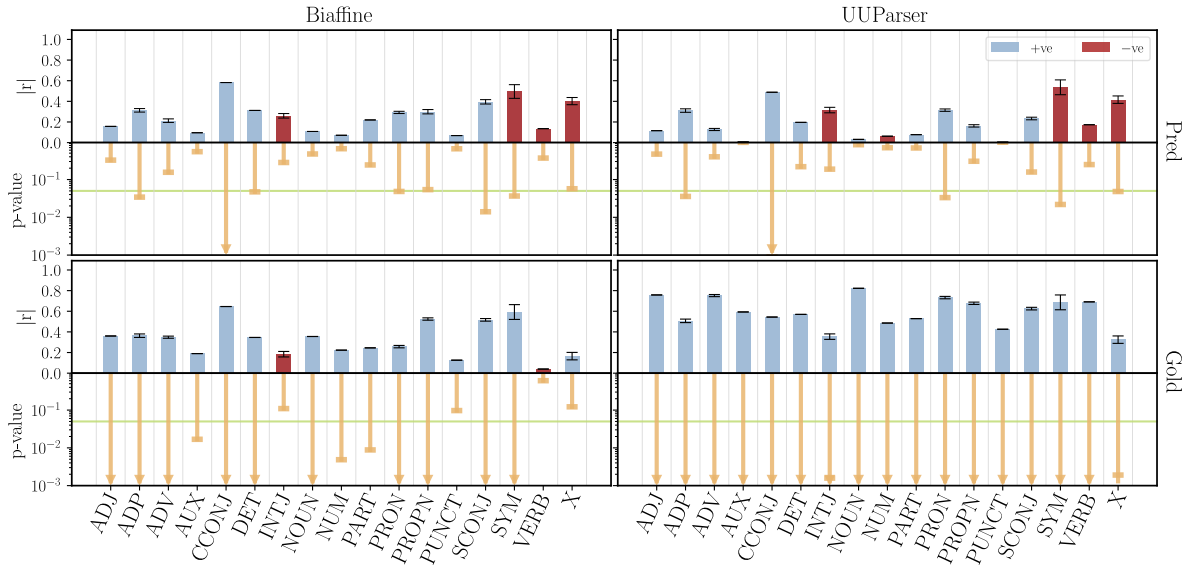


Figure 3: Pearson coefficients for the F1-score for separate POS tags and global LAS where positive (+ve) coefficients are shown in blue and negative (-ve) are shown in red. The corresponding p-values are shown below (orange) where an arrow head means the value was below 0.001. Left subplots are for Biaffine parsers, right for UUParsers, top row is for parsers trained with predicted tags, and bottom for parsers trained with gold tags.

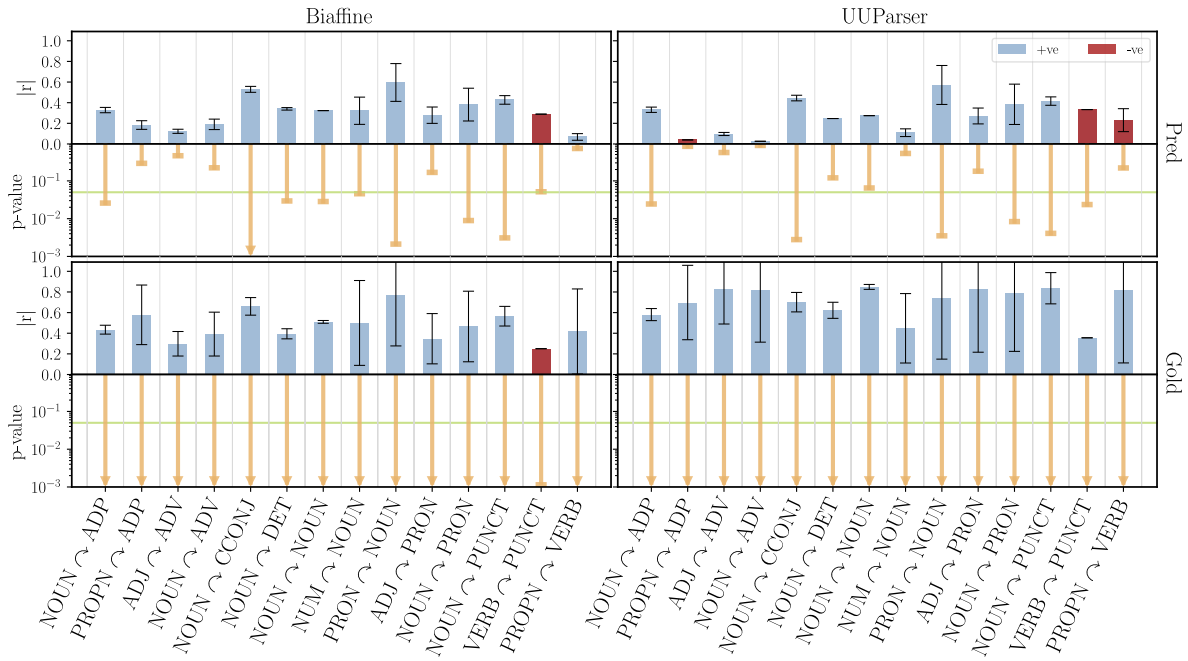


Figure 4: Pearson coefficients for the tagging F1-score for child↷head pairs and global LAS where positive (+ve) coefficients are shown in blue and negative (-ve) are shown in red. The corresponding p-values are shown below (orange) where an arrow head means the value was below 0.001. Left subplots are for Biaffine parsers, right for UUParsers, top row is for parsers trained with predicted tags, and bottom for parsers trained with gold tags.

efficient) between tagging accuracy for each POS tag and global parsing performance. For these correlation results and all those that follow, we use the same taggers and parsers from Experiment 1. We only report results for LAS for the sake of space.

Figure 3 shows the Pearson coefficient with the corresponding p-value for the correlations between the F1-score for each POS tag and the global LAS score for both Biaffine and UUParser, for both predicted and gold POS tags used in training. Training with gold tags, the accuracy for every tag is positively correlated with parsing performance for UUParser and the correlations are all statistically meaningful. The correlations range from about 0.4 (INTJ and X) to about 0.8 (ADJ, ADV, NOUN, and PRON). For Biaffine, the correlations are much weaker ranging from 0.2 (AUX) to 0.6 (CCONJ, coordinating conjunction, and SYM, symbol) for those which are statistically significant.

For the systems trained with predicted POS tags, the correlations are much weaker for UUParser and only 5 are statistically significant. UUParser and Biaffine have much more similar correlations under these settings, where Biaffine has 2 other tags significantly correlated but its set contains those of UPParser. Of those that are significantly correlated for both, SYM and X are actually negatively correlated, suggesting that the taggers either fail to generalise or fail to capture certain tagging patterns. A noticeable exception is the CCONJ tag which is both strongly correlated (about 0.6 for both) and statistically significant for both parsers. This is likely due to the nature of conj relations, where dependents are connected to the conjunct rather than the head of the conjunct (e.g. the second conjoined object of a verb is connected to the first) and so should be parsed differently than if they occurred without a CCONJ.

Figure 24 in the Appendix shows the correlation for the tagging accuracy of the head for each tag type. Across all systems, there is a correlation for the head of INTJ nodes (0.7 for predicted training, 0.5 for gold). This is perhaps due to INTJ nodes typically being attached to VERB or NOUN nodes, and that this narrow context means that the parsers will always look for a node like these and if the correct node is incorrectly tagged, this could disrupt the arc predictions and would be better off without the tagging information.

ADP (adposition) nodes are similar but with a lower correlation (about 0.4 for all systems). And again this might be due to these nodes occuring in less diverse contexts. X nodes are strongly negatively correlated for Biaffine for both gold and predicted training systems (0.6 and and 0.8 respectively) and similarly SCONJ (subordinating con-

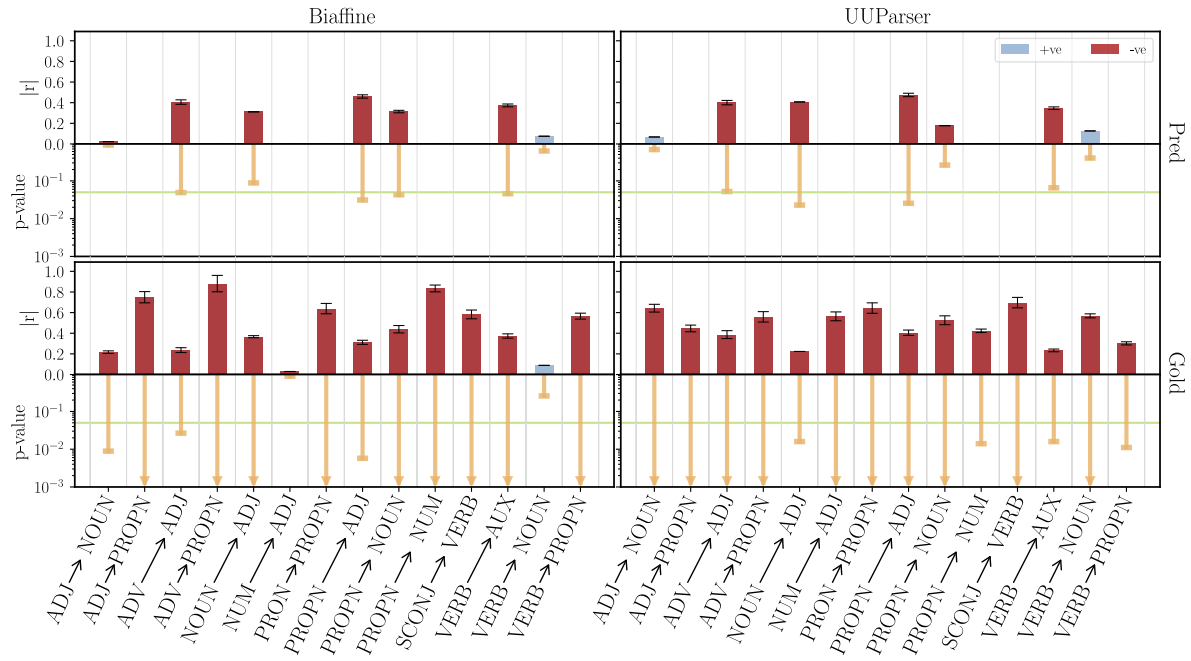


Figure 5: Pearson coefficients for the error rate of individual error types $\texttt{POS}_X \rightarrow \texttt{POS}_Y$ and global LAS where positive (+ve) coefficients are shown in blue and negative (-ve) are shown in red. The corresponding p-values are shown below (orange) where an arrow head means the value was below 0.001. Left subplots are for Biaffine parsers, right for UUParsers, top row is for parsers trained with predicted tags, and bottom for parsers trained with gold tags.

junction) nodes (0.7 and 0.5). Perhaps the diversity of the contexts in which these tags occur makes it difficult for the parser to leverage POS information. ADV nodes follow a similar trend, being negatively correlated for 3 of the 4 systems (gold UUParser being the exception), which could also be related to diversity of contexts: adverbs such as *very* never attach to verbs, but to other adverbs, and often the use of ADV covers situations where a word doesn't satisfy the definition of another POS tag.

Figure 4 shows the correlation of combining the accuracy of POS tags and the tags that govern them with global LAS scores. Only pairs that occur 10 times in 4 treebanks are included. The union of pairs with the highest correlations across all systems are shown (the 10 most highly correlated and statistically significant for each parser). The correlations are positive with one exception of PUNCT nodes headed by VERB nodes, which are weakly negative for all systems except gold UUParser. Conversely, PUNCT nodes headed by NOUN nodes have positive correlations for all systems (0.4 for all except gold UUParser which is about 0.8). Other than this, CCONJ nodes headed by NOUN nodes are positively correlated (0.6-0.7) for all systems, which adds to the discussion above regarding CCONJ tags and suggests that it helps more specifically when conjuncts are NOUN nodes.

### 3.2 Dependency distance

Figures 16–19 in the Appendix show the attachment scores and occurence rates for each POS tag in dependency distance bins. Most tags decrease in performance as the distance increases. Other than NOUN, PUNCT, and VERB, the occurrence of longer-distanced edges are significantly lower than short-distanced ones. Of these, NOUN has a much more significant drop in performance as distance increases across all systems.

Figure 25 in the Appendix shows the combinations of POS tag and dependency distance with highest correlation with LAS. CCONJ appears in 8 pairs (out of 24) and appears 3 out of 6 times for the distances of 3 or less. This further supports the findings from above that awareness of CCONJ nodes is especially beneficial. Beyond this, most pairs (19) have distances of 4 or greater which is larger than the mean dependency distance typically observed in natural languages, e.g. it is 3.6 (0.4) averaged over all treebanks in UD v.2.4 using the equation from Liu (2008).

### 3.3 Error types

Finally, we evaluated which type of tagging errors are the most likely to impact parsing performance.

In Figures 20–23 in the Appendix, we show the corresponding attachment scores and counts of each error of tagging a gold tag of $\texttt{POS}_X$ as $\texttt{POS}_Y$ in confusion matrices for both parser types, for training with gold tags and with predicted tags with taggers of accuracy 80, 86, 91(3). We include these to supplement the following analysis and allow for comparisons to other error types that aren't shown. However, we can see that a lower occurrence rate of errors is associated with lower attachment scores and errors have a larger impact on LAS than UAS.

Figure 5 shows the highest correlated and statistically significant tagging errors. Correlations are between the error rate and the global LAS scores. Only error types that occur 10 times in the output of at least two taggers for at least 4 treebanks are included (the 5 most correlated for each parser). This is due to the fact that looking at the correlation between error rates and LAS when an error type rarely occurs will still give *statistically meaningful* correlations, as the absence of stats is one step removed from the correlation calculation. Error types are negatively correlated with parsing performance (the exceptions are those which aren't statistically significant for some systems). Correlations are strongest when training with gold POS tags. For Biaffine they are either much stronger or much weaker than UUParser, e.g. ADJ→PROPN is over 0.8 whereas it is only about 0.5 for UUParser, PROPN→NUM is about 0.8 for Biaffine and about 0.4 for UPParser. In contrast, ADJ→NOUN and ADV→ADJ are only about 0.2 for Biaffine but are about 0.6 and 0.4, respectively, for UUParser.

Two POS tag pairs appear in error types where both directions are observed, PROPN↔ADJ and NOUN↔ADJ for both parsers trained with gold tags. For the former, it appears that qualifiers that refer to nations or groups are often problematic as a similar form or the same one appear as PROPN and ADJ, e.g. *Sunni*, *African*, *Mexican*. For the error type ADJ→PROPN, another issue seems to be the capitalisation of certain words which either appear on their own or with limited punctuation, e.g *Wonderful!*, *Marvelous!*, or refer to something fixed but not quite a named entity, e.g. *Parliamentary elections*, *Perfect Score*. This is the case in English, and we apologise for the Anglo-bias, but the author isn't proficient in the other languages used. However, these errors do occur at a general level. It appears to be similar in Russian (Бургундского - *Burgundy*, Гомельская - *Gomel Region*); Finnish (*Suomalaisen - Finnish*, *eurooppalaisen - European*); and in Hebrew (איטלקית - *Italian*, גרמנית - *German*). The only language where neither of these error types occur is Wolof as it doesn't have an adjective category (Dione, 2019).

For the other bidirectional error type (NOUN↔ADJ), there appears a similar issue for ADJ→NOUN as ADJ↔PROPN for nations or groups, but NOUN is used instead of PROPN. Beyond this, when a NOUN is incorrectly tagged as an ADJ this occurs 44.7 (14.4)% when it is governed by another NOUN. This is especially prominent for English and Hebrew (65.8% and 64.4% respectively) with the lowest rate occurring for Ancient Greek and Tamil (25.8% and 28.9% respectively). The issue of tagging NOUN tokens governed by another NOUN token is also apparent in Figure 4 where this pair has a correlation coefficient of about 0.4 for both Biaffine systems and 0.8 for the gold trained UUParser system (for the predicted POS tag UUParser system, it isn't statistically significant). Again Wolof is an outlier as the error NOUN→ADJ never occurs, presumably because it never has any ADJ tokens to learn.

Only two error types are statistically significant for all systems: ADV→ADJ and PROPN→ADJ, the latter having been discussed above. The former isn't particularly prevalent, occurring with an error rate of 5.8 (5.7)% on average across all languages (except Wolof) with Russian and Tamil having the highest rates (15.5% and 15.0%, respectively) and Chinese and Hebrew having the lowest (0.6% and 1.7%). For English at least, two issues are clear. Words that have the same form when used as an adverb or adjective are commonly mis-tagged as ADJ when they should be ADV, e.g. *more*, *worst*, *better*, and so on. And also when an adverb is used in hyphenated adjectival phrases such as *fully* in *fully-fledged* and *ill* in *ill-advised*.

As Wolof was such an outlier with respect to common tagging errors (with those that impacted parsing performance) we looked at those most common in Wolof. DET→TAG occur more often than average, especially DET→VERB (error rate of 10.0% compared to 2.4(4.1)% for other languages) and DET→PRON (error rate of 13.5% compared to 7.6(6.3)% for other languages). DET→NOUN is also common but similar to the other languages (error rate of 8.0% compared to 7.0(7.0)% for other languages). DET→VERB and DET→NOUN are negatively correlated for Wolof with an average coefficient of $-0.85(0.11)$ across all systems and all with $p < 0.05$. Clearly, further language-specific

analyses are needed.

## 4 Conclusion

We have evaluated the impact POS tag accuracy has on parsing performance for leading graph- and transition-based parsers across a diverse range of UD treebanks, highlighting the stark difference between using predicted POS tags and gold POS tags at runtime. We observed a non-linear increase in performance when using gold tags, suggesting they are somehow exceptional, i.e., precisely the tag patterns that not even the most accurate taggers can correctly predict (the last 2-3 percentage points towards 100% accuracy) seem to be the most important for parsing. This could be due to the parsers implicitly learning POS tag information, in such a way that the taggers learn nothing new to contribute or not enough to avoid a loss in performance due to the errors disrupting what the parsers have learnt. Our analysis also shows that practitioners should evaluate the efficacy of using predicted tags for a given system or language. We have also analysed what aspects of erroneous tagging predictions have the greatest impact and correlation to parsing performance. We observed some global trends, like the importance of `CCONJ`, but also language-specific issues which highlight the need to evaluate the usefulness of POS tags per language. The results also suggest that using a subset of POS tags might be effective.

## 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 *all* the reviewers for their detailed and constructive comments 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.

Waleed Ammar, George Mulcaire, Miguel Ballesteros, Chris Dyer, and Noah A Smith. 2016. Many languages, one parser. *Transactions of the Association for Computational Linguistics*, 4:431–444.

Mark Anderson and Carlos Gómez-Rodríguez. 2020. Distilling neural networks for greener and faster dependency parsing. In *Proceedings of the 16th International Conference on Parsing Technologies*, pages 2–13.

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

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

Cheikh M Bamba Dione. 2019. Developing universal dependencies for Wolof. In *Proceedings of the Third Workshop on Universal Dependencies (UDW, SyntaxFest 2019)*, pages 12–23.

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.

Filip Ginter, Jan Hajic, Juhani Luotolahti, Milan Straka, and Daniel Zeman. 2017. CoNLL 2017 shared task-automatically annotated raw texts and word embeddings. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics, Charles University.

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)*.

Benjamin Heinzerling and Michael Strube. 2018. BPEmb: Tokenization-free pre-trained subword embeddings in 275 languages. In *Proceedings of the Eleventh 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.

Zuchao Li, Shexia He, Zhuosheng Zhang, and Hai Zhao. 2018. Joint learning of POS and dependencies for multilingual universal dependency parsing. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 65–73.

Haitao Liu. 2008. Dependency distance as a metric of language comprehension difficulty. *Journal of Cognitive Science*, 9(2):159–191.

Dat Quoc Nguyen and Karin Verspoor. 2018. An improved neural network model for joint POS tagging and dependency parsing. *CoNLL 2018*, page 81.

Joakim Nivre, Mitchell Abrams, Željko Agić, et al. 2019. Universal Dependencies 2.4. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Vikas Raunak. 2017. Simple and effective dimensionality reduction for word embeddings. *Proceedings of NIPS LLD Workshop*.

Aaron Smith, Bernd Bohnet, Miryam de Lhoneux, Joakim Nivre, Yan Shao, and Sara Stymne. 2018. 82 treebanks, 34 models: Universal dependency parsing with multi-treebank models. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 113–123.

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. In *EMNLP 2018: 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2711–2720.

Michalina Strzyz, David Vilares, and Carlos Gómez-Rodríguez. 2019. Viable dependency parsing as sequence labeling. In *NAACL-HLT 2019: Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 717–723.

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.

Jie Yang and Yue Zhang. 2018. NCRF++: An open-source neural sequence labeling toolkit. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*.

Liner Yang, Meishan Zhang, Yang Liu, Maosong Sun, Nan Yu, and Guohong Fu. 2017. Joint POS tagging and dependence parsing with transition-based neural networks. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 26(8):1352–1358.

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*.

Yuan Zhang and David Weiss. 2016. Stack-propagation: Improved representation learning for syntax. *arXiv preprint arXiv:1603.06598*.

# Appendix A Treebank performances

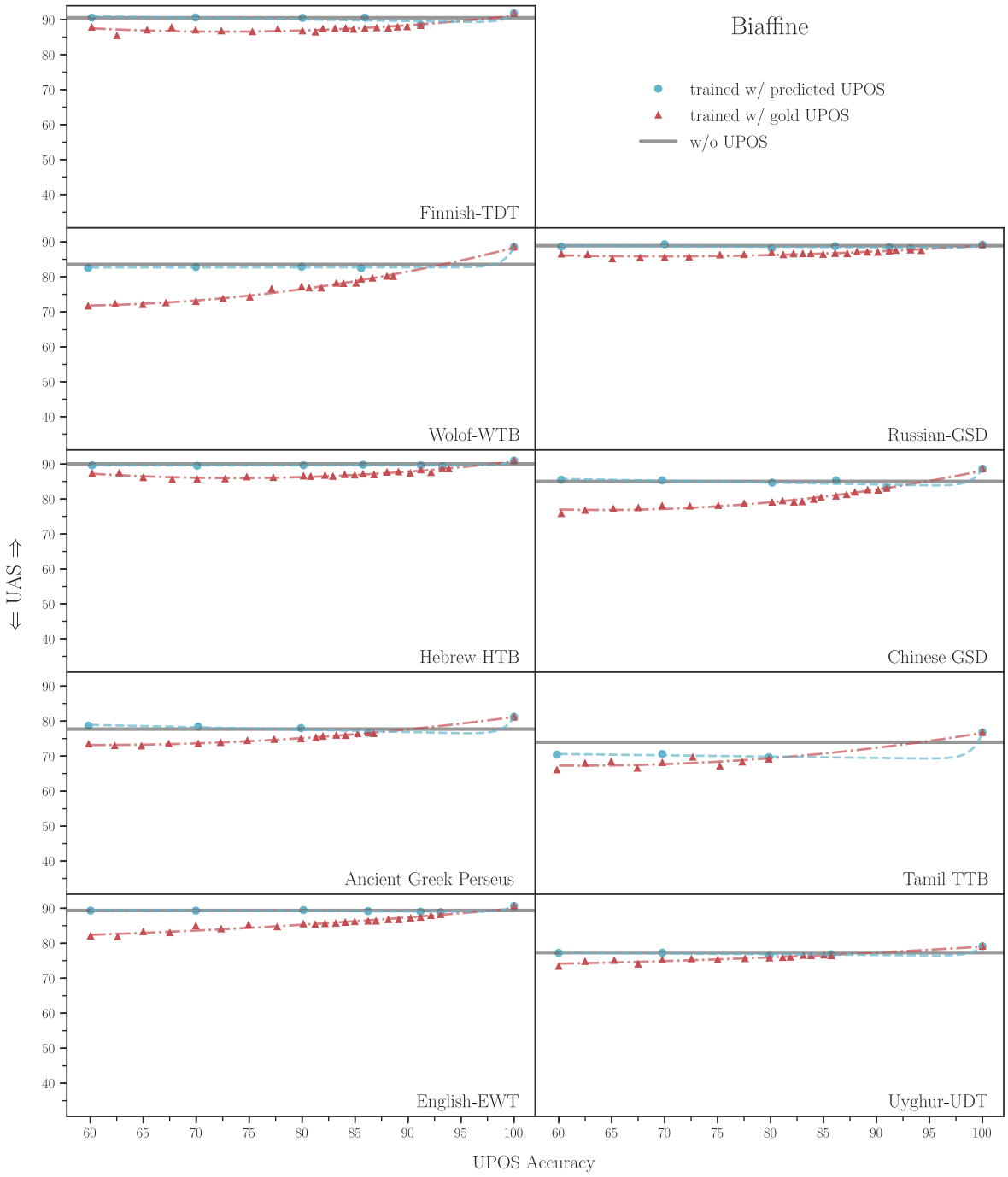


Figure 6: UAS for each treebank for Biaffine training with gold (red, triangles) and predicted (blue, circles) POS tags. Baseline parser trained without POS tags is shown in grey.

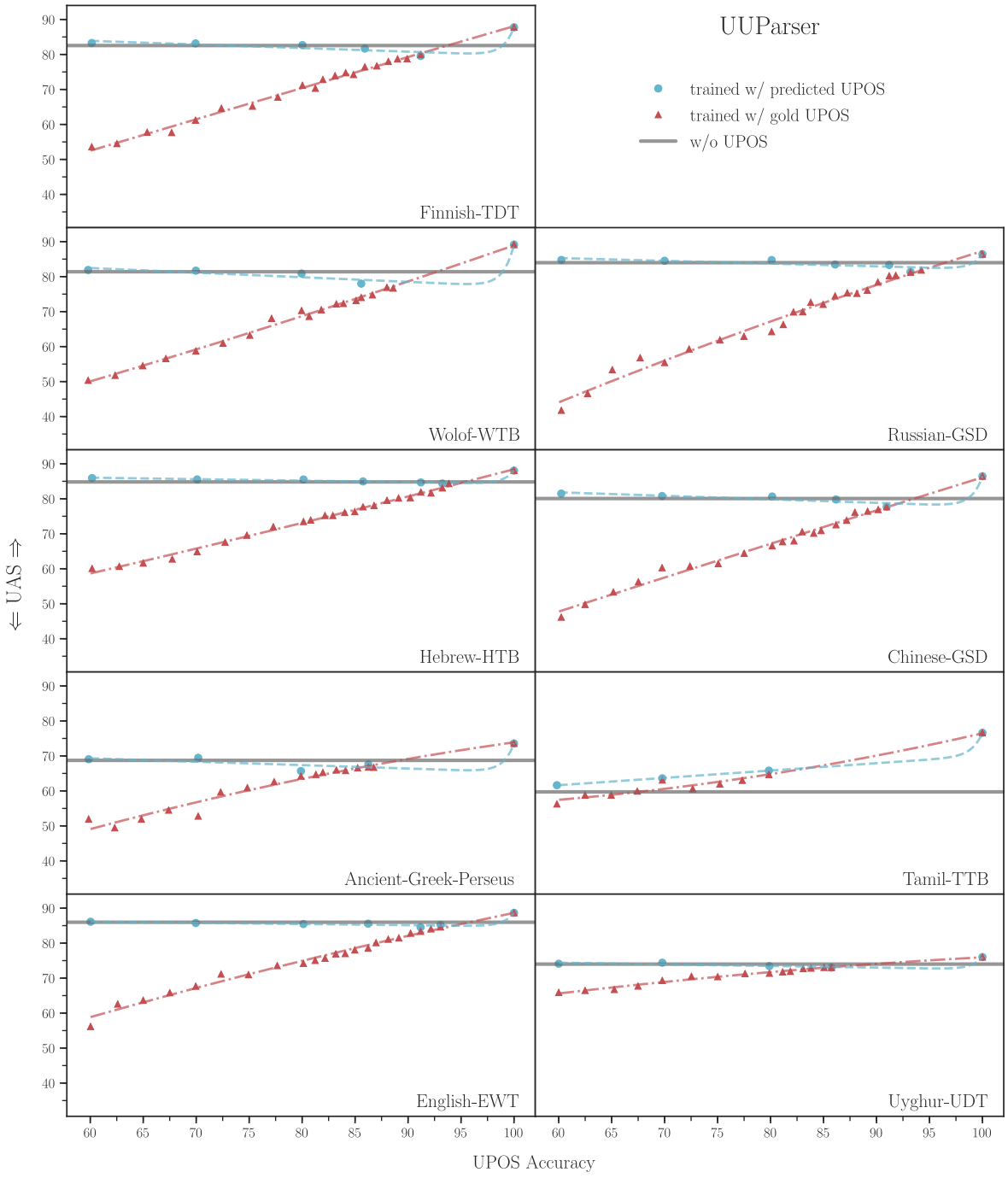


Figure 7: UAS for each treebank for UUParser training with gold (red, triangles) and predicted (blue, circles) POS tags. Baseline parser trained without POS tags is shown in grey.

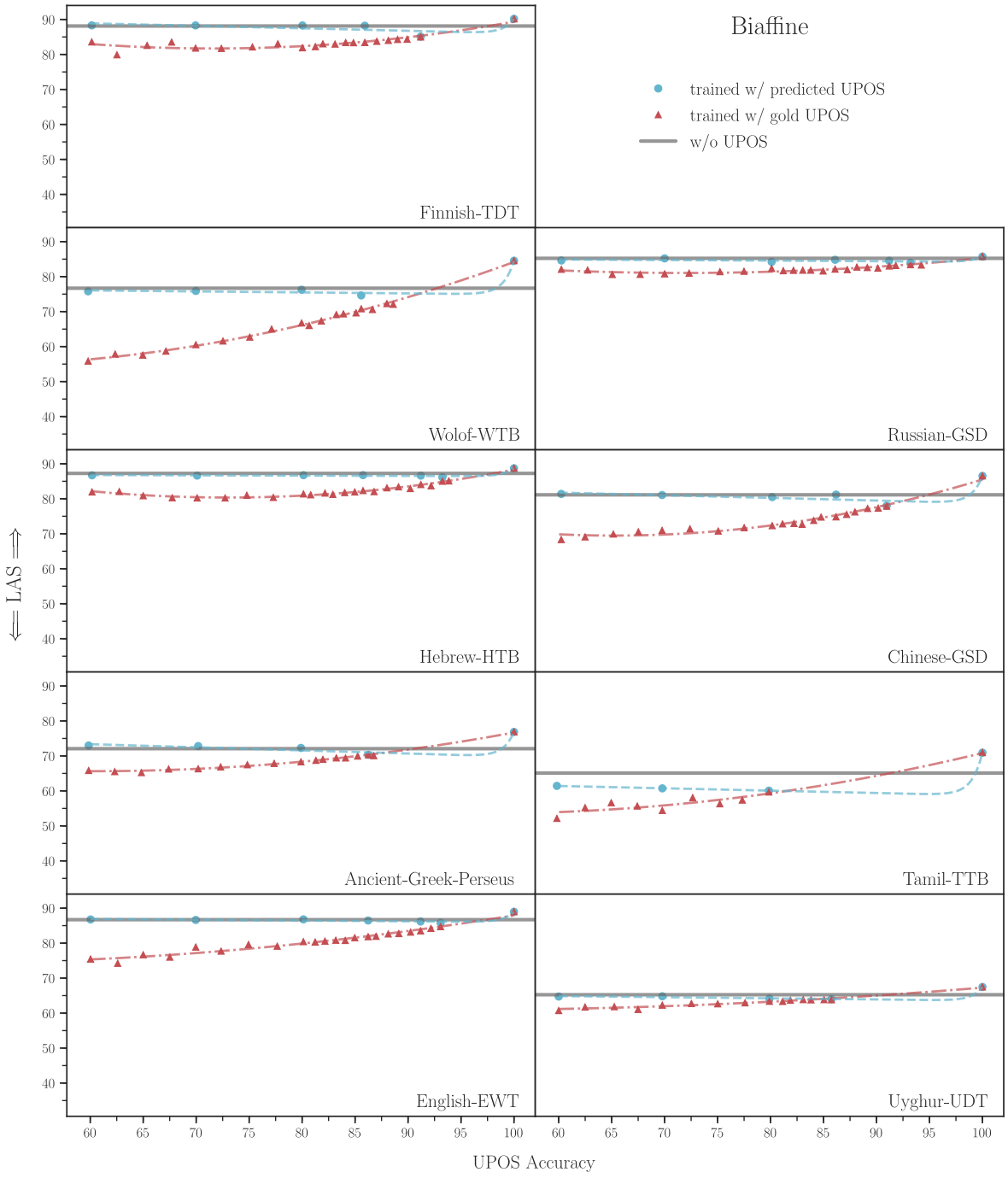


Figure 8: LAS for each treebank for Biaffine training with gold (red, triangles) and predicted (blue, circles) POS tags. Baseline parser trained without POS tags is shown in grey.

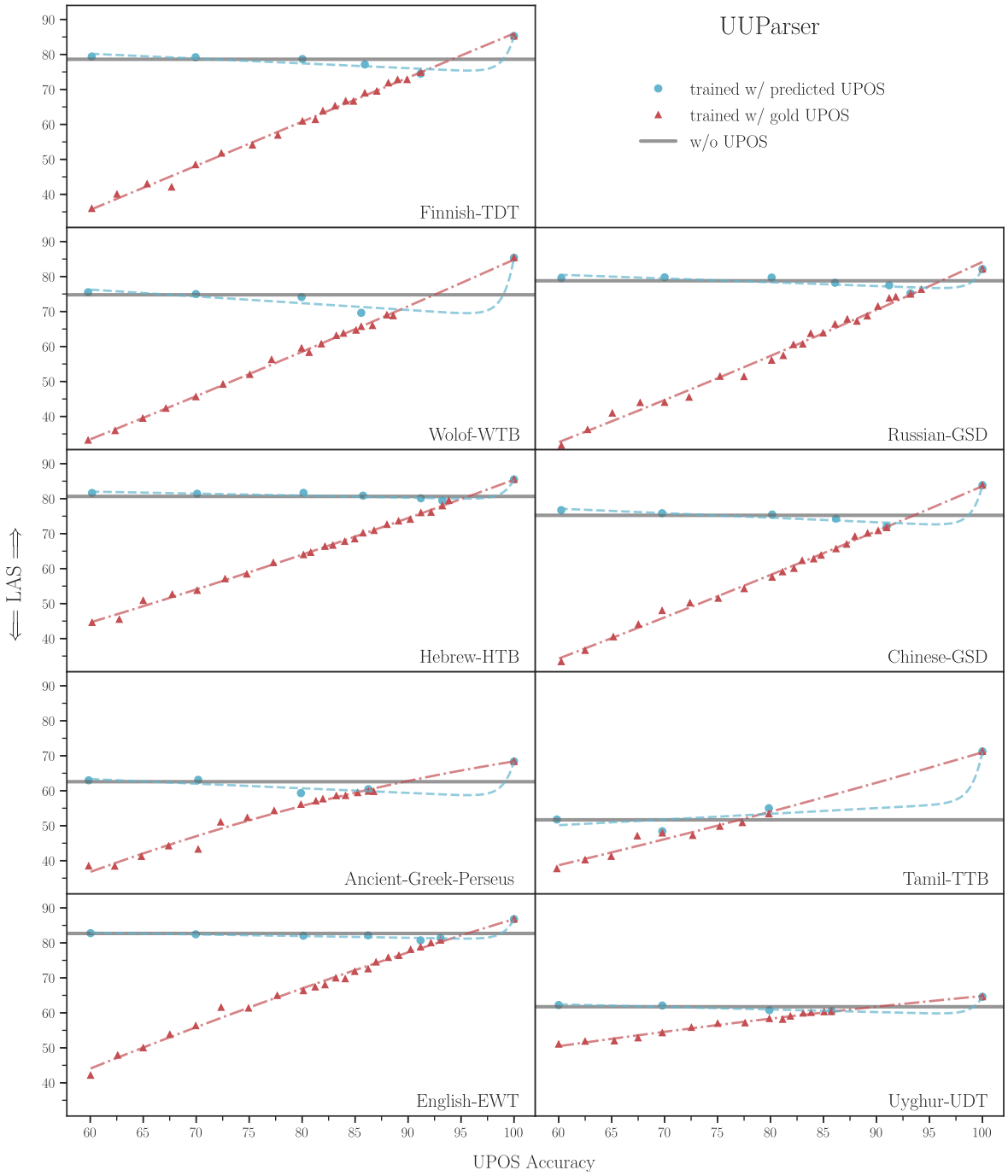


Figure 9: LAS for each treebank for UUParser training with gold (red, triangles) and predicted (blue, circles) POS tags. Baseline parser trained without POS tags is shown in grey.

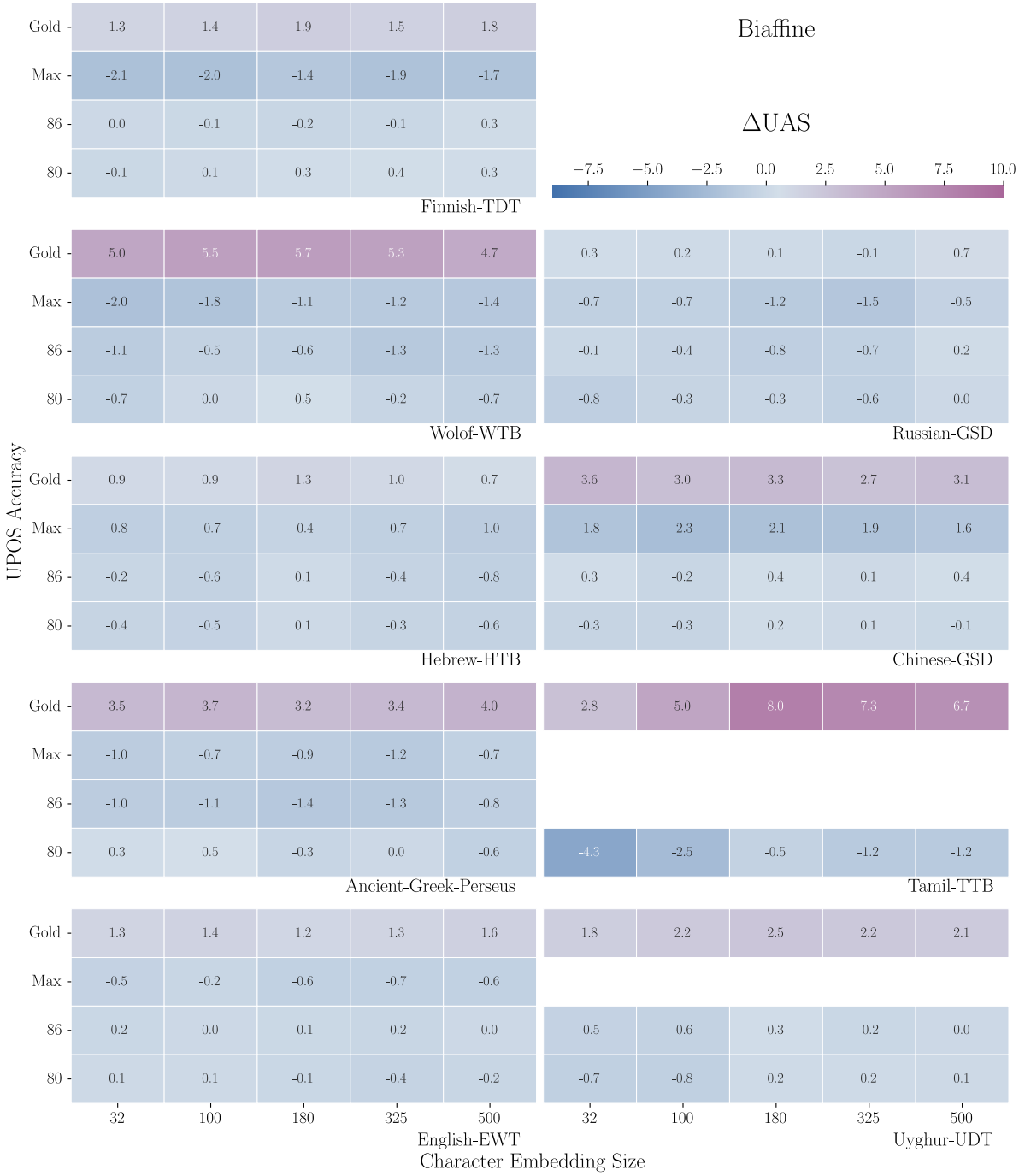


Figure 10: Δ UAS for each treebank for Biaffine compared to the baseline parsers trained without POS tags for different character embedding sizes and different POS tag accuracies.


Figure 11: Δ UAS for each treebank for UUParser compared to the baseline parsers trained without POS tags for different character embedding sizes and different POS tag accuracies.


Figure 12: Δ LAS for each treebank for Biaffine compared to the baseline parsers trained without POS tags for different character embedding sizes and different POS tag accuracies.


Figure 13: Δ LAS for each treebank for UUParser compared to the baseline parsers trained without POS tags for different character embedding sizes and different POS tag accuracies.

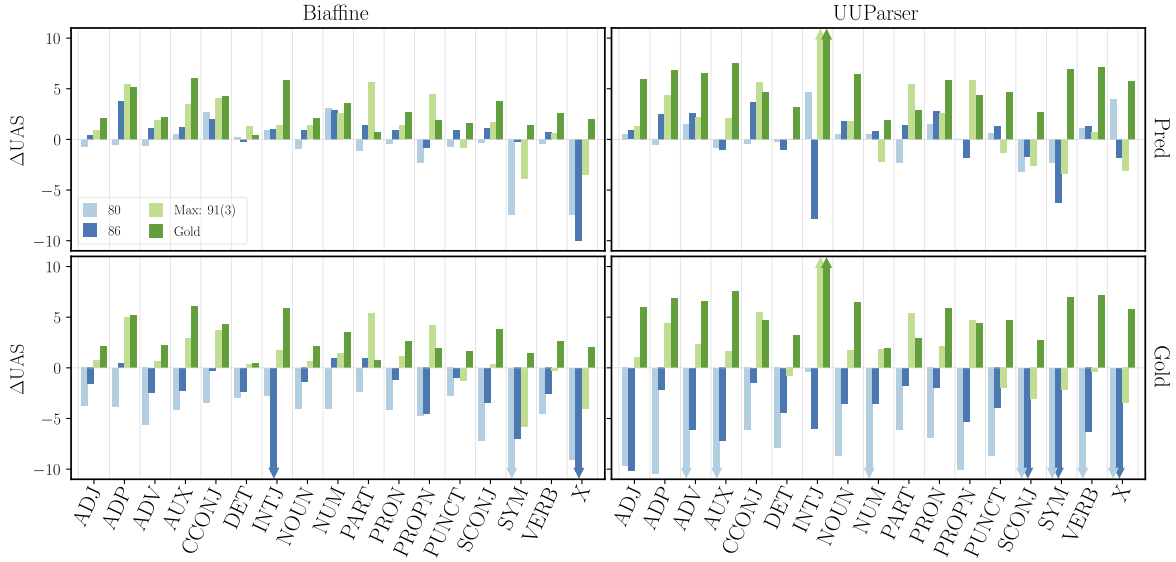


Figure 14: Average ΔUAS across all treebanks for models trained with POS tags from taggers of 80, 86, max POS accuracy of 91(3), and with gold tags for each POS tag.

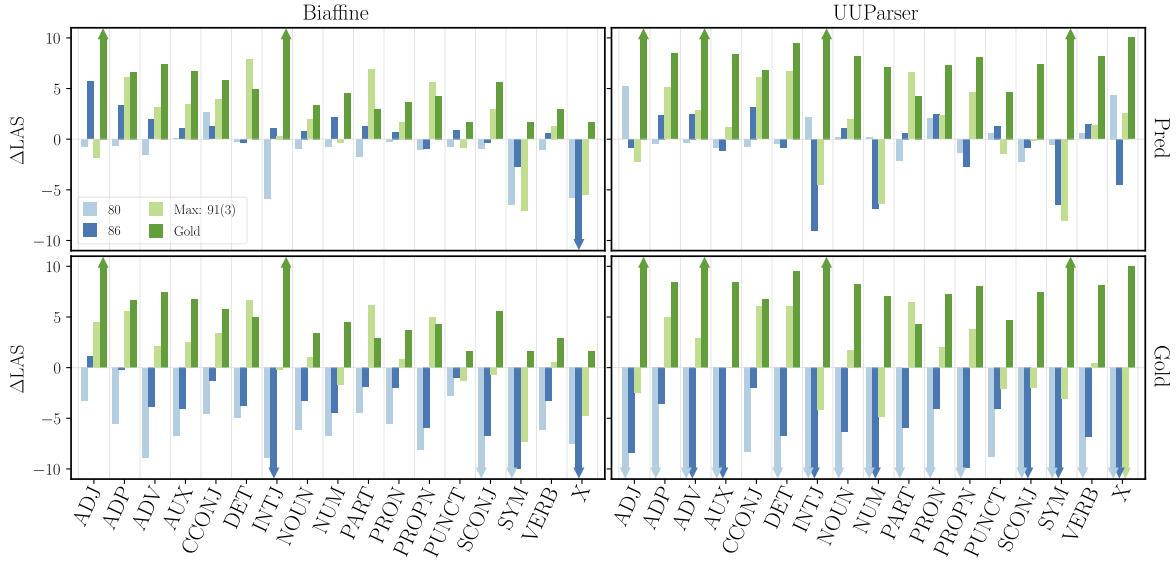


Figure 15: Average ΔLAS across all treebanks for models trained with POS tags from taggers of 80, 86, max POS accuracy of 91(3), and with gold tags for each POS tag.

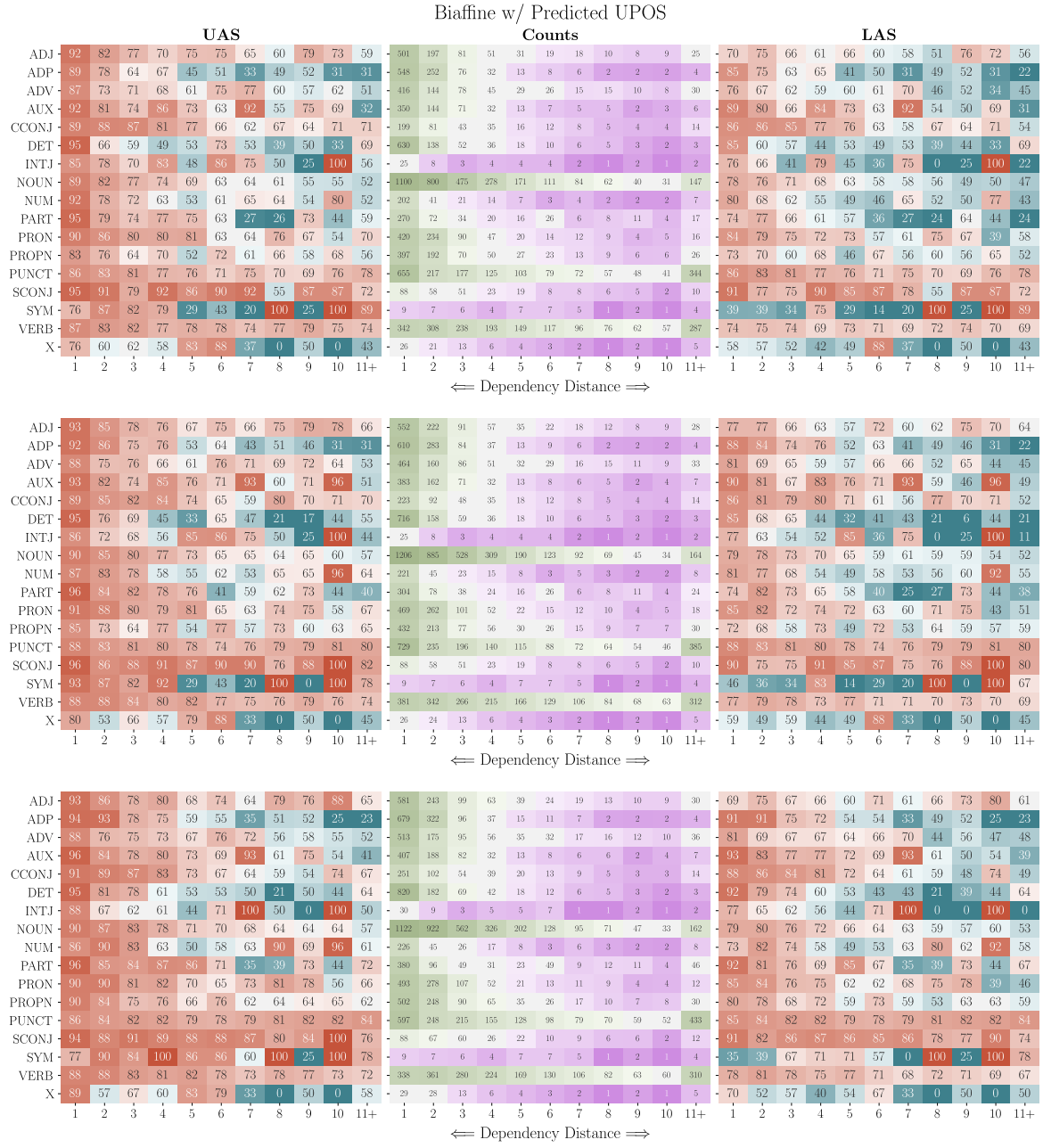


Figure 16: Average UAS (left column) and LAS (right column) across treebanks with Biaffine for models trained with predicted tags from taggers with 80, 86, and max POS accuracy of 91(3). UAS and LAS metrics are shown for each gold tag (y-axis) with a given dependency distance (x-axis). The counts of each pair of POS tag and dependency distance are shown in the middle column.

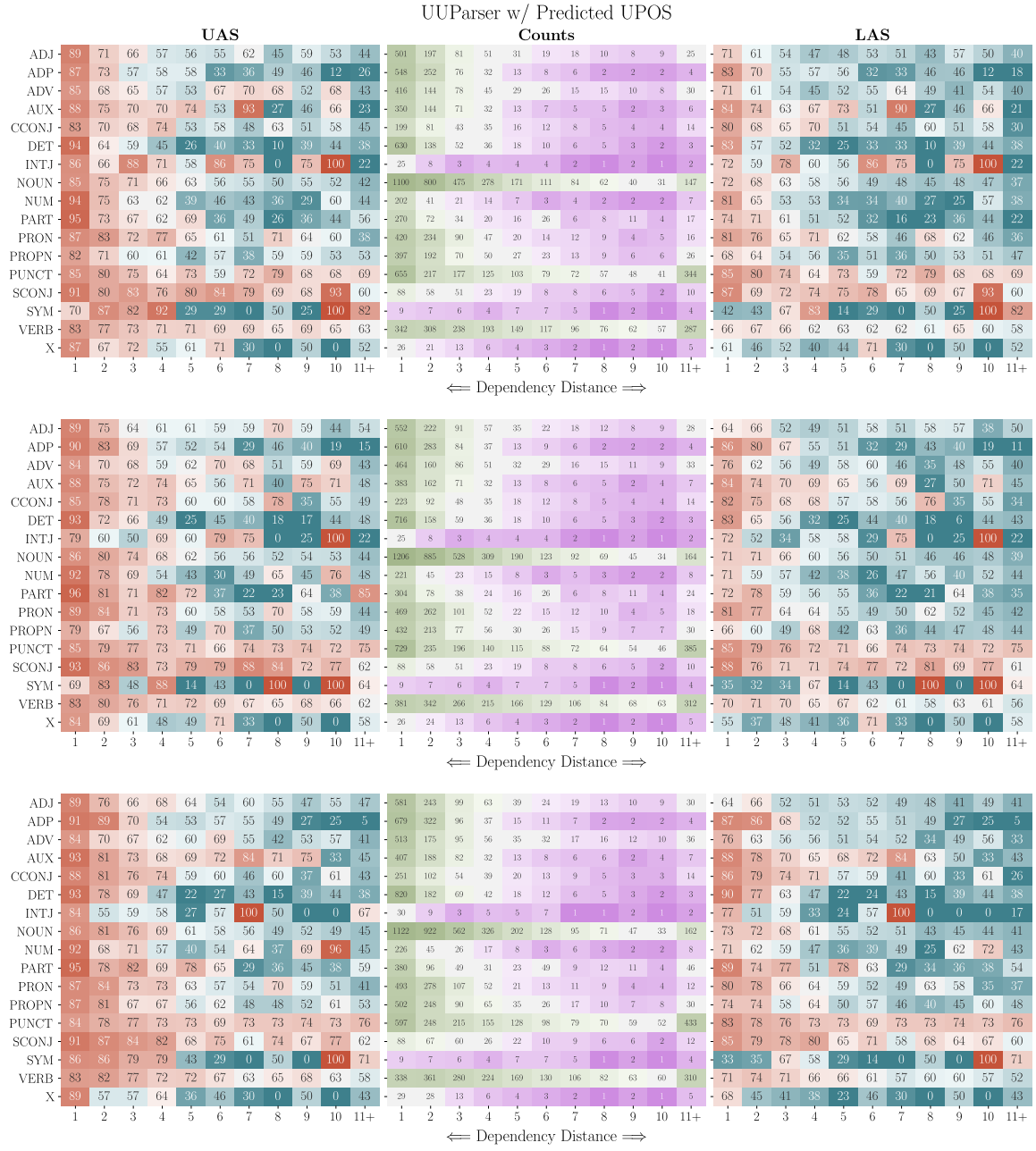


Figure 17: Average UAS (left column) and LAS (right column) across treebanks with UUParser for models trained with predicted tags from taggers with 80, 86, and max POS accuracy of 91(3). UAS and LAS metrics are shown for each gold tag (y-axis) with a given dependency distance (x-axis). The counts of each pair of POS tag and dependency distance are shown in the middle column.

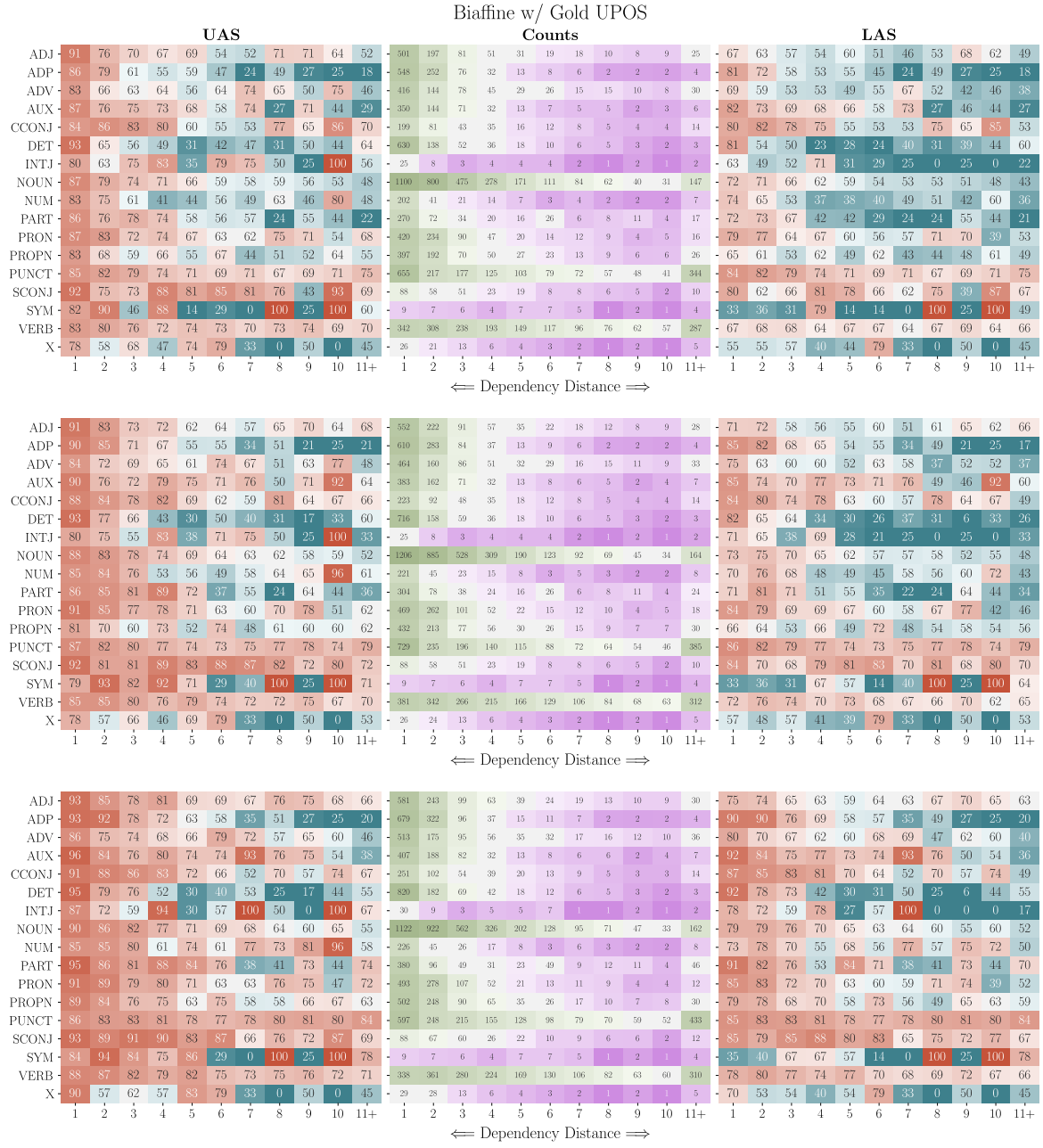


Figure 18: Average UAS (left column) and LAS (right column) across treebanks with Biaffine for models trained with gold tags but using predicted tags from taggers with 80, 86, and max POS accuracy of 91(3). UAS and LAS metrics are shown for each gold tag (y-axis) with a given dependency distance (x-axis). The counts of each pair of POS tag and dependency distance are shown in the middle column.

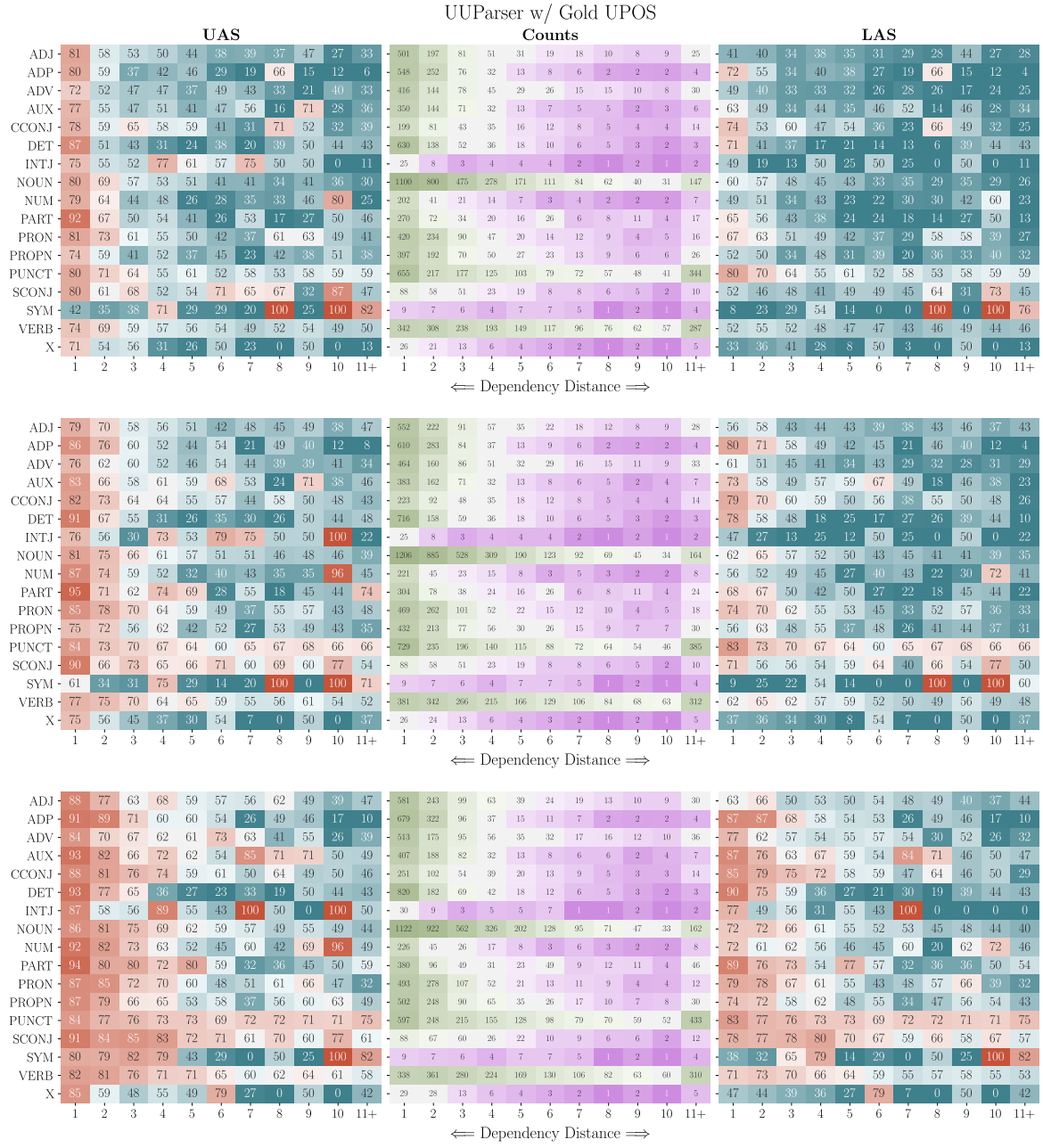


Figure 19: Average UAS (left column) and LAS (right column) across treebanks with UUParser for models trained with gold tags but using predicted tags from taggers with 80, 86, and max POS accuracy of 91(3). UAS and LAS metrics are shown for each gold tag (y-axis) with a given dependency distance (x-axis). The counts of each pair of POS tag and dependency distance are shown in the middle column.

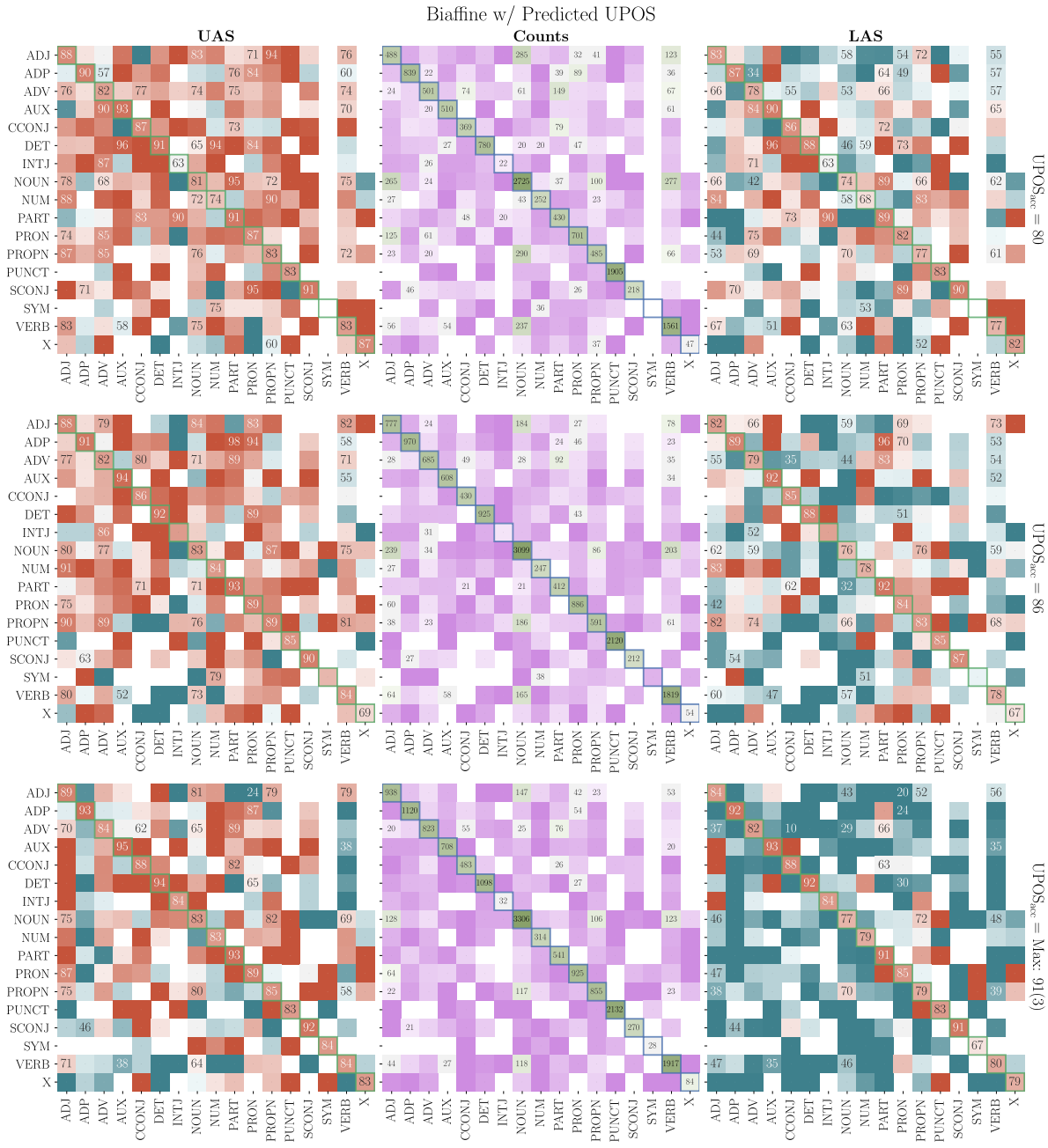


Figure 20: Average UAS (left column) and LAS (right column) across treebanks with Biaffine for models trained with POS tags from taggers of 80, 86, and max POS accuracy of 91(3). UAS and LAS metrics are shown for each gold tag (y-axis) predicted as any other tag (x-axis). The numbers are annotated when the average count (shown in the centre column) of a particular error is greater than 20.

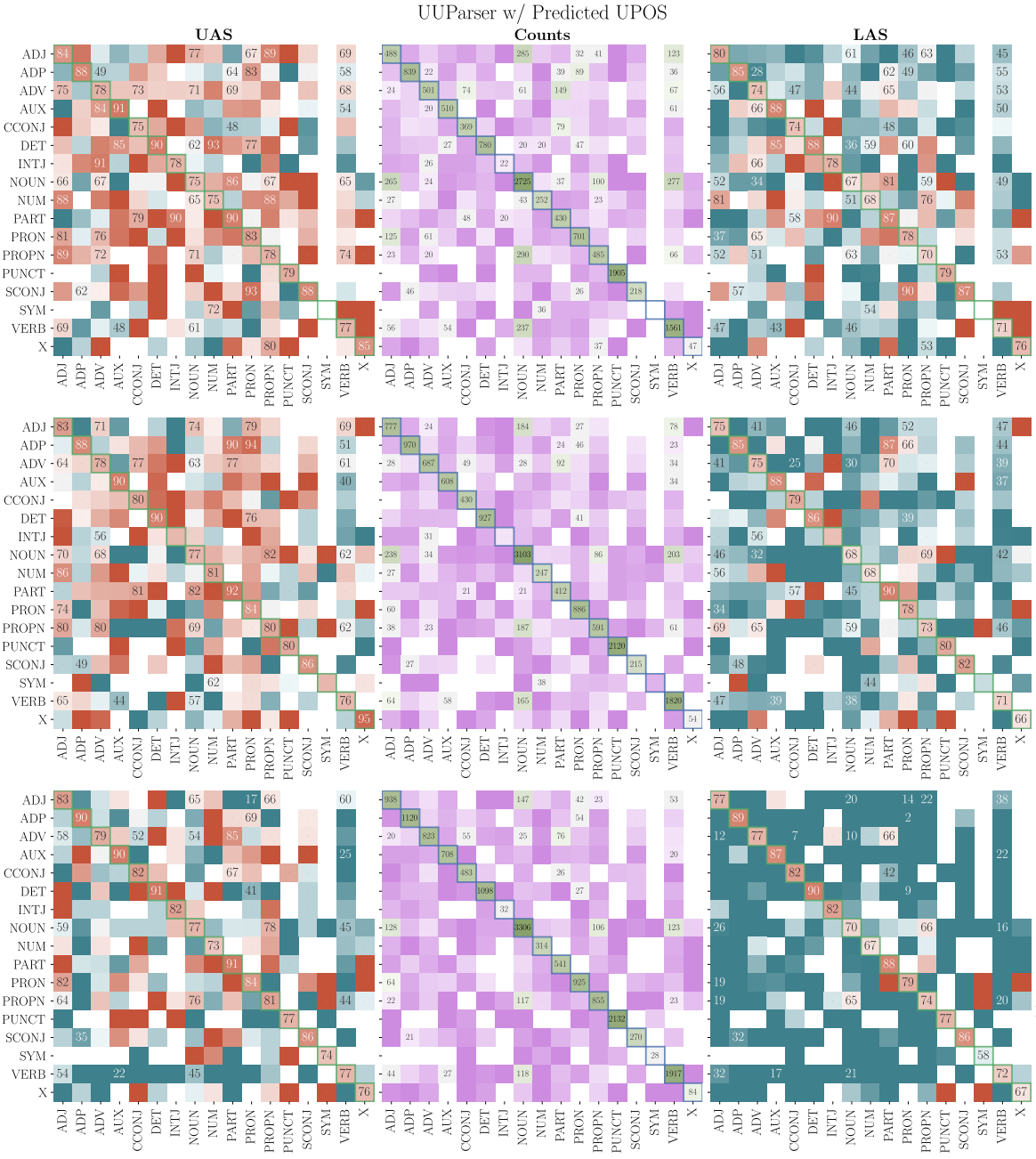


Figure 21: Average UAS (left column) and LAS (right column) across treebanks with UUParser for models trained with POS tags from taggers of 80, 86, and max POS accuracy of 91(3). UAS and LAS metrics are shown for each gold tag (y-axis) predicted as any other tag (x-axis). The numbers are annotated when the average count (shown in the centre column) of a particular error is greater than 20.

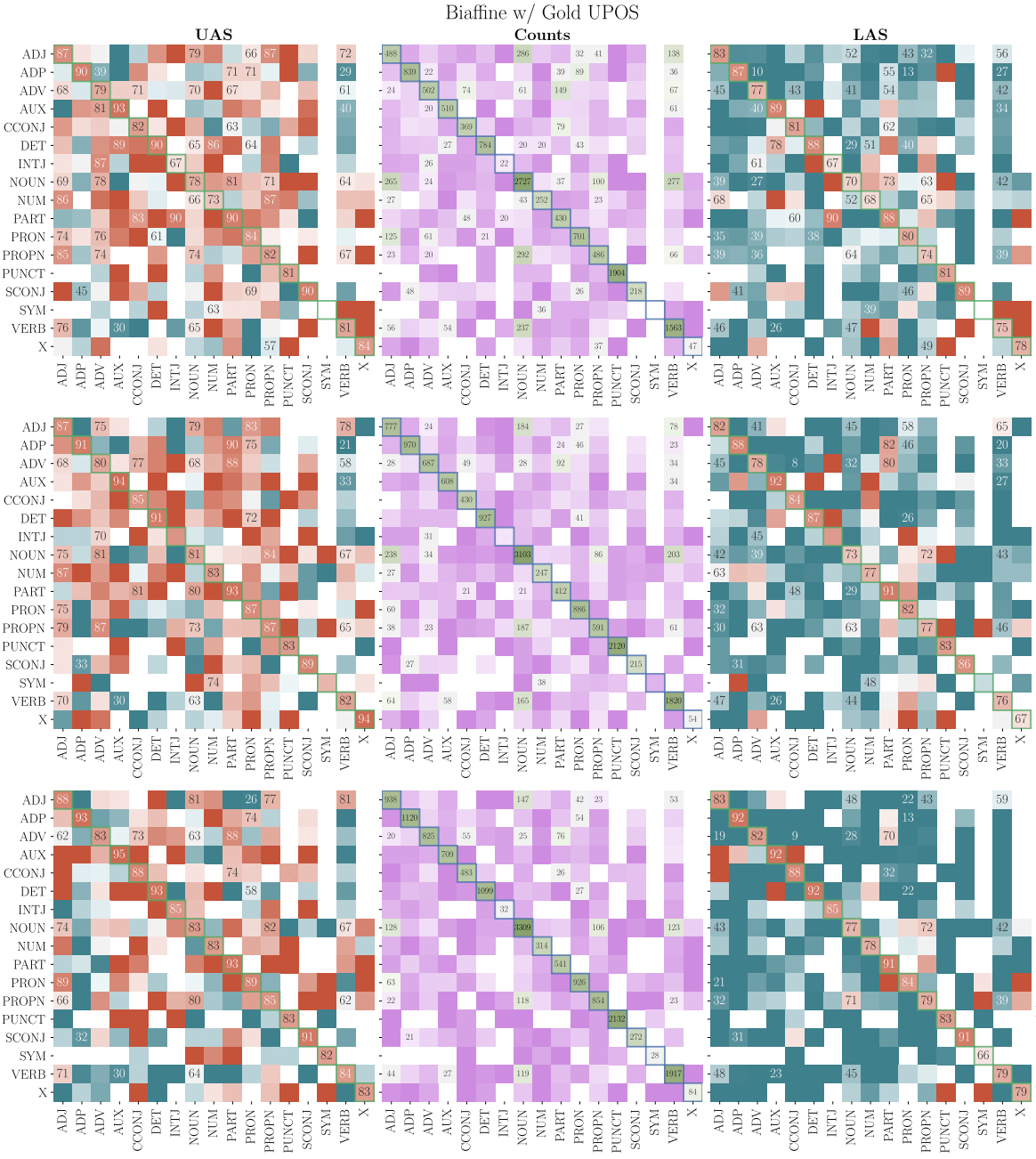


Figure 22: Average UAS (left column) and LAS (right column) across treebanks with Biaffine for models trained with gold tags but using predicted tags from taggers with 80, 86, and max POS accuracy of 91(3). UAS and LAS metrics are shown for each gold tag (y-axis) predicted as any other tag (x-axis). The numbers are annotated when the average count (shown in the centre column) of a particular error is greater than 20.

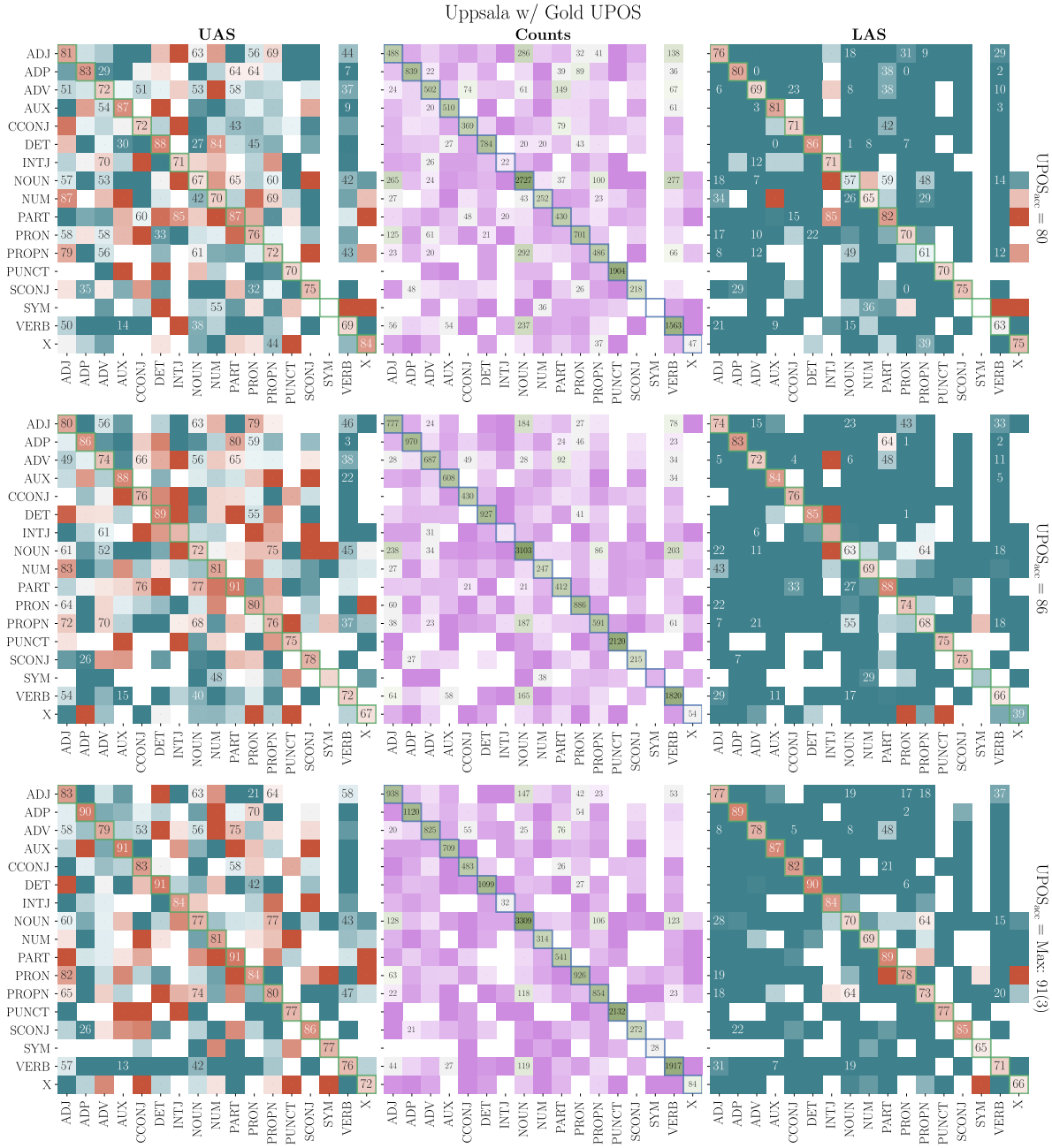


Figure 23: Average UAS (left column) and LAS (right column) across treebanks with UUParser for models trained with gold tags but using predicted tags from taggers with 80, 86, and max POS accuracy of 91(3). UAS and LAS metrics are shown for each gold tag (y-axis) predicted as any other tag (x-axis). The numbers are annotated when the average count (shown in the centre column) of a particular error is greater than 20.

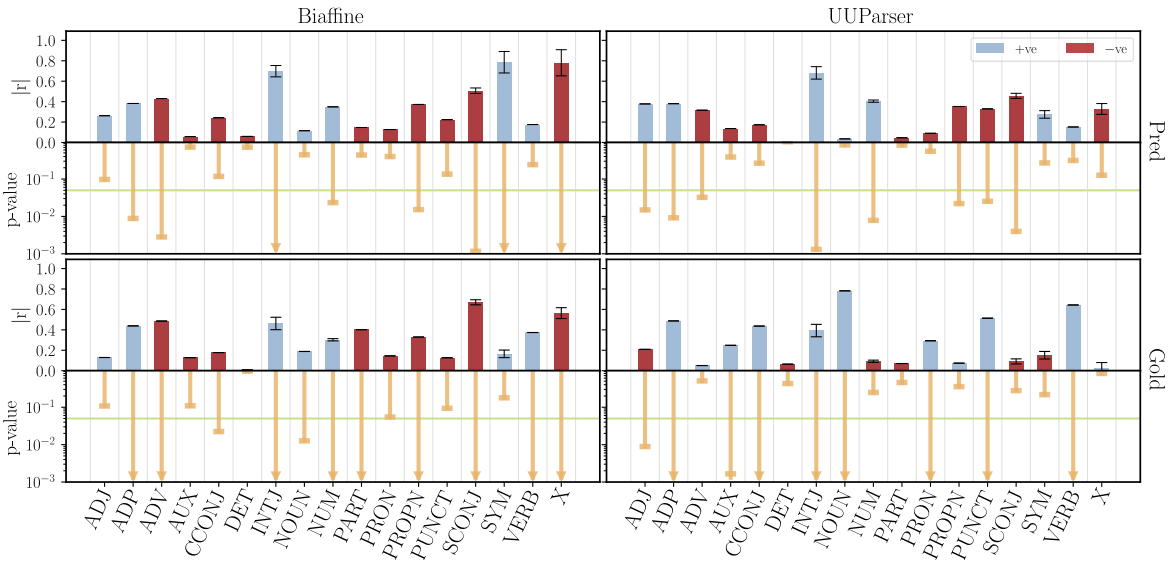


Figure 24: Pearson coefficients for the F1-score of the *head* of separate POS tags and global LAS where positive (+ve) coefficients are shown in blue and negative (-ve) are shown in red. The corresponding p-values are shown below (orange) where an arrow head means the value was below 0.001. Left subplots are for Biaffine parsers, right for UUParsers, top row is for parsers trained with predicted tags, and bottom for parsers trained with gold tags.

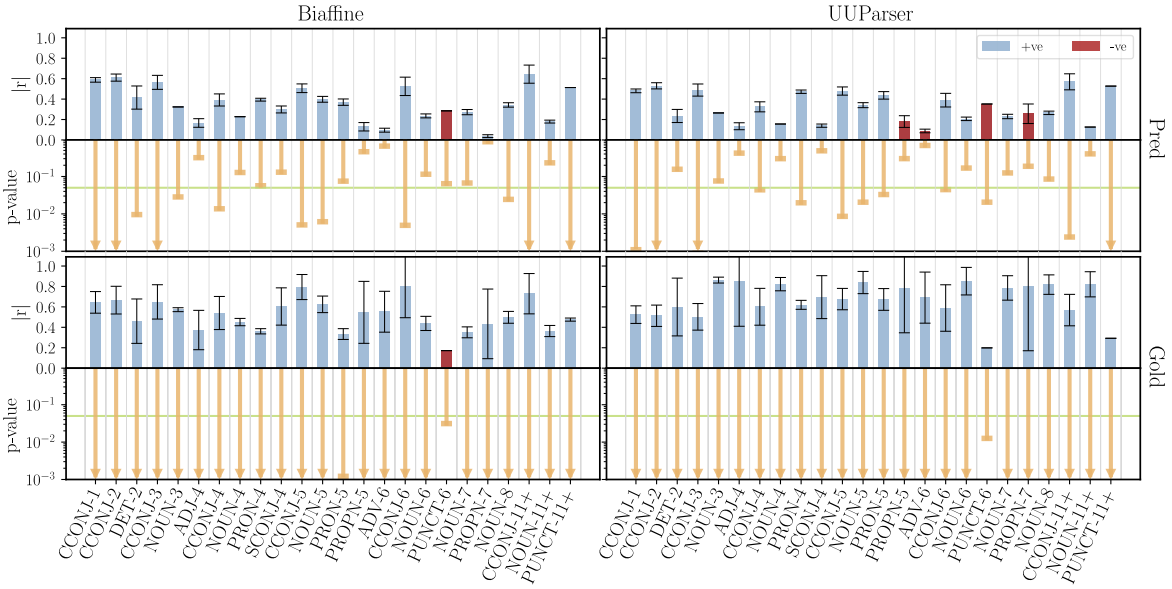


Figure 25: Pearson coefficients for the F1-score for individual POS tags with different dependency distances (`POS`-distance) and global LAS where positive (+ve) coefficients are shown in blue and negative (-ve) are shown in red. The corresponding p-values are shown below (orange) where an arrow head means the value was below 0.001. Left subplots are for Biaffine parsers, right for UUParsers, top row is for parsers trained with predicted tags, and bottom for parsers trained with gold tags.