An Investigation of the Interactions Between Pre-Trained Word Embeddings, Character Models and POS Tags in Dependency Parsing

Aaron Smith **Miryam de Lhoneux** **Sara Stymne** **Joakim Nivre**
Department of Linguistics and Philology, Uppsala University

Abstract

We provide a comprehensive analysis of the interactions between pre-trained word embeddings, character models and POS tags in a transition-based dependency parser. While previous studies have shown POS information to be less important in the presence of character models, we show that in fact there are complex interactions between all three techniques. In isolation each produces large improvements over a baseline system using randomly initialised word embeddings only, but combining them quickly leads to diminishing returns. We categorise words by frequency, POS tag and language in order to systematically investigate how each of the techniques affects parsing quality. For many word categories, applying any two of the three techniques is almost as good as the full combined system. Character models tend to be more important for low-frequency open-class words, especially in morphologically rich languages, while POS tags can help disambiguate high-frequency function words. We also show that large character embedding sizes help even for languages with small character sets, especially in morphologically rich languages.

1 Introduction

The last few years of research in natural language processing (NLP) have witnessed an explosion in the application of neural networks and word embeddings. In tasks ranging from POS tagging to reading comprehension to machine translation, a unique dense vector is learned for each word type in the training data. These word embeddings have been shown to capture essential semantic and morphological relationships between words (Mikolov et al., 2013), and have precipitated the enormous success of neural network-based architectures across a wide variety of NLP tasks (Plank et al., 2016; Dhingra et al., 2017b; Vaswani et al., 2017).

When task-specific training data is scarce or the morphological complexity of a language leads to sparsity at the word-type level, word embeddings often need to be augmented with sub-word or part-of-speech (POS) tag information in order to release their full power (Kim et al., 2016; Sennrich et al., 2016; Chen and Manning, 2014). Initialising vectors with embeddings trained for a different task, typically language modelling, on huge unlabelled corpora has also been shown to improve results significantly (Dhingra et al., 2017a). In dependency parsing, the use of character (Ballesteros et al., 2015) and POS (Dyer et al., 2015) models is widespread, and the majority of parsers make use of pre-trained word embeddings (Zeman et al., 2017).

While previous research has examined in detail the benefits of character and POS models in dependency parsing and their interactions (Ballesteros et al., 2015; Dozat et al., 2017), there has been no systematic investigation into the way these techniques combine with the use of pre-trained embeddings. Our results suggest a large amount of redundancy between all three techniques: in isolation, each gives large improvements over a simple baseline model, but these improvements are not additive. In fact combining any two of the three methods gives similar results, close to the performance of the fully combined system.

We set out to systematically investigate the ways in which pre-trained embeddings, character and POS models contribute to improving parser quality. We break down results along three dimensions—word frequency, POS tag, and language—in order to tease out the complex interactions between the three techniques. Our main findings can be summarized as follows:

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2711–2720
Brussels, Belgium, October 31 - November 4, 2018. ©2018 Association for Computational Linguistics

- For all techniques, improvements are largest for low-frequency and open-class words and for morphologically rich languages.
- These improvements are largely redundant when the techniques are used together.
- Character-based models are the most effective technique for low-frequency words.
- Part-of-speech tags are potentially very effective for high-frequency function words, but current state-of-the-art taggers are not accurate enough to take full advantage of this.
- Large character embeddings are helpful for morphologically rich languages, regardless of character set size.

2 Related Work

Chen and Manning (2014) introduced POS tag embeddings: a learned dense representation of each tag designed to exploit semantic similarities between tags. In their greedy transition-based parser, the inclusion of these POS tag embeddings improved labelled attachment score (LAS) by 1.7 on the English Penn Treebank (ETB) and almost 10 on the Chinese Penn Treebank (CTB). They also tested the use of pre-trained word embeddings for initialisation of word vectors, finding gains of 0.7 for PTB and 1.7 for CTB.

Dyer et al. (2015) in their Stack Long Short-Term Memory (LSTM) dependency parser, show that POS tag embeddings in their architecture improve LAS by 0.6 for English and 6.6 for Chinese. Unlike Chen and Manning (2014), they do not use pre-trained word embeddings for initialisation, instead concatenating them as a fixed vector representation to a separate randomly-initialised learned representation. This leads to improvements in LAS of 0.9 and 1.6 of English and Chinese, respectively.

Following on from the work of Dyer et al. (2015), Ballesteros et al. (2015) introduced the first character-based parsing model. They found that a model based purely on character information performed at the same level as a model using a combination of word embeddings and POS tags. Combining character and POS models produced even better results, but they conclude that POS tags are less important for character-based parsers. They also showed that character models are particularly effective for morphologically rich languages, but that performance remains good in languages with little morphology, and that character models help substantially with out-of-vocabulary (OOV) words, but that this does not fully explain the improvements they bring. The use of pre-trained embeddings was not considered in their work.

Kiperwasser and Goldberg (2016), in the transition-based version of their parser based on BiLSTM feature extractors, found that POS tags improved performance by 0.3 LAS for English and 4.4 LAS for Chinese. Like Dyer et al. (2015), they concatenate a randomly-initialised word embeddings to a pre-trained word vector; however in this case the pre-trained vector is also updated during training. They find that this helps LAS by 0.5–0.7 for English and 0.9–1.2 for Chinese, depending on the specific architecture of their system.

Dozat et al. (2017), building on the graph-based version of Kiperwasser and Goldberg (2016), confirmed the relationship between character models and morphological complexity, both for POS tagging and parsing. They also examined the importance of the quality of POS tags on parsing, showing that their own tagger led to better parsing results than a baseline provided by UDPipe v1.1 (Straka et al., 2016).

3 The Parser

We use and extend UUParser[1] (de Lhoneux et al., 2017a; Smith et al., 2018), a variation of the transition-based parser of Kiperwasser and Goldberg (2016) (K&G). The K&G architecture can be adapted to both transition- and graph-based dependency parsing, and has quickly become a *de facto* standard in the field (Zeman et al., 2017). In a K&G parser, BiLSTMs (Hochreiter and Schmidhuber, 1997; Graves, 2008) are employed to learn useful representations of tokens in context. A multi-layer perceptron (MLP) is trained to predict transitions and possible arc labels, taking as input the BiLSTM vectors of a few tokens at a time. Crucially, the BiLSTMs and MLP are trained together, enabling the parser to learn very effective token representations for parsing. For further details we refer the reader to Nivre (2008) and Kiperwasser and Goldberg (2016), for transition-based parsing and BiLSTM feature extractors, respectively.

Our version of the K&G parser is extended with a SWAP transition to facilitate the construction

[1] https://github.com/UppsalaNLP/uuparser

of non-projective dependency trees (Nivre, 2009). We use a static-dynamic oracle to allow the parser to learn from non-optimal configurations at training time in order to recover better from mistakes at test time, as described in de Lhoneux et al. (2017b).

In this paper we experiment with a total of eight variations of the parser, where the difference between each version resides in the vector representations x_i of word types w_i before they are passed to the BiLSTM feature extractors (see Section 3 of Kiperwasser and Goldberg (2016)). In the simplest case, we set x_i equal to the word embedding $e^r(w_i)$:

$$x_i = e^r(w_i)$$

The superscript r refers to the fact that the word embeddings are initialised randomly at training time. This is the setup in our BASELINE system.

For our +CHAR system, the word embedding $e^r(w_i)$ is concatenated to a character-based vector, obtained by running a BiLSTM over the characters $ch_{1:m}$ of w_i:

$$x_i = e^r(w_i) \circ \text{BiLSTM}(ch_{1:m})$$

In the +POS setting, the word embedding is instead concatenated to an embedding $p(w_i)$ of the word's universal POS tag (Nivre et al., 2016):

$$x_i = e^r(w_i) \circ p(w_i)$$

This scenario necessitates knowledge of the POS tag of w_i; at test time, we therefore need a POS tagger to provide predicted tags.

In another version of our parser (+EXT), pre-trained embeddings are used to initialise the word embeddings.[2] We use the superscript t to distinguish these from randomly initialised vectors:

$$x_i = e^t(w_i)$$

We use the embeddings that were released as part of the 2017 CoNLL Shared Task on Universal Dependency Parsing (CoNLL-ST-17) (Zeman et al., 2017). Words in the training data that do not have pre-trained embeddings are initialised randomly. At test time, we look up the updated embeddings for all words seen in the training data; OOV words are assigned their un-updated pre-trained embedding where it exists, otherwise a learnt OOV vector.

In our COMBINED setup, we include pre-trained embeddings along with the character vector and POS tag embedding:

$$x_i = e^t(w_i) \circ \text{BiLSTM}(ch_{1:m}) \circ p(w_i)$$

The three remaining versions of the vector x_i constitute all possible combinations of two techniques of pre-trained embeddings, the character model and POS tags. We refer to these versions of the parser as −EXT, −CHAR, and −POS, respectively.

4 Experimental setup

4.1 Data

We ran our experiments on nine treebanks from Universal Dependencies (Nivre et al., 2016) (v2.0): Ancient Greek PROIEL, Arabic, Chinese, English, Finnish, Hebrew, Korean, Russian and Swedish. Inspired partially by de Lhoneux et al. (2017c), these treebanks were chosen to reflect a diversity of writing systems, character set sizes, and morphological complexity. As error analysis is carried out on the results, we perform all experiments on the *dev* data sets.

Table 1 shows some statistics of each treebank. Of particular note are the large character set sizes in Chinese and Korean, an order of magnitude bigger than those of all other treebanks. The high type-token ratio for Finnish, Russian and Korean also stands out; this is likely due to the high morphological complexity of these languages.

Treebank	Sentences		TTR	Chars
Ancient Greek	14864	1019	0.15	179
Arabic	6075	909	0.10	105
Chinese	3997	500	0.16	3571
English	12534	2002	0.07	108
Finnish	12217	1364	0.26	244
Hebrew	5241	484	0.11	53
Korean	4400	950	0.46	1730
Russian	3850	579	0.30	189
Swedish	4303	504	0.16	86

Table 1: Treebank statistics. Number of sentences in *train* and *dev* sets, type-token ratio (TTR), and character set size.

4.2 Parser settings

The parser is trained three times for each language with different random seeds for 30 epochs each. At the end of each epoch we parse the *dev* data

[2] This strategy proved more successful in preliminary experiments than others for incorporating pre-trained embeddings discussed in Section 2.

Word embedding size	100
Character embedding size	500
Character BiLSTM output size	100
POS tag embedding size	20

Table 2: Embedding sizes.

and calculate LAS. For each training run, results are averaged over the five best epochs for each language. In this way, we attempt to make our results more robust to variance due to randomness in the training procedure.[3] Our macro-averaged scores are based on a total of 135 different epochs (3 random seeds × 5 best epochs × 9 languages).

Table 2 shows the embedding sizes we found to produce best results in preliminary work and which we use in all experiments in Section 5. Note our unusually large character embedding size; we will discuss this in more detail in Section 6. We use predicted UPOS tags from the system of Dozat et al. (2017) for experiments with POS tags,[4] other than in Section 7 where we compare results with different taggers and gold POS tags, in order to set a ceiling on the potential gains from a perfect POS tagger. For all other hyperparameters we use default values (Smith et al., 2018).

4.3 Analysis

The hypothesis underlying our choice of analysis is that the three techniques under study here—pretrained embeddings, character vectors and POS tag embeddings—affect words differently depending on their frequencies, POS tags, and the language of the sentence. We do not claim this to be an exhaustive list; many other dimensions of analysis are clearly possible (dependency relation would be another obvious choice for example), but we believe that these are likely to be three of the most informative factors. In the frequency and POS tag cases, we want to examine the overall contribution to LAS of words from each category. We expect changing the representation of a token to affect how likely it is to be assigned the correct head in the dependency tree, but also how likely it is to be assigned correctly as the head of other words. We thus introduce a new metric for this part of the analysis: the head and dependents labelled attachment score, which we refer to as HDLAS.

When calculating HDLAS, the dependency analysis for a given token is only considered correct if the token has the correct labelled head *and* the complete set of correctly labelled dependents. This is a harsher metric than LAS, which only considers whether a token has the correct labelled head. Note that when calculating HDLAS for all tokens in a sentence, each dependency relation is counted twice, once for the head word and once for the dependent. It only makes sense to use this metric when analysing individual tokens in a sentence, or when grouping tokens into different categories across multiple sentences.

4.3.1 Frequency

In this analysis, we first label each token in the *dev* data for each language by its relative frequency in the *train* data, with add-one smoothing.[5] Frequency categories are created by rounding the log relative frequency down to the nearest integer. We calculate the HDLAS for each frequency category for each language, before macro-averaging the results across the nine languages to produce a final score for each frequency class.

4.3.2 POS tag

In this case, we label each word from the *dev* data by its gold POS tag, before calculating HDLAS for each category and taking the macro average across languages. Here the total number of tokens in each category varies across several orders of magnitude: the most common category NOUNs make up 26.0% of all words, while the smallest class SYM represents just 0.1%. For this reason, and to make our graphs more readable, we do not show results for the six smallest categories: INTJ, NUM, PART, SCONJ, SYM, and X.

4.3.3 Language

Here we consider LAS directly for each language; the HDLAS metric used in the previous two sections is not relevant as all tokens in a given sentence are assigned to the same category determined by the language of the sentence.

5 Results

Table 3 gives the LAS for each of the eight systems described in Section 3. We observe that pre-

[3] Changing the random seed has been shown to produce results that appear statistically significant different in neural systems (Reimers and Gurevych, 2017).

[4] Available at https://web.stanford.edu/~tdozat/.

[5] The smoothing ensures that OOV tokens, those that appear in *dev* but not *train*, are not assigned zero frequency; this alleviates the problem of taking log(0) in the subsequent conversion to log relative frequency.

BASELINE	67.7	COMBINED	81.0
+EXT	76.1	−EXT	79.9
+CHAR	78.3	−CHAR	79.2
+POS	75.9	−POS	80.3

Table 3: Mean LAS across nine languages for a baseline system employing randomly-initialised word embeddings only, compared to three separate systems using pre-trained word embeddings (+EXT), a character model (+CHAR), and POS tags (+POS). Scores are also shown for a combined system that utilises all three techniques and corresponding systems where one of the three techniques is ablated (−EXT, −CHAR and −POS).

trained embeddings (+8.4), the character model (+10.6) and POS tags (+8.2) all give large improvements in LAS over the baseline system. The combined system is the best overall, but the improvement of 13.3 LAS is far from the sum of its components. Employing two of the techniques at a time reduces LAS by only 0.7–1.8 compared to the combined system.

5.1 Frequency

Fig. 1 and Fig. 2 compare systems by word frequency. As expected, accuracy improves with frequency for all systems: the parser does better with words it sees more often during training. There is a levelling off for the highest frequency words, probably due to the fact that these categories contain a small number of highly polysemous word types.

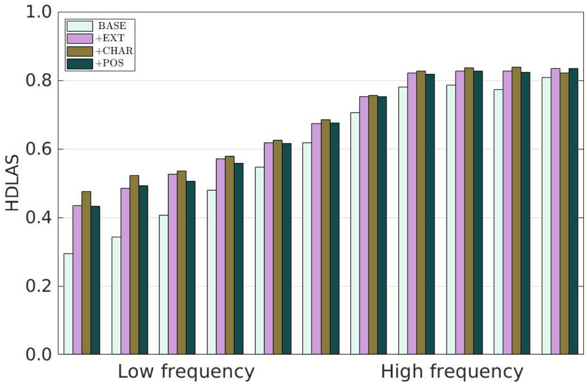

Figure 1: BASELINE system compared to pre-trained embeddings (+EXT), character model (+CHAR) and POS tags (+POS).

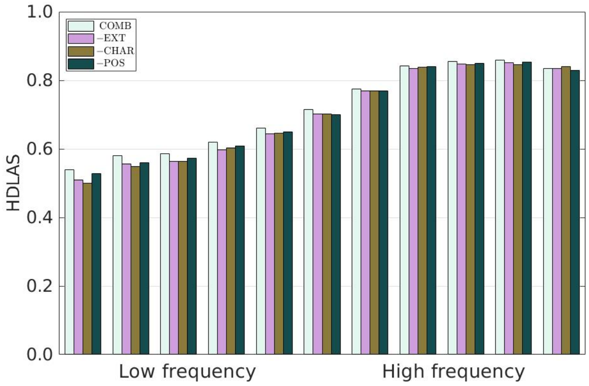

Figure 2: COMBINED system compared to ablated systems where pre-trained embeddings (−EXT), character models (−CHAR) and POS tags (−POS) are removed.

Fig. 1 demonstrates a clear trend in the improvement achieved by each of the individual techniques over the baseline, with larger gains for lower frequency words. This confirms a result from Ballesteros et al. (2015), who found that character models help substantially with OOV words. We can generalise this to say that character models improve parsing quality most for low frequency words (including OOV words), and that this is also true, albeit to a slightly lesser effect, of POS tags and pre-trained word embeddings. It is notable however that HDLAS increases universally across all frequency classes: even the highest frequency words benefit from enhancements to the basic word representation.

What immediately stands out in Fig. 2 is that for mid- and high frequency words, there is little difference in HDLAS between different combinations of two of the three techniques, and for the highest frequency words this is at a level almost indistinguishable from the full COMBINED system. The slight improvements we see for COMBINED in Table 3 compared to the three ablated systems thus principally also come from the low-frequency range.

5.2 POS tags

In Fig. 3 systems are compared by POS tag. We observe a universal improvement across all POS tags for each of the three variations of the system compared to the baseline. However, it is notable that the biggest gains in HDLAS are for open word classes: NOUNs, VERBs and ADJs. As these make up a large overall proportion of words, these differences have an overall relatively large impact on LAS.

For the most frequent POS categories NOUN and VERB we again see a clear victory for the character model (note that while these POS categories are frequent, they contain a large number of low-frequency words). Overall the character model succeeds best for the open-class POS categories, while having the right POS tag is marginally better for closed-class categories such as DET, CCONJ, and AUX. It is interesting that the character model is not as strong for PROPN, despite the fact that these are open-class low-frequency words; for these words pre-trained embeddings are the best single technique. This may be due to the fact that the rules governing the composition of names at the character level are different from other words in the language.

It is perhaps surprising that the advantage of POS tag embeddings is not greater when it comes to auxiliary verbs, for example, where the distinction from main verbs can be difficult and crucial for a correct syntactic analysis. The reason probably lies in the fact that this distinction is equally difficult for the POS tagger. We will investigate this further in Section 7.

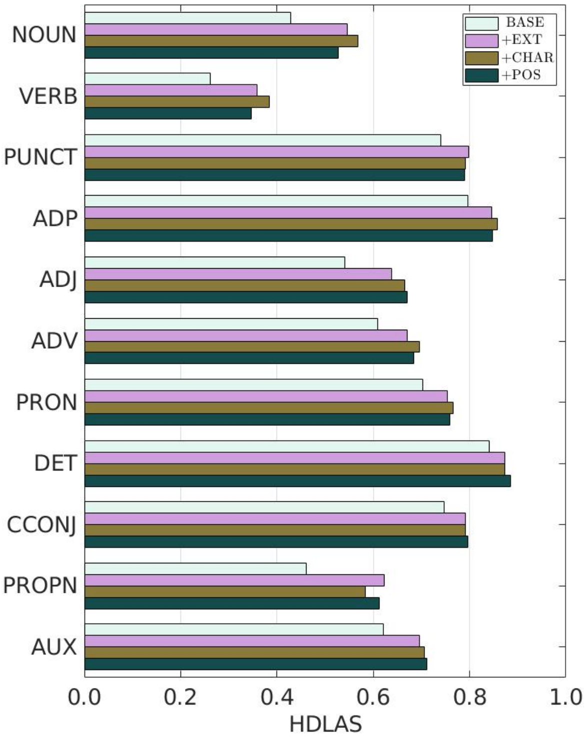

Figure 3: Comparison by POS tag of BASELINE system to +EXT, +CHAR, and +POS. Tags are sorted by frequency.

5.3 Language

Fig. 4 compares the systems by language. Once again improvement is universal for each system compared to the baseline. There are however substantial differences between languages. The three biggest overall improvements are for Finnish, Korean and Russian, with a particularly notable increase in the Korean case. This suggests that the baseline model struggles to learn adequate representations for each word type in these languages. These are the three languages we identified in Section 4.1 as having high type-token ratios in their training data. It is also notable that the character model becomes more important compared to other methods for these three languages. In fact, despite the overall superiority of the character model (see Table 3), it is only the best single technique for 4 of the 9 languages, the three already mentioned plus Ancient Greek.

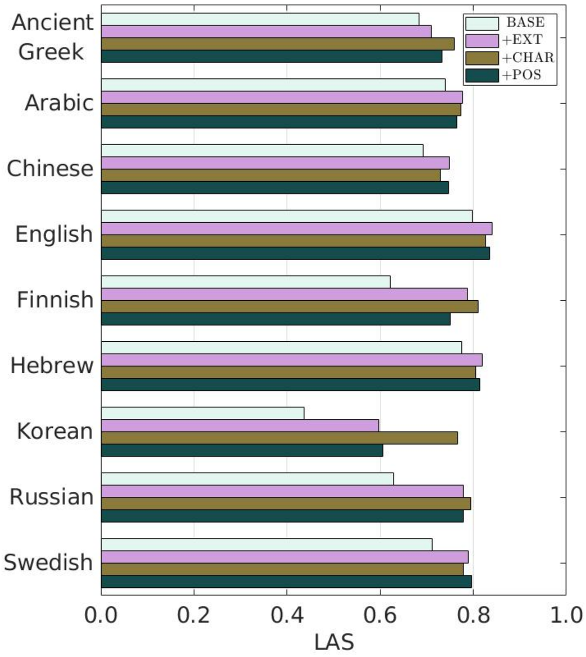

Figure 4: Comparison by language of BASELINE system to +EXT, +CHAR, and +POS.

6 Character Embedding Size

All results with character models observed thus far make use of a character embedding of dimension 500. This value is large compared to typical sizes used for character models (Kim et al., 2016; Ballesteros et al., 2015). A common belief is that larger character embedding sizes are justified for languages with larger character set sizes such as Chinese: in other words, the embedding size should be related to the number of entities being embedded (Shao, 2018).

In Table 4, we show how LAS varies with a few values of this hyperparameter when averaged across our nine-language sample. We see a steady

BASELINE	67.7	−CHAR	79.2
+CH-24	76.8	+CH-24	80.5
+CH-100	77.7	+CH-100	80.6
+CH-500	78.3	+CH-500	81.0

Table 4: Mean LAS across nine languages for BASELINE system compared to systems with character vectors of different sizes. Comparison also shown for systems employing pre-trained word vectors and POS tag embeddings.

improvement in LAS as the character embedding size increases, both when compared to a baseline with randomly initialised word embeddings only and when compared to a system that also employs pre-trained word vectors and POS tag embeddings.[6]

It is particularly interesting to break down the effects here by language. In Table 5 we show results for Chinese, Finnish, Korean and Russian. It is particularly striking that the larger character embeddings do not help for Chinese; the score for the largest character embedding size is actually marginally lower than a baseline without a character model at all. This is despite the fact that a small character embedding improves LAS, albeit marginally, suggesting that there is some useful information in the characters even when pre-trained embeddings and POS tags are present. Conversely, the large character models are very effective for Finnish, a treebank with a character set less than a tenth of the size of Chinese (see Table 1).

	−CHAR	+CH-24	+CH-100	+CH-500
Chinese	76.0	76.1	75.9	75.8
Finnish	81.9	83.7	83.8	84.7
Korean	70.1	78.0	78.2	79.4
Russian	82.0	81.4	81.5	82.5

Table 5: Comparison by language of different character embedding sizes.

We claim therefore that character set size is not in fact a good metric to use in determining character embedding sizes. Our tentative explanation is that while languages like Finnish have relatively small character sets, those characters interact with each other in much more complex ways, thus requiring larger embeddings to store all the necessary information. While there are many characters in Chinese, the entropy in the interactions between characters appears to be smaller, enabling smaller character embeddings to do just as good a job.

It is also worth noting from Tables 4 and 5 that, in the presence of POS tags and pre-trained embeddings, the improvement gained from increasing the character embedding size from 24 to 100 is small (0.1 LAS for Finnish, 0.2 for Korean, 0.1 for Russian; 0.1 on average across the nine treebanks). This perhaps gives the impression of diminishing returns; that going even larger is likely to lead to ever smaller improvements. This may be the reason that smaller character embeddings have generally been preferred previously. However, we in fact observe a much *greater* gain when increasing from 100 to 500 (0.9 for Finnish, 1.2 for Korean, 1.0 for Russian; 0.4 on average across the nine treebanks), suggesting that very large character embeddings are effective, and particularly useful for morphologically rich languages.

7 POS tagger

In this section we apply our POS tag analysis to the effect of the POS tagger used to produce tags at test time. We compare three setups: firstly using tags predicted by UDPipe (Straka and Straková, 2017), which was the baseline model for CoNLL-ST-2017, secondly using tags predicted by the winning Stanford system (Dozat et al., 2017), and thirdly using gold tags. Note that for the Stanford system, we train on gold tags and use predicted tags at test time, while for UDPipe we train on a jackknifed version of the train data with predicted tags that was released as part of CoNLL-ST-2017.

BASELINE	67.7	−POS	80.3
UDPipe	73.4	UDPipe	80.2
Stanford	75.9	Stanford	81.0
Gold	78.4	Gold	83.8

Table 6: Mean LAS across nine languages for BASELINE system compared to systems with POS tags predicted by different systems. Comparison also shown for systems employing pre-trained word vectors and a character vector.

Table 6 shows how LAS varies with the different POS taggers when averaged across the nine-language sample. We see a clear improvement

[6] Note that for character embeddings of dimension 24, we use an output size for the character BiLSTM of 50, for character embeddings of dimension 100, we use an output size of 75, and for character embeddings of dimension 500, we use an output size of 100. We checked in separate experiments that the improvements are not simply due to the increase in output size.

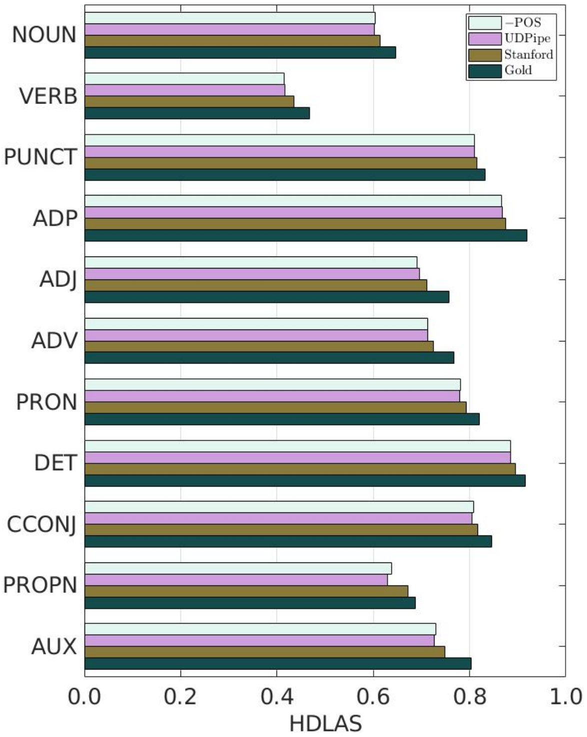

Figure 5: Comparison by POS tag of POS taggers.

from UDPipe to Stanford and then from Stanford to gold tags over the baseline system. This partially confirms results from Dozat et al. (2017), where the Stanford tagger was found to improve parsing results significantly over the UDPipe baseline. More surprising perhaps is the result when comparing to the −POS system, which also makes use of pre-trained word embeddings and a character model. Here, results do not improve at all by adding predicted tags from UDPipe. Stanford tags do give an improvement of 0.7 LAS over −POS, but this is a long way from the improvement of 8.2 LAS we see when adding them on top of BASELINE. Gold tags do however still give a big improvement over −POS (3.5 LAS), suggesting strongly that both UDPipe and Stanford struggle with the decisions that would be most beneficial to parsing accuracy.

In Fig. 5 we present the parsing results broken down by POS tag for the various POS taggers. It is particularly notable that results when tagging with UDPipe are no better than for −POS, which does not use POS tags at all, across most categories, and particularly for the closed-classes ADP, PRON, DET, CCONJ and AUX. Stanford tags do marginally better, but access to gold tags is particularly important in these cases; we see a particularly striking improvement when ADPs and AUXs are correctly tagged over an already strong baseline.

8 Parser speed

It should be noted that increasing the character embedding size and character BiLSTM output dimension as in Section 6 slows down the parser during training and at test time. We found no noticeable difference in speed between the baseline system and versions of the parser with smaller character embedding sizes (24/100), with approximately 20 sentences per second being processed on average during training and 65 sentences per second parsed at test time on the Taito super cluster.[7] There was however a discernible difference when the character embedding size was increased to 500, with only 12 sentences processed per second during training and 44 during testing.

Adding a POS tag embedding makes no appreciable difference to parser speed,[8] but necessitates a pipeline system that first predicts POS tags (assuming gold tags are unavailable). The application of pre-trained embeddings, meanwhile, requires expensive pre-training on large unlabelled corpora. Loading these embeddings into the parser takes time and can occupy large amounts of memory, but does not directly impact the time it takes to process a sentence during training or parsing.

9 Conclusions and Future Work

In this article we examined the complex interactions between pre-trained word vectors, character models and POS tags in neural transition-based dependency parsing. While previous work had shown that POS tags are not as important in the presence of character models, we extend that conclusion to say that in the presence of two of the three techniques, the third is never as important. The best system, however, is always a combination of all three techniques.

We introduced the HDLAS metric to capture the overall effect on parsing quality of changes to the representation of a particular word. We found that all three techniques produce substantial improvements across a range of frequency classes, POS tags, and languages, but the biggest improvements for all techniques were for low-frequency, open-class words. We suggest that this goes some way

[7] https://research.csc.fi/taito-supercluster

[8] Note that the POS tag embedding we use is small relative to the other components of the word type representation (see Table 2).

to explaining the redundancy between the three techniques: they target the same weaknesses in the baseline word-type level embedding.

We confirmed a previous result that the character model is particularly important for morphologically rich languages with high type-token ratios, and went on to show that these languages also benefit from larger character embedding sizes, whereas morphologically simpler languages make do with small character embeddings, even if the character set size is large.

POS tag embeddings can improve results for difficult closed-class categories, but our current best POS taggers are not capable of making the distinctions necessary to really take advantage of this. The strength of pre-trained embeddings is that they are trained on much larger corpora than the task-specific data; the use of character models and POS tag embeddings however seems to allow us to generalise much better from smaller data sets, as each character and each POS tag is normally seen many times, even if each word type is rare.

We saw that increasing the character embedding size slows the parser down; whether this tradeoff is worthwhile will depend on the application in question. If accuracy is all that matters, we recommend using a fully combined system with large character embeddings in tandem with POS tags and pre-trained embeddings. Where speed is more important, it may be worth considering a system that employs a smaller character embedding and does without POS tags, using just pre-trained embeddings.

In future work it would be interesting to investigate whether the patterns observed here also hold true for other types of models in dependency parsing; possible variations to examine include alternative character models such as convolutional neural networks, joint tagging-parsing models, and graph-based parsers.

Acknowledgments

We acknowledge the computational resources provided by CSC in Helsinki and Sigma2 in Oslo through NeIC-NLPL (www.nlpl.eu). Aaron Smith was supported by the Swedish Research Council. We would like to thank Sujoung Baeck for a valuable discussion regarding Korean morphology.

References

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.

Danqi Chen and Christopher Manning. 2014. A Fast and Accurate Dependency Parser using Neural Networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 740–750.

Bhuwan Dhingra, Hanxiao Liu, Ruslan Salakhutdinov, and William W Cohen. 2017a. A Comparative Study of Word Embeddings for Reading Comprehension. *arXiv preprint arXiv:1703.00993*.

Bhuwan Dhingra, Hanxiao Liu, Zhilin Yang, William Cohen, and Ruslan Salakhutdinov. 2017b. Gated-Attention Readers for Text Comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1832–1846.

Timothy Dozat, Peng Qi, and Christopher D Manning. 2017. Stanford's Graph-based Neural Dependency Parser at the CoNLL 2017 Shared Task. *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 20–30.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-Based Dependency Parsing with Stack Long Short-Term Memory. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 334–343.

Alex Graves. 2008. *Supervised Sequence Labelling with Recurrent Neural Networks*. Ph.D. thesis, Technical University Munich.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. 2016. Character-aware Neural Language Models. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pages 2741–2749.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and Accurate Dependency Parsing Using Bidirectional LSTM Feature Representations. *Transactions of the Association of Computational Linguistics*, 4:313–327.

Miryam de Lhoneux, Yan Shao, Ali Basirat, Eliyahu Kiperwasser, Sara Stymne, Yoav Goldberg, and

Joakim Nivre. 2017a. 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.

Miryam de Lhoneux, Sara Stymne, and Joakim Nivre. 2017b. Arc-Hybrid Non-Projective Dependency Parsing with a Static-Dynamic Oracle. In *Proceedings of the 15th International Conference on Parsing Technologies*, pages 99–104.

Miryam de Lhoneux, Sara Stymne, and Joakim Nivre. 2017c. Old School vs. New School: Comparing Transition-Based Parsers with and without Neural Network Enhancement. In *Proceedings of the 15th Treebanks and Linguistic Theories Workshop*, pages 99–110.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119.

Joakim Nivre. 2008. Algorithms for Deterministic Incremental Dependency Parsing. *Computational Linguistics*, 34(4):513–553.

Joakim Nivre. 2009. Non-Projective Dependency Parsing in Expected Linear Time. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 351–359.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajič, Christopher D. Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. 2016. Universal Dependencies v1: A Multilingual Treebank Collection. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*.

Barbara Plank, Anders Søgaard, and Yoav Goldberg. 2016. Multilingual Part-of-Speech Tagging with Bidirectional Long Short-Term Memory Models and Auxiliary Loss. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 412–418.

Nils Reimers and Iryna Gurevych. 2017. Reporting Score Distributions Makes a Difference: Performance Study of LSTM-networks for Sequence Tagging. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 338–348.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural Machine Translation of Rare Words with Subword Units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725.

Yan Shao. 2018. *Segmenting and Tagging Text with Neural Networks*. Ph.D. thesis, Uppsala University.

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*.

Milan Straka, Jan Hajic, and Jana Straková. 2016. UDPipe: Trainable Pipeline for Processing CoNLL-U Files Performing Tokenization, Morphological Analysis, POS Tagging and Parsing. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*.

Milan Straka and Jana Straková. 2017. Tokenizing, POS Tagging, Lemmatizing and Parsing UD 2.0 with UDPipe. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 88–99.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. In *Advances in Neural Information Processing Systems*, pages 6000–6010.

Daniel Zeman, Martin Popel, Milan Straka, et al. 2017. CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 1–19.