Annotations Matter: Leveraging Multi-task Learning to Parse UD and SUD

Zeeshan Ali Sayyed
Indiana University
Department of Computer Science
zasayyed@iu.edu

Daniel Dakota
Uppsala University
Department of Linguistics
ddakota@lingfil.uu.se

Abstract

Using multiple treebanks to improve parsing performance has shown positive results. However, to what extent similar, yet competing annotation decisions play in parser behavior is unclear. We investigate this within a multi-task learning (MTL) dependency parser setup on two parallel treebanks, UD and SUD, which, while possessing similar annotation schemes, differ in specific linguistic annotation preferences. We perform a set of experiments with different MTL architectural choices, comparing performance across various input embeddings. We find languages tend to pattern in loose typological associations, but generally the performance within an MTL setting is lower than single model baseline parsers for each annotation scheme. The main contributing factor seems to be the competing syntactic annotation information shared between treebanks in an MTL setting, which is shown in experiments against differently annotated treebanks. This suggests that the impact of how the signal is encoded for annotations and its influence on possible negative transfer is more important than that of the input embeddings in an MTL setting.

1 Introduction

Multi-task learning (MTL; Caruana, 1997) has shown promise in various NLP tasks such as semantic dependency parsing (Peng et al., 2017; Hershcovich et al., 2018; Kurita and Søgaard, 2019), machine translation (Dong et al., 2015) and multiword expression detection (Taslimipoor et al., 2019).

MTL inherently is designed to share information between tasks, which has helped various NLP components (Collobert and Weston, 2008). One active research question however is what information in specific tasks should be shared, as well was what indicators can be used to predetermine the cost-benefit trade-offs of MTL for a given application. Findings have shown that label distributions (Martínez Alonso and Plank, 2017), data sizes (Bollmann et al., 2018) and single task loss curves (Bingel and Søgaard, 2017) have all been respective indicators for MTL performance. Different tasks, data sizes, and settings can all show different relative performance gains (Adouane and Bernardy, 2020). Thus, it is still an open question under which circumstance MTL can be used to achieve max performance boosts over a single task system.

In syntactic parsing, learning a closely related task (e.g. POS tagging) in a joint paradigm benefits overall performance (Bohnet and Nivre, 2012; Zhang and Weiss, 2016), and work has also exploited MTL by leveraging two or more treebanks against each other (see section 2). We often assume simply increasing data and the sharing of syntactic information will inherently benefit all parsers, but this assumes that all syntactic sharing, specifically all annotation sharing, is positive and complementary. However, annotation decisions have been shown to favor parsing preferences (Rosa, 2015; Rehbein et al., 2017; Kohita et al., 2017). This means that is is not necessarily clear if sharing annotations benefits all parsers equally. This is especially true if two annotation schemes choose drastically different approaches when annotating specific linguistic phenomena.

We look to examine this issue further by utilizing a set of treebanks that are annotated on parallel data, Universal Dependencies (UD; Nivre et al., 2016) and Surface-Syntactic Universal Dependencies (SUD; Gerdes et al., 2018), to examine how two competing syntactic annotation schemes behave when used in an MTL setup. Using parallel treebanks also removes the lexical variation and influences of domain differences that are present in most MTL treebank setups. Whether this is a positive or negative in an MTL setup is unclear,

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 3467–3481

but reduction in domain differences tend to benefit single model parsers.

We utilize the graph-based Deep Biaffine Parser of Dozat and Manning (2017) in an MTL architecture, treating each UD and SUD treebank of a selected language as a task, and experiment with sharing different embeddings, layers, and loss functions. Additionally, we look at the how different embeddings interact with these annotations along with their role in encoding the signal utilized by the MTL parsers, and whether results follow any linguistic patterns. Finally, we perform additional experiments with treebanks from SPMRL shared task (Seddah et al., 2013, 2014) to support our analysis. We look to investigate the following questions:

1. How will competing syntactic annotations schemes on parallel treebanks behave in an MTL parser?

2. What impact do different input embeddings have on behavior in such a setup?

3. When a treebank is paired with a non-parallel treebank possessing noticeably different syntactic annotations, do trends hold?

2 Related Work

The use of multiple treebanks has been successfully incorporated in parsing strategies. Recent multilingual multi-treebank work by Schuster et al. (2019) extended the Biaffine parser by Dozat and Manning (2017) to incorporate deep contextualized multilingual embeddings in combination with multiple treebank sources, demonstrating gains for zero-shot parsing. Smith et al. (2018a) noted using smaller groups of closely related languages is preferable to larger datasets of dissimilar ones. Multiple synthetic treebanks derived from closely related languages were used to parse Faroese by Barry et al. (2019), though a single language source model yielded the best results.

More directly related work is Johansson (2013), who shares features between two treebanks of the same language that differ in annotation schemes by identifying overlapping features. Using a graph-based parser, he achieved noticeable relative error reduction in UAS for four language pairs, with the largest performance gains on the smaller treebanks. This was followed by Johansson and Adesam (2020) using a neural transition-based parser and leveraging a mixture of treebanks, three dependency and two constituency, against a single constituency treebank in a multi-treebank setup. They find that in all settings, performance on the target constituency treebank improves, with the highest gain coming from using all five as an auxiliary treebank. Kankanampati et al. (2020) use the Multidimensional Easy First approach introduced by Constant et al. (2016) to parse the Arabic CATiB (Habash and Roth, 2009) and its converted UD representation in a multi-task setup. They note that both treebanks showed error reduction, but that improvements were due to partial dependencies, and not primarily driven through lexical sharing.

Little direct work exists on extensive empirical investigations between UD and SUD with parsers. Recent work by Kulmizev et al. (2020) performed probing experiments across a set of languages to extract dependency graphs from BERT (Devlin et al., 2019) and ELMO (Peters et al., 2018) language models, finding that both models prefer UD, with tree shape directly correlated to preference strength.

One of the advantages of MTL is the ability to share information as well as altering objective functions between tasks. Early work examined the impact different loss functions have on downstream applications (Hall et al., 2011) and how in a hierarchy of tasks, sharing of individual layers benefits other tasks differently, with lower level task sharing most beneficial (Søgaard and Goldberg, 2016).

Both hard and soft sharing of parameters have proven successful. Duong et al. (2015) exploited soft parameter sharing between different cross-lingual treebanks possessing the same annotation schemes achieving results on the target language with only half the needed annotated data. Soft sharing of parameters allows nuances between languages of the same treebank when hard sharing all other parameters (Stymne et al., 2018).

Parameter sharing has proven effective in both monolingual (Guo et al., 2016) and multilingual parsing (Ammar et al., 2016; Kitaev et al., 2019). However, what are the optimal parameters to share, and where to do so in the architecture, particularly in cross-lingual setups, is not consistent as shown by de Lhoneux et al. (2018) in extensive experiments in sharing word and character LSTM parameters.

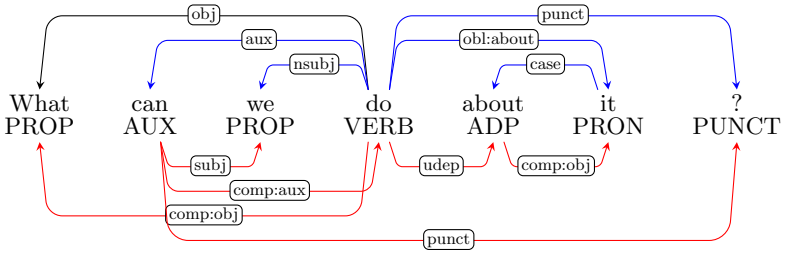


Figure 1: UD and SUD annotation example from English EWT Treebank

Treebank			ar-padt	de-gsd	el-gdt	en-ewt	fi-tdt	fr-gsd	hu-szeged	ko-gsd	ru-gsd	tr-imst	vt-vtb	zh-gsd
UD	Train	Non-Proj	.0892	.0950	.0596	.0523	.0606	.0399	.2572	.1620	.0623	.1105	.0314	.0233
	Dev	Non-Proj	.0825	.0626	.0521	.0275	.0689	.0407	.3356	.1568	.0587	.1204	.0213	.0014
SUD	Train	Non-Proj	.2331	.2118	.1420	.0948	.1423	.0857	.3384	.1764	.0901	.1444	.0957	.4412
	Dev	Non-Proj	.2090	.1815	.1266	.0679	.1645	.0800	.4082	.1684	.0933	.1478	.1075	.4320
	Total Train		6075	13814	1662	12543	12217	14449	910	4400	3850	3664	1400	3997
	Total Dev		909	799	403	2002	1364	1476	441	950	579	988	800	500

Table 1: Proportion of Non-Projective Trees in UD and SUD Train and Dev Sets

3 Experimental Setup

3.1 Data

UD have become a de facto standard as a source for treebanks for dependency parsing. A main annotation choice in UD is the prioritization of content words as the head. While some functional distinctions are kept, such as those between subjects and objects, many other are merged, such as complements and adjuncts. Importantly, function words are dependents of the content words.

SUD were developed as a counter-balance to UD with the belief that UD are not syntactically motivated enough, with a particular linguistically argued objection to the prioritization of content words as heads, stemming from the belief that the distributional context of words should drive headedness. While many individual labels are kept, several are collapsed into a single label (e.g. nsubj & csubj $\rightarrow$ subj). The primary result of function words becoming heads is the inherent reversal of syntactic relationships of many words.

Fig. 1 is an example of how the SUD conversion alters an English sentence from its original UD representation. One of the more noticeable differences is that the projective UD tree is now non-projective in the SUD schema. The main cause, in this example, is because the auxiliary verb *can* is now the root in SUD, rather than the content word *do* in UD. Furthermore, the only word to retain the same head word between the two sentences is *what*, while all others have new heads. We wish to emphasize however, that not all trees show such stark contrasts, but simply want to highlight how a simple choice in annotation can produce distinctly different trees, and the resultant impact on non-projectivity.

By using UD and SUD, we eliminate one of the variables in many multi-treebank setups, the different distribution of the underlying vocabulary. This effectively eliminates domain differences between the treebanks (see section 3.2), as both parsers will get more similar outputs from the BiLSTM layer, and identical ones in a joint loss setting.

We Use UD and SUD version 2.7 and select 12 different language from 10 language families. This was done in order capture sufficient linguistic variation in terms of how UD and SUD may impact various linguistic phenomena found in typologically different languages, and subsequently annotation schemes.[1] Table 1 presents statistics on the treebanks in respect to their variation in training and dev sizes. Additionally, we also note the proportion of non-projective trees found in each annotation scheme.[2] All languages show higher number of non-projective trees in SUD when compared to their UD counterparts, but for some it is much more substantial. A noticeable example is Chinese (zh) which has 40% more absolute non-projective trees in its SUD treebank compared to its UD counterpart. Noticeable increases can also be seen in Arabic and German, but most languages show only moderate differences. Hungarian (hu) is

[1] We restrict ourselves to treebanks that contained complete training, dev, and test splits.

[2] We note that this is not always entail explicit linguistic non-projectivity, as in many cases punctuation is the source of non-projectivity which can be viewed as non-linguistic.

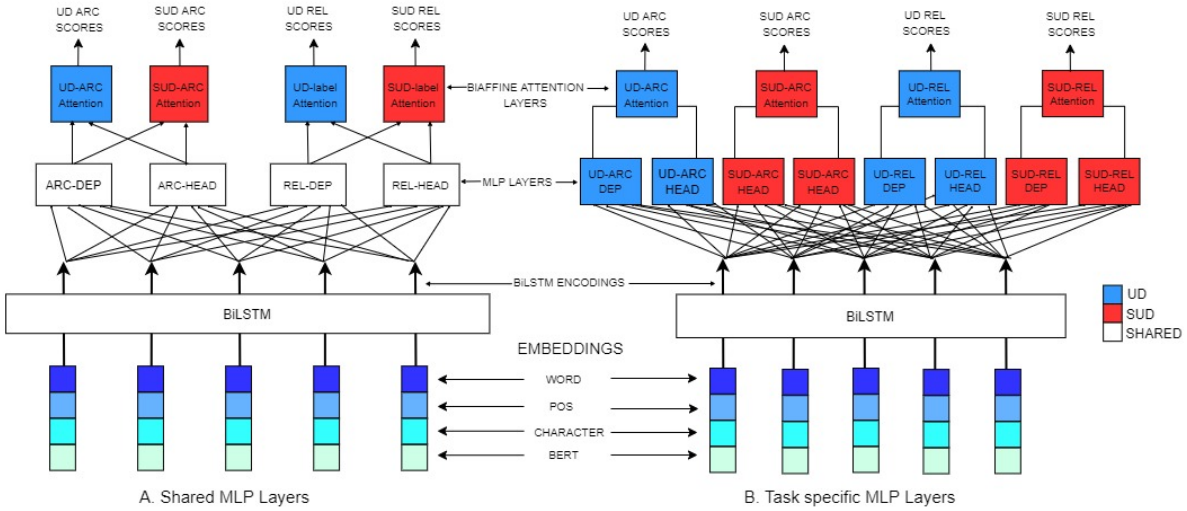


Figure 2: Multi-task model architecture.

interesting as it is the only language that shows a high proportion of non-projective trees in UD and only a moderate increase for SUD.

3.2 MTL Parsing Architecture

We use the PyTorch (Paszke et al., 2019) implementation of the Biaffine parser of Dozat and Manning (2017) provided by Zhang et al. (2020),[3] and extend it to an MTL architecture.[4]

We modify the base parser by treating parsing of each annotation scheme as a separate task. Each task shares the BiLSTM layer that is used to encode the concatenation of all input embeddings. These BiLSTM encodings are then passed through dimension reducing MLPs to strip away arc and relationship information information deemed not relevant. We implement two MLP schemes, one in which we share them across tasks (shared; Figure 2A) and the other in which each task has its own MLP layers (unshared; Figure 2B). Considering the overlap in the annotation schemes, a shared MLP setting allows us to examine the behavior of sharing information between the two annotation scheme when irrelevant information is minimized. Finally, in order for the model to learn task specific information, we apply task specific biaffine attention layers to the MLP outputs to produce scores for both arcs and labels.

The common practice in MTL is to have separate losses for different tasks and to optimize for each of them separately (*alternating loss*; Ruder, 2017). This is particularly the case when the different tasks do not share the same input. However, our dataset contains parallel sentences albeit with different annotations. It thus then becomes possible to experiment with using a *joint loss* for training both tasks as the parsers receive the same input, and a joint loss has shown improvements when joint learning POS tags and dependency parsing (Li et al., 2018). We do this by optimizing for the sum of losses of each of the tasks. Since the losses of both tasks are of nearly the same magnitude, we do not have to worry about imbalance and a simple sum suffices.[5] We experiment with both types of losses.

In the alternating loss setting, we randomly choose a task from the given tasks and then randomly choose a batch of sentences along with their annotations from that task before calculating the loss of that batch and backpropagating the errors. In a given epoch we chose sentences without replacement. For joint loss, we randomly choose a batch of the same sentences from both the tasks, along with their different annotations. Losses are calculated based on those annotations and summed together before backpropagating the errors. We posit that joint loss should allow for faster convergence as both the tasks affect the parameter updates of the shared layers simultaneously, thus helping the optimization process to move towards the goal more quickly.

The two choices of losses combined with the op-

[3] https://github.com/yzhangcs/parser

[4] Our code is available at https://github.com/zeeshansayyed/multiparser

[5] When losses in an MTL setting do not have comparable magnitude, then the joint loss tends to more influenced by the task with larger loss; thus, producing a learning bias.

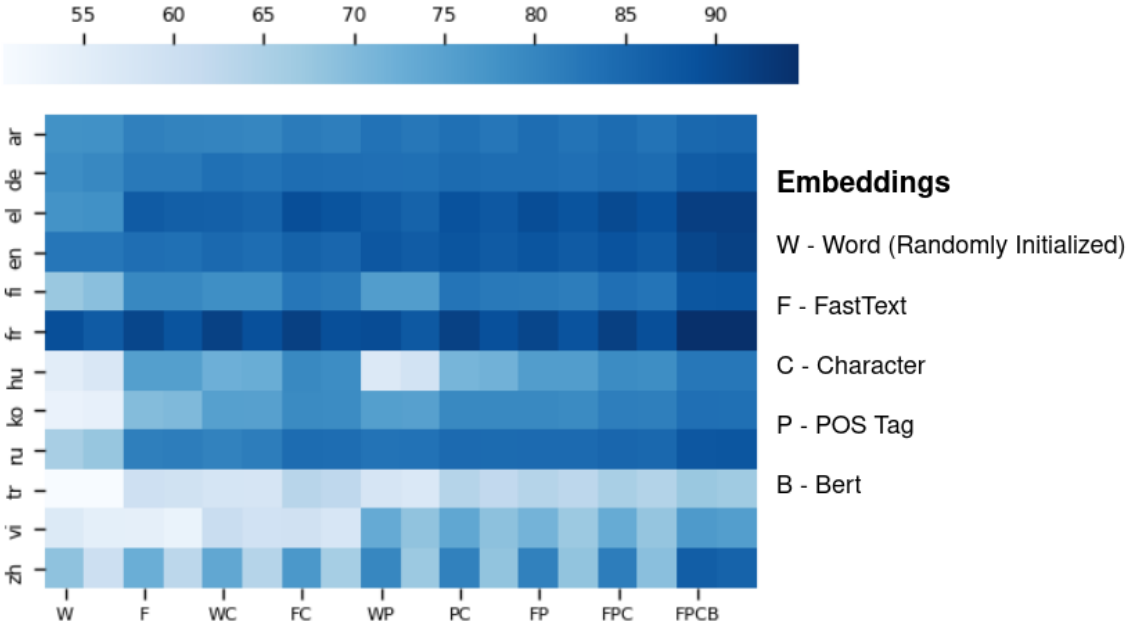


Figure 3: Heatmap illustrating the mean LAS scores for the four MTL settings across all languages and embedding types. UD-LAS is represented in the left column and SUD-LAS in the right column for each embedding input.

tional sharing of MLP layers gives rise to four different experimental settings: alternating-unshared, alternating-shared, joint-unshared, and joint-shared. In addition, we experiment with internally randomly initialized word and POS[6] embeddings, external embeddings (FastText; Bojanowski et al. (2017)) and BERT (Devlin et al., 2019)), and their concatenations as inputs to the BiLSTM layers. All results are reported on the dev sets using the CoNLL 2018 Shared Task Scorer (Zeman et al., 2018).

4 Results

The overall performances of the four experimental settings, namely alternating vs joint loss and shared vs unshared MLP layers, are very close to each other. The convergence statistics for joint and alternating loss settings are reported in Table 2.[7] It can be noted that despite taking a greater number of epochs to converge when compared to alternating loss, joint loss converges faster in terms of time because it performs the forward propagation through the shared layers only once for both tasks, whereas alternating loss has to perform it separately for each task.

As we are more interested in the MTL parser behavior across experimental settings, we report the mean LAS score over the four MTL settings in all our experiments to capture the general trends of the MTL parser.

Parameter	# Epochs	Time (seconds)
Joint Loss	342	2 774
Alternating Loss	297	3 907

Table 2: Convergence statistics for Joint and Alternating Loss

To analyze the impact of different embedding types on the MTL parsing setup, we change the specificity of information by using different embedding types with the MTL parser as discussed in section 3.2, results of which are presented in Fig 3. We see that adding more information yields in higher LAS across languages (moving from left to right on the heatmap) with the concatenation of all embeddings (rightmost columns) performing the best.

However, given that we are more interested in examining whether the parallel UD-SUD treebanks can benefit from an MTL setup, we choose instead to focus on how the MTL parsers compare to the single UD and SUD baseline parsers across the different embedding choices. Fig. 4 shows a heatmap depicting the difference of the mean LAS of all four settings with respect to the corresponding single

[6] We use gold POS tags.

[7] All experiments were performed on Nvidia V100 GPUs. Tables 2 and 3 analyses do not include BERT experiments.

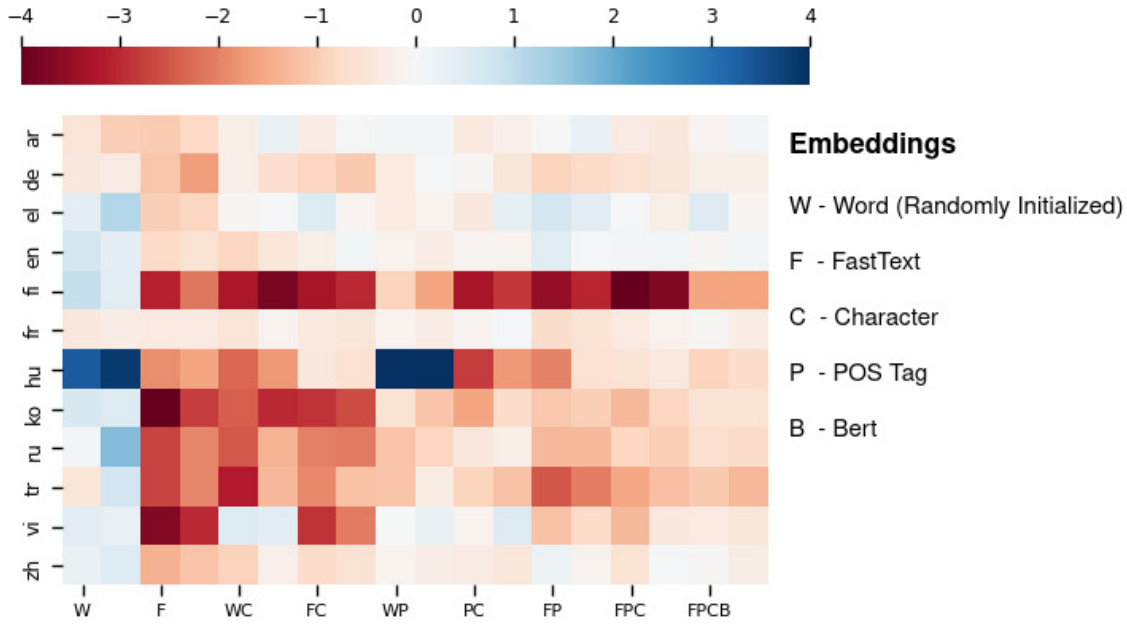


Figure 4: Heatmap illustrating the performance difference between MTL parsers compared to corresponding single task baseline parser. Each block represents the difference between the mean LAS score of four MTL settings and the respective single task baseline LAS score.

baseline parser, for each embedding input.[8]

The mean drop in LAS scores for MTL settings when compared to the baselines across all languages and all the different feature embeddings (432 runs) are reported in Table 3, with lower numbers indicating better performance. No particular setting shows a significant improvement over the other. Keeping this in consideration, we still see that joint loss performs slightly better than alternating loss. Sharing of MLP layers seems to help a little compared to the setting where we have task specific layers. As mentioned in section 3.2, the role of dimension reducing MLPs is to remove all the information that is not necessary for performing the task at hand. This would indicate that the two tasks remove similar unnecessary information, thereby sharing the signal necessary for making parsing decisions.

One of the most striking observations is that randomly initialized word embeddings (seen in the far left two columns) are noticeably lighter across all languages. This stands in stark contrast to the subsequent FastText (FT), word+char and FT+char embeddings. Hungarian shows particularly noticeable improvements, though it may be due to its size. However, given that we also see some moderate improvements for English, Greek and Russian, the size of the treebanks is not the only contributing factor.

Parameter	Mean Drop (LAS)
Joint Loss	0.70
Alternating Loss	0.75
Shared MLP	0.69
Unshared MLP	0.76

Table 3: Mean drop in LAS compared to baseline

Once inputs include word embeddings initialized with FT embeddings or randomized char embeddings, we see some interesting trends. Finnish, Hungarian, Korean, Turkish, and Russian show consistent degradation in performance with any inclusion of character-based embeddings. Some languages show more stable results, regardless of the input embeddings, namely Greek, English, French, and Chinese. Vietnamese clearly performs worse when using FT embeeddings to initialize the word embeddings, but otherwise is rather stable in other settings. However, even with BERT (B) embeddings, we do not see any noticeable improvement over the baselines.

[8]We also experimented with task-specific fine-tuning following Liu et al. (2019) on MTL parsers. While it did lead to improvements, the overall distribution across all the different settings and languages was not considerably different. Also see Appendix for additional heatmaps contrasting shared vs unshared and alternating vs joint loss settings. The overall pattern remains the same.

When we begin incorporating POS tag embeddings, we see that drops relative to the baseline for these languages become less pronounced, but few languages ultimately show improvements, with Hungarian being the noticeable exception. The pattern in the reduction in degradation of performance when including POS tag embeddings continues across all settings. The exception being Finnish, which still shows large drops in almost all settings, but show slightly reduced drops when including BERT embeddings.

5 Discussion

5.1 UD vs SUD

We see, in general, a systematic decrease in performance when using parallel UD and SUD treebanks in an MTL setup across many languages. When looking for linguistic behaviors, we can clearly see that agglutinative languages (Hungarian, Finnish Turkish, and Korean) all suffer severe performance drops when using character-based embeddings, but the concatenation of POS embeddings helps mitigate the degradation. The absolute differences of Hungarian and Finnish are noticeably different compared to one another. This may be somewhat unexpected, given they are in the same language family. However, the modern forms are quite different and treebank sizes may play a role, as input embeddings pattern similarly overall between the two.

The morphological complexity of other languages in relation to their behavior is not necessarily a good predictor of behavior. However, if we view the other eight remaining languages on a continuum of fusional and analytical properties, we can see some general patterns.

Russian, a fusional language, patterns with the agglutinative languages in its behavior with character embeddings, but is also one of the more morphologically rich languages (MRL) of the non agglutinative languages. The other more fusional MRLs, German and Greek, also do not see as much volatility, although German tends to be worse respective to the baseline, while Greek shows some more positive results, but this could again be due to treebank sizes. English and French are more analytical than the other fusional languages and contain far less morphology. While both show rather consistent minimal degradation regardless of the input embedding, English occasionally shows some improvement, while French virtually none.

Arabic and Vietnamese, however, are somewhat odd cases. Arabic is both fusional and an MRL, whereas Vietnamese is much more analytical. Vietnamese, though, patterns more with the other MRLs, while Arabic patterns more like the analytical language, particularly with its less overall performance degradation compared to baselines across settings. Vietnamese shows one of the larger performance drops relative to the baseline compared to the other eight languages when using FT embeddings in the input, but this is diminished when combined with additional embeddings.

Chinese, an extremely analytical language, presents an additionally interesting case. The LAS for SUD is usually on average 10% absolute lower than its UD counterpart, which can be seen in Fig. 3. This probably is a direct result of the SUD treebank having 40% absolute more non-projective trees. However, this massive disparity in non-projectivity has seemingly not resulted in additional performance degradation in the MTL setup (as seen in Fig. 4), suggesting that sharing between treebanks that show large differences in non-projectivity is not necessarily detrimental.

Given the general behavior across settings, performance degradation can most likely be attributed to negative transfer derived from the different annotation preferences UD and SUD encode, which is not seen in the single model baselines. When different embeddings are used, the negative transfer is either accentuated depending upon the language, as seen with character embeddings, or some embeddings seem to help mitigate the negative transfer, as with POS embeddings. Interestingly, although character-based embeddings show significant improvements in the single baseline models compared to word embeddings, the signals they encode seem to be detrimental in an MTL setting, as performances drops relative to their respective single model baselines. This would seemingly suggest that in an MTL setup, word and POS embeddings are encoding more beneficial signals that help both annotation schemes, reducing possible negative transfer from each treebank, whereas character embeddings are maximally beneficial when used to train a single model. This is in line with recent work showing that the linguistic information POS tags convey, when highly accurate or gold, still have value for specific use cases, and are beneficial in certain dependency parsing architectures or as auxiliary tasks (Anderson and Gómez-Rodríguez,

Exp.	MLP	Arabic				German				Hungarian			
		Char + FastText				Char + FastText				Char + FastText			
		UD		SUD		UD		SUD		UD		SUD	
		UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS
	baseline	86.59	82.25	86.50	81.44	89.18	85.11	87.81	84.98	84.73	79.97	84.03	79.02
UD-SUD	unshared	86.33	81.93	86.34	81.33	87.95	84.06	86.74	83.70	84.28	79.42	83.53	78.56
	shared	86.33	81.97	86.22	81.33	87.83	84.08	86.76	83.73	84.50	79.72	85.83	79.05
SPMRL	unshared	87.78	83.47	87.52	82.62	90.16	86.44	89.81	87.18	87.04	82.23	86.99	82.73
	shared	87.50	83.24	87.38	82.46	90.01	86.39	89.36	86.50	72.16	82.76	87.91	83.32

Table 4: Results for MTL experiments with SPMRL dataset. All MTL experiments are trained with the alternating batch loss setting to allow for comparison with experiments involving SPMRL. We cannot use joint loss when training SPMRL with either UD or SUD as they are not parallel treebanks. The UD-SUD experiment shows results for UD and SUD when they are trained together in an MTL setting, whereas the SPMRL experiment shows results for UD and SUD when each of them is separately trained along with the corresponding SPMRL dataset instead of each other (UD-SPMRL & SUD-SPMRL).

2020; Zhou et al., 2020).

One specific often overlooked annotation issue that helps convey this point is punctuation.[9] In both annotation schemes, punctuation attachment is rather straight forward. However, as seen in Fig 1, it is one of the competing annotation decisions. In both annotation schemes, punctuation is simply attached directly to the root. However, in UD the root is a content word, while in SUD the root is a function word. Thus, while straight forward form an annotation perspective for both schemes, an MTL system is now learning both attachment possibilities simultaneously and preferences and errors regarding both are now being encoded in the global attachment decisions. When looking at specific attachment errors, across almost all examined experiments, there were substantial increases in punctuation attachment errors. This can be seen as a direct result of switching the content versus function oriented headedness and creates systematic, competing attachment decisions for an MTL parser exposed to both attachment possibilities.

5.2 UD and SUD vs SPMRL

To further explore whether the competing annotation decisions between UD and SUD are indeed contributing to the noticeable performance degradation, we choose to compare a subset of languages in an MTL setup but with a differently annotated treebank. Using a different treebank runs the risk of adding additional domain issues into our experiments; however, character-level embeddings have proven effective at handling OOV words (Ballesteros et al., 2015; Vania et al., 2018), thus domain differences should be reduced.

We perform experiments where we use the Arabic (Habash and Roth, 2009), German (Brants et al., 2004), and Hungarian (Vincze et al., 2010) treebanks from the SPMRL Shared task (Seddah et al., 2013, 2014), each of which were annotated with language specific linguistic phenomena in mind.[10] To mitigate size difference issues as the smaller treebank tends to benefit more in a multi-treebank setup (Johansson, 2013), we randomly select a train and dev set from the SPMRL data respectively to match the corresponding size of the UD-SUD treebanks.[11]

Results for the SPMRL experiments using char+FT embeddings are presented in Table 6.[12] All results in UD-SPRML and SUD-SPMRL MTL experiments show improved performance over the baseline. Importantly, this includes settings in which the UD-SUD MTL experiments show noticeable decreases relative to the baseline, and specifically we see that character-based embeddings are able to yield benefit in an MTL setup relative to the baseline.

These results suggests that the annotation

[9] The CoNLL 2018 evaluation scores punctuation. We are not, however, making a claim as to whether punctuation is or is not a linguistic issue, rather simply highlight it is an annotation attachment issue that illustrates different possible attachment distributions between the annotation schemes. From a linguistic perspective, punctuation can be argued to be irrelevant, from a parsing perspective, unless removed, it still influences attachment decisions.

[10] We refer to the reader to the cited papers for more detailed information on the individual treebank annotations.

[11] We note that the both German and Hungarian UD-SUD Treebanks are derived from a small section of the TiGer and Szeged Treebanks respectively, which are also the treebanks for the SPMRL data, thus there is a possibility of sentence overlap in the random selection. The UD-SUD Arabic treebank is derived from the Prague Arabic Dependency Treebank (Hajič et al., 2004) but is also annotated on newswire.

[12] Results using word+POS embeddings are provided in the Appendix.

schemes are indeed contributing to why UD and SUD make poor tasks for each other in an MTL setup, and not strictly the embeddings themselves. Rather, the information conveyed by each individual annotation scheme is important in terms of the possible gains that MTL parsers can make over the baseline parsers. It may simply be that how the annotations are embedded into the architecture and shared are more influential in what signals are encoded in the network than the embeddings themselves in terms of how they benefit treebanks in an MTL setup. If the annotations themselves encode information that results in negative transfer in the network due to their competing nature, an MTL setup cannot benefit as effectively.

6 Conclusion

We implemented an MTL architecture leveraging parsing UD and SUD as separate tasks to examine how their syntactic annotation overlaps and differences influence parser behavior. We find that models from an MTL setup perform generally worse than their single model baselines, regardless of input embeddings. Interestingly, POS embeddings seemingly help mitigate some of the performance loss caused from negative transfer as the POS information may help resolve possible linguistic ambiguities with which character embeddings struggle (Vania et al., 2018; Smith et al., 2018b). This stands in contrast to much multi-treebanking research which has yielded positive performance gains when using multiple treebanks, particularly if they are of the same language, though this is not always the case (Barry et al., 2019).

We then further investigated the possible influence annotations have in an MTL setup by training a subset of SPMRL treebanks against their UD-SUD counterparts, finding increases in performance across the chosen languages and input embeddings not seen when pitting UD and SUD together. We argue that this indicates that in an MTL setup, simply adding another treebank is not inherently going to yield better performance, rather the information that each additional treebank can learn from the other, specifically from their annotation schemes, and how this is then subsequently encoded in the network is a more pivotal factor in yielding performance gains.

We conclude that the syntactic annotation schemes are pertinent when determining performance gains in an MTL parsing setup, as extensive competing annotations provides too many mixed signals in an MTL architecture, hampering the ability of both parsers to benefit from shared information, yielding worse results.

Future research will include incorporating more treebanks with different annotation schemes to examine in which directions and annotations parsers will optimize towards in MTL. We also wish to further explore how constituency parsing and dependency parsing can be leveraged against each other in similar MTL setups.

Acknowledgements

The authors would like to thank Sandra Kübler and members of the Uppsala Parsing Group: Artur Kulmizev, Joakim Nivre, and Sara Stymne for their feedback, as well as the anonymous reviewers for their comments. This research was supported in part by Lilly Endowment, Inc., through its support for the Indiana University Pervasive Technology Institute. The second author is supported by the Swedish strategic research programme eSSENCE.

References

Wafia Adouane and Jean-Philippe Bernardy. 2020. When is multi-task learning beneficial for low-resource noisy code-switched user-generated Algerian texts? In *Proceedings of the The 4th Workshop on Computational Approaches to Code Switching*, pages 17–25, Marseille, France. European Language Resources Association.

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. On the frailty of universal POS tags for neural UD parsers. In *Proceedings of the 24th Conference on Computational Natural Language Learning*, pages 69–96, Online.

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, Lisbon, Portugal.

James Barry, Joachim Wagner, and Jennifer Foster. 2019. Cross-lingual parsing with polyglot training and multi-treebank learning: A Faroese case study. In *Proceedings of the 2nd Workshop on Deep Learning Approaches for Low-Resource NLP (DeepLo 2019)*, pages 163–174, Hong Kong, China.

Joachim Bingel and Anders Søgaard. 2017. Identifying beneficial task relations for multi-task learning in deep neural networks. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, pages 164–169, Valencia, Spain.

Bernd Bohnet and Joakim Nivre. 2012. A transition-based system for joint part-of-speech tagging and labeled non-projective dependency parsing. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1455–1465, Jeju Island, Korea.

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.

Marcel Bollmann, Anders Søgaard, and Joachim Bingel. 2018. Multi-task learning for historical text normalization: Size matters. In *Proceedings of the Workshop on Deep Learning Approaches for Low-Resource NLP*, pages 19–24, Melbourne. Association for Computational Linguistics.

Sabine Brants, Stefanie Dipper, Peter Eisenberg, Silvia Hansen, Esther König, Wolfgang Lezius, Christian Rohrer, George Smith, and Hans. Uszkoreit. 2004. TIGER: Linguistic Interpretation of a German Corpus. *Journal of Language and Computation*, 2004 (2):597–620.

Rich Caruana. 1997. Multitask learning. *Machine Learning*, 28(1):41–75.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pages 160–167, Helsinki, Finland.

Matthieu Constant, Joseph Le Roux, and Nadi Tomeh. 2016. Deep lexical segmentation and syntactic parsing in the easy-first dependency framework. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1095–1101, San Diego, California.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4171–4186, Minneapolis, Minnesota.

Daxiang Dong, Hua Wu, Wei He, Dianhai Yu, and Haifeng Wang. 2015. Multi-task learning for multiple language translation. 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 1723–1732, Beijing, China.

Timothy Dozat and Christopher Manning. 2017. Deep biaffine attention for neural dependency parsing. In *5h International Conference on Learning Representations (ICLR 2017)*, Toulon, France. Conference Track Proceedings.

Long Duong, Trevor Cohn, Steven Bird, and Paul Cook. 2015. Low resource dependency parsing: Cross-lingual parameter sharing in a neural network parser. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 845–850, Beijing, China.

Kim Gerdes, Bruno Guillaume, Sylvain Kahane, and Guy Perrier. 2018. SUD or surface-syntactic Universal Dependencies: An annotation scheme near-isomorphic to UD. In *Proceedings of the Second Workshop on Universal Dependencies (UDW 2018)*, pages 66–74, Brussels, Belgium.

Jiang Guo, Wanxiang Che, Haifeng Wang, and Ting Liu. 2016. A universal framework for inductive transfer parsing across multi-typed treebanks. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 12–22, Osaka, Japan.

Nizar Habash and Ryan Roth. 2009. CATiB: The Columbia Arabic treebank. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 221–224, Suntec, Singapore. Association for Computational Linguistics.

Jan Hajič, Otakar Smrž, Petr Zemánek, Jan Šnaidauf, and Emanuel Beška. 2004. Prague Arabic Dependency Treebank: Development in Data and Tools. In *Proceedings of the NEMLAR International Conference on Arabic Language Resources and Tools*, pages 110–117, Cairo, Egypt.

Keith Hall, Ryan McDonald, Jason Katz-Brown, and Michael Ringgaard. 2011. Training dependency parsers by jointly optimizing multiple objectives. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1489–1499, Edinburgh, Scotland, UK.

Daniel Hershcovich, Omri Abend, and Ari Rappoport. 2018. Multitask parsing across semantic representations. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 373–385, Melbourne, Australia.

Richard Johansson. 2013. Training parsers on incompatible treebanks. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 127–137, Atlanta, Georgia.

Richard Johansson and Yvonne Adesam. 2020. Training a Swedish constituency parser on six incompatible treebanks. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 5219–5224, Marseille, France.

Yash Kankanampati, Joseph Le Roux, Nadi Tomeh, Dima Taji, and Nizar Habash. 2020. Multitask easy-first dependency parsing: Exploiting complementarities of different dependency representations. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2497–2508, Barcelona, Spain (Online).

Nikita Kitaev, Steven Cao, and Dan Klein. 2019. Multilingual constituency parsing with self-attention and pre-training. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3499–3505, Florence, Italy.

Ryosuke Kohita, Hiroshi Noji, and Yuji Matsumoto. 2017. Multilingual back-and-forth conversion between content and function head for easy dependency parsing. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 1–7, Valencia, Spain.

Artur Kulmizev, Vinit Ravishankar, Mostafa Abdou, and Joakim Nivre. 2020. Do neural language models show preferences for syntactic formalisms? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4077–4091, Online.

Shuhei Kurita and Anders Søgaard. 2019. Multi-task semantic dependency parsing with policy gradient for learning easy-first strategies. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2420–2430, Florence, Italy. Association for Computational Linguistics.

Miryam de Lhoneux, Johannes Bjerva, Isabelle Augenstein, and Anders Søgaard. 2018. Parameter sharing between dependency parsers for related languages. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4992–4997, Brussels, Belgium.

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, Brussels, Belgium.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019. Multi-task deep neural networks for natural language understanding. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 4487–4496. Association for Computational Linguistics.

Héctor Martínez Alonso and Barbara Plank. 2017. When is multitask learning effective? semantic sequence prediction under varying data conditions. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 44–53, Valencia, Spain.

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)*, pages 1659–1666, Portorož, Slovenia. European Language Resources Association (ELRA).

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *arXiv preprint arXiv:1912.01703*.

Hao Peng, Sam Thomson, and Noah A. Smith. 2017. Deep multitask learning for semantic dependency parsing. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 2037–2048, Vancouver, Canada. Association for Computational Linguistics.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana.

Ines Rehbein, Julius Steen, Bich-Ngoc Do, and Anette Frank. 2017. Universal Dependencies are hard to parse – or are they? In *Proceedings of the Fourth International Conference on Dependency Linguistics (Depling 2017)*, pages 218–228, Pisa,Italy.

Rudolf Rosa. 2015. Multi-source cross-lingual delexicalized parser transfer: Prague or Stanford? In *Proceedings of the Third International Conference on Dependency Linguistics (Depling 2015)*, pages 281–290, Uppsala, Sweden.

Sebastian Ruder. 2017. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*.

Tal Schuster, Ori Ram, Regina Barzilay, and Amir Globerson. 2019. Cross-lingual alignment of contextual word embeddings, with applications to zero-shot dependency parsing. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1599–1613, Minneapolis, Minnesota.

Djamé Seddah, Sandra Kübler, and Reut Tsarfaty. 2014. Introducing the SPMRL 2014 Shared Task on Parsing Morphologically-rich Languages. In *Proceedings of the First Joint Workshop on Statistical Parsing of Morphologically Rich Languages and Syntactic Analysis of Non-Canonical Languages*, pages 103–109, Dublin, Ireland. Dublin City University.

Djamé Seddah, Reut Tsarfaty, Sandra Kübler, Marie Candito, Jinho D. Choi, Richárd Farkas, Jennifer Foster, Iakes Goenaga, Koldo Gojenola Galletebeitia, Yoav Goldberg, Spence Green, Nizar Habash, Marco Kuhlmann, Wolfgang Maier, Joakim Nivre, Adam Przepiórkowski, Ryan Roth, Wolfgang Seeker, Yannick Versley, Veronika Vincze, Marcin Woliński, Alina Wróblewska, and Eric Villemonte de la Clergerie. 2013. Overview of the SPMRL 2013 Shared Task: A Cross-Framework Evaluation of Parsing Morphologically Rich Languages. In *Proceedings of the Fourth Workshop on Statistical Parsing of Morphologically-Rich Languages*, pages 146–182, Seattle, Washington, USA.

Aaron Smith, Bernd Bohnet, Miryam de Lhoneux, Joakim Nivre, Yan Shao, and Sara Stymne. 2018a. 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, Brussels, Belgium.

Aaron Smith, Miryam de Lhoneux, Sara Stymne, and Joakim Nivre. 2018b. An investigation of the interactions between pre-trained word embeddings, character models and POS tags in dependency parsing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2711–2720, Brussels, Belgium.

Anders Søgaard and Yoav Goldberg. 2016. Deep multi-task learning with low level tasks supervised at lower layers. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 231–235, Berlin, Germany.

Sara Stymne, Miryam de Lhoneux, Aaron Smith, and Joakim Nivre. 2018. Parser training with heterogeneous treebanks. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 619–625, Melbourne, Australia.

Shiva Taslimipoor, Omid Rohanian, and Le An Ha. 2019. Cross-lingual transfer learning and multitask learning for capturing multiword expressions. In *Proceedings of the Joint Workshop on Multiword Expressions and WordNet (MWE-WN 2019)*, pages 155–161, Florence, Italy.

Clara Vania, Andreas Grivas, and Adam Lopez. 2018. What do character-level models learn about morphology? the case of dependency parsing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2573–2583, Brussels, Belgium.

Veronika Vincze, Dóra Szauter, Attila Almási, György Móra, Zoltán Alexin, and János Csirik. 2010. Hungarian dependency treebank. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, Valletta, Malta. European Language Resources Association (ELRA).

Daniel Zeman, Jan Hajič, Martin Popel, Martin Potthast, Milan Straka, Filip Ginter, Joakim Nivre, and Slav Petrov. 2018. CoNLL 2018 shared task: Multilingual parsing from raw text to Universal Dependencies. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 1–21, Brussels, Belgium.

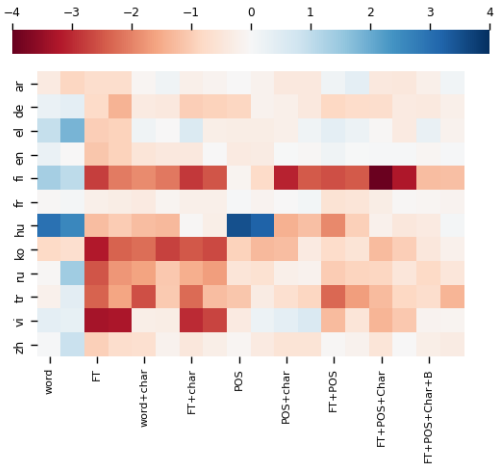
Yu Zhang, Zhenghua Li, and Min Zhang. 2020. Efficient second-order TreeCRF for neural dependency parsing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3295–3305, Online. Association for Computational Linguistics.

Yuan Zhang and David Weiss. 2016. Stack-propagation: Improved representation learning for syntax. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 1557–1566, Berlin, Germany.

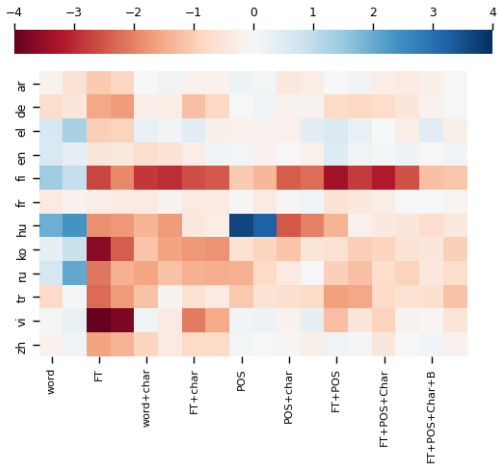
Houquan Zhou, Yu Zhang, Zhenghua Li, and Min Zhang. 2020. Is pos tagging necessary or even helpful for neural dependency parsing? In *Natural Language Processing and Chinese Computing*, pages 179–191, Cham. Springer International Publishing.

Appendix

A Heatmaps for Shared and Unshared MLP layer settings

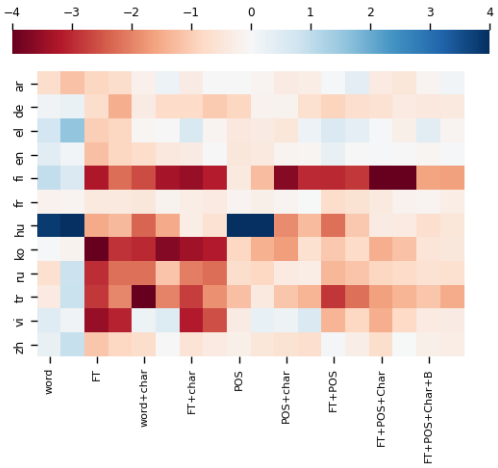


Unshared UAS

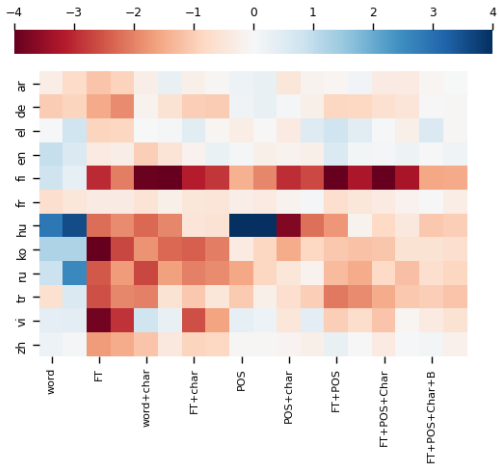


Shared UAS

Figure 5: Heatmaps depicting the shared and unshared MLP layers settings. Each block represents the mean UAS score across alternating and joint loss setting for the corresponding embedding and MLP setting.



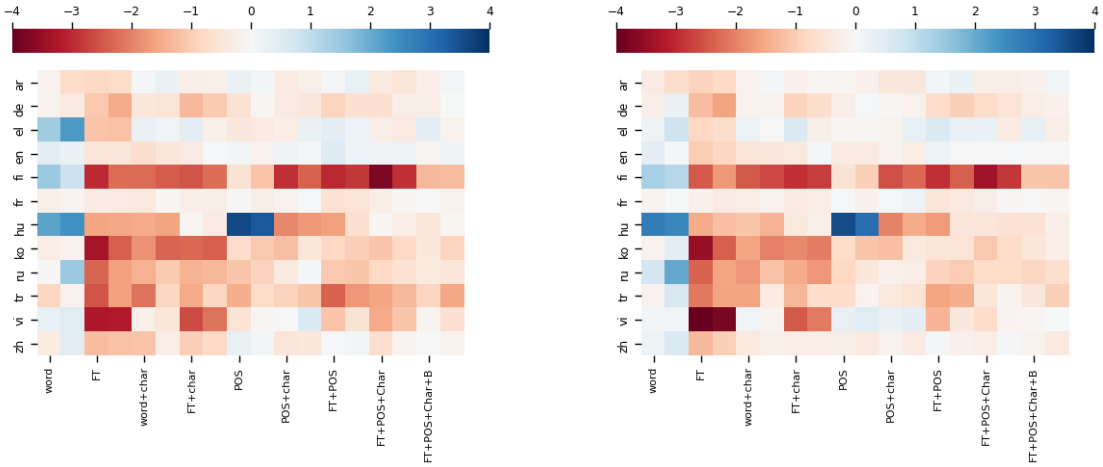
Unshared LAS



Shared LAS

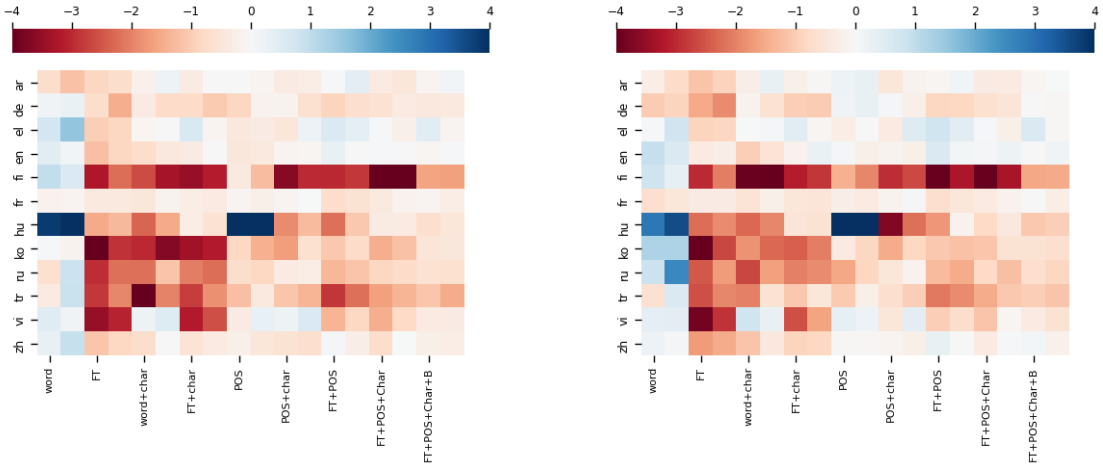
Figure 6: Heatmaps depicting the shared and unshared MLP layers settings. Each block represents the mean LAS score across alternating and joint loss setting for the corresponding embedding and MLP setting.

B Heatmaps for Alternating vs Joint Loss settings



Alternating Loss (UAS) Joint Loss (UAS)

Figure 7: Heatmaps depicting the when joint loss or alternating loss settings. Each block represents the mean UAS score across shared and unshared MLP settings for the corresponding embedding and loss setting.



Unshared LAS Shared LAS

Figure 8: Heatmaps depicting the when joint loss or alternating loss settings. Each block represents the mean LAS score across shared and unshared MLP settings for the corresponding embedding and loss setting.

C Training Hyperparameters

Hyperparameters	Value
Embedding Dimensions	300
Character Embedding Dimension	50
POS Tag Embedding Dimension	100
Bert Mapping Dimenstion	100
Number of BERT Layers Used	4
Embed Dropout	0.33
Number of LSTM Layers	400
LSTM Hidden Layer Dimension	400
LSTM Dropout	0.33
MLP (Arc) Output Dimension	500
MLP (Rel) Output Dimension	100
MLP Dropout	0.33
Optimizer	Adam
Patience	50
Batch Size	20000 tokens
Learning Rate	2e-3
Eps	1e-12
Betas	(.9, .9)
Clip	5.0
Decay	0.75
Decay Steps	5000

Table 5: Hyperparameter settings

D UD-SUD vs SPRML MTL Results Table

		Arabic								German								Hungarian							
Exp.	MLP	Word + POS				Char + FastText				Word + POS				Char + FastText				Word + POS				Char + FastText			
		UD		SUD		UD		SUD		UD		SUD		UD		SUD		UD		SUD		UD		SUD	
		UAS	LAS	UAS	LAS	UA	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS
	baseline	86.66	83.30	86.87	82.40	86.59	82.25	86.50	81.44	88.50	84.22	86.64	83.63	89.18	85.11	87.81	84.98	65.20	52.26	65.80	54.35	84.73	79.97	84.03	79.44
UD-SUD	unshared	86.79	83.49	86.79	82.39	86.33	81.93	86.34	81.33	87.15	82.84	86.23	83.13	87.95	84.06	86.74	83.70	69.05	57.02	69.16	58.92	84.28	79.42	83.53	78.56
	shared	87.05	83.70	87.17	82.94	86.33	81.97	86.22	81.33	88.68	84.43	86.97	84.03	87.83	84.08	86.76	83.73	68.58	57.16	69.15	59.14	84.50	79.72	85.83	79.05
SPMRL	unshared	87.52	84.13	87.69	83.15	87.78	83.47	87.52	82.62	89.25	85.40	88.93	86.28	90.16	86.44	89.81	87.18	71.86	60.95	73.68	63.37	87.04	82.23	86.99	82.73
	shared	87.62	84.39	87.67	83.15	87.50	83.24	87.38	82.46	89.27	85.34	88.91	86.09	90.01	86.39	89.36	86.50	72.16	61.64	73.42	63.67	87.63	82.76	87.91	83.32

Table 6: Results for MTL experiments with SPMRL dataset. All MTL experiments are trained with the alternating batch loss setting to allow for comparison with experiments involving SPMRL. We cannot use joint loss when training SPMRL with either UD or SUD as they are not parallel treebanks. The UD-SUD experiment shows results for UD and SUD when they are trained together in an MTL setting, whereas the SPMRL experiment shows results for UD and SUD when each of them is separately trained along with the corresponding SPMRL dataset instead of each other (UD-SPMRL & SUD-SPMRL).