Bidirectional Domain Adaptation Using Weighted Multi-Task Learning

Daniel Dakota
Uppsala University
ddakota@lingfil.uu.se

Zeeshan Ali Sayyed
Indiana University
zasayyed@iu.edu

Sandra Kübler
Indiana University
skuebler@indiana.edu

Abstract

Domain adaption in syntactic parsing is still a significant challenge. We address the issue of data imbalance between the in-domain and out-of-domain treebank typically used for the problem. We define domain adaptation as a Multi-task learning (MTL) problem, which allows us to train two parsers, one for each domain. Our results show that the MTL approach is beneficial for the smaller treebank. For the larger treebank, we need to use loss weighting in order to avoid a decrease in performance below the single task. In order to determine to what degree the data imbalance between two domains and the domain differences affect results, we also carry out an experiment with two imbalanced in-domain treebanks and show that loss weighting also improves performance in an in-domain setting. Given loss weighting in MTL, we can improve results for *both* parsers.

1 Introduction

Domain adaption in syntactic parsing is still a significant challenge. While recent work has shown steady improvements, we have not necessarily achieved proportionally better results as expected with neural models (Fried et al., 2019). One simple reason can be attributed to the fact that we have severe limitations in terms of existing data, data sizes, and the imbalance between the two treebanks typically present in domain adaptation settings.

Multi-task learning (MTL; Caruana, 1997) allows for joint learning, which can help facilitate cross information sharing between tasks. This has proven particularly beneficial for tasks that have large data imbalances, with the smaller data tasks benefiting substantially more (Johansson, 2013; Benton et al., 2017; Ruder et al., 2019), and should thus also be useful in domain adaptation.

We define domain adaptation as an MTL problem where the two tasks correspond to training on two treebanks from different domains. Note that in this setting, we do not have a primary and a secondary task, but instead we can interpret both tasks as primary.

One of the inherent difficulties we face in cross-domain parsing are large discrepancies in the size of the treebanks. This creates a default training scenario of imbalance across the domains that should benefit the smaller domain, but it may inversely impact the larger domain, resulting in a degradation in performance due to negative transfer in a multi-task learning model, compared to their single task (STL) baselines. We investigate here whether it is possible to control the negative impact of the smaller domain on the larger one, and how the imbalance factor impacts this balance. This means that in all cases, we evaluate on *both* domains.

More specifically, we investigate the following questions:

1. How does the MTL parser handle different levels of data imbalance? This assumes that in a domain adaptation setting, normally a small in-domain treebank is combined with a large out-of-domain treebank.

2. How effective is loss weighting in addressing the data imbalance? Can we optimize both MTL tasks given the data imbalance?

3. Is it more important to address the data imbalance or the differences between domains for successful domain adaptation?

2 Related Work

2.1 MTL in Parsing

MTL inherently allows for the joint learning of tasks. Learning related tasks, such as POS tagging and dependency parsing, has been shown to be beneficial (Bohnet and Nivre, 2012; Zhang and Weiss,

Proceedings of the 17th International Conference on Parsing Technologies (IWPT 2021), pages 93–105
Bangkok, Thailand (online), August 6, 2021. ©2021 Association for Computational Linguistics

2016). A practical assumption is that there is shared information that can be beneficial, particularly if the tasks are closely related.

Neural networks have increased the ability and ease by which models can exploit information sharing. Much recent parsing research has examined the impact of parameter sharing across treebanks and languages (Ammar et al., 2016; Kitaev et al., 2019), though often not explicitly within an MTL setup, where different treebanks/languages are treated as multiple tasks. Soft sharing of parameters has proven effective on treebanks of the same annotation style in lower resource settings (Duong et al., 2015) as well as for multiple treebanks of the same language when hard sharing all other parameters (Stymne et al., 2018). However, sharing too many parameters between unrelated languages has been shown not to be beneficial (de Lhoneux et al., 2018).

More explicit MTL settings with treebanks representing different individual tasks have proven successful in array of settings across languages and architectures (Guo et al., 2016; Johansson and Adesam, 2020; Kankanampati et al., 2020).

2.2 MTL in Domain Adaptation

Much recent work has resulted in significant gains in domain adaptation across several languages via the direct or indirect transfer of parameters and embeddings for languages such as Chinese (Li et al., 2019, 2020), English (Joshi et al., 2018; Fried et al., 2019), Finnish (Virtanen et al., 2019), and French (Martin et al., 2020).

More explicit MTL work by Søgaard and Goldberg (2016) found that lower level tasks are best kept at lower layers, as the shared representations benefit from the sequence of information learned, with the approach demonstrating success in domain adaption for chunking for English. Using hyperlinks as a form of weak supervision was used by Søgaard (2017) to improve several NLP tasks both in-domain and out-of-domain, including chunking, for both English and Quechua. Peng and Dredze (2017) use an MTL setting to leverage Chinese word segmentation and NER across two domains, news and social media. They share lower levels but retain domain specific projection layers with task specific models. Results outperform disjoint adaption methods and suffer less from diminishing returns as training sizes increase.

2.3 MTL Performance

While MTL has resulted in improvements across many tasks and settings, an STL can still outperform an MTL model (Martínez Alonso and Plank, 2017; Bingel and Søgaard, 2017; Liang et al., 2020). Reasons for such a lack of increase or even degradation in performance for a certain task may be found in negative transfer as tasks may learn at different rates, and a single task may dominate the learning (Lee et al., 2016), or poor scheduling may result in catastrophic forgetting (French, 1999).

Another key fact is the correct choice of tasks. However, it is not clear how to best select tasks. Auxiliary task label distributions (Martínez Alonso and Plank, 2017), the learning curve of the primary task (Bingel and Søgaard, 2017), the difficulty of the auxiliary task (Liebel and Körner, 2018), the relationship between the data of the tasks in terms of size (Luong et al., 2015; Benton et al., 2017; Augenstein and Søgaard, 2017; Schulz et al., 2018) and properties (Wu et al., 2020), among other findings[1], have all shown to influence the effectiveness of MTL.

One way to mitigate the negative transfer is to give different weights to the tasks, helping to maximize the contributions for the more pertinent tasks and lessen the impact of sub-optimal tasks (Lee et al., 2016, 2018). Such strategies have shown promise in computer vision, where optimal loss weights can allow an MTL model to improve over a corresponding STL when it would otherwise show a degradation in performance (Kendall et al., 2018).

Winata et al. (2018) weighted losses for language modeling and POS tagging in an MTL setting, finding a lower weight to language modeling yielded a reduction in perplexity in modeling code-switching between Chinese and English. A multi-task supervised pretraining adaption strategy using a hierarchical architecture that learns multiple tasks on a source domain before fine-tuning them on the target was implemented by Meftah et al. (2020). By using different weights for the different level tasks, starting with higher weights for lower tasks before incrementally increasing weights to higher level tasks during training, they achieve a noticeable error reduction in POS tagging, dependency parsing, and chunking.

Our experiments focus on improving parsing in a domain adaptation setting using MTL plus separate

[1] See (Søgaard and Goldberg, 2016; Guo et al., 2019; Schröder and Biemann, 2020) for more discussion.

		Train	Dev	Test
German	GSD	13 814	799	977
	tweeDe	1 000	150	151
Italian	ISDT	13 121	564	482
	TWITTIRÒ	1 000	144	142
	PoSTWITA	1 000	150	150
	ParTUT	1 000	150	150

Table 1: Treebank sizes (number of sentences).

loss weighting to improve both tasks.

3 Methodology

3.1 Treebanks

For our experiments, we focus on German and Italian, since both languages have smaller treebanks based on Twitter data, which will allow us to avoid domain differences in the smaller domain. We use treebanks annotated with Universal Dependencies V2.7 (Nivre et al., 2020). For German, we use GSD, which is based on news, reviews, and Wikipedia pages, and tweeDe (Rehbein et al., 2019) as the Twitter treebank. For Italian, we use ISDT and ParTUT, which consist of legal, news, and Wikipedia texts, plus TWITTIRÒ (Cignarella et al., 2019) and PoSTWITA (Sanguinetti et al., 2018) as the Twitter treebanks.

Table 1 shows the sizes of the treebanks used in our experiments. For tweeDe we used the first 1 000 sentences for train, and the following 150 and 151 sentences for dev and test respectively. In order to account for treebank size variations of the Twitter treebanks, we limit the maximal sizes of train and dev for TWITTIRÒ and PoSTWITA to the first 1 000 train sentences and 150 dev and test sentences respectively, but we do not reduce the GSD or ISDT treebanks. For the in-domain experiments in section 6, we also use the ParTUT treebank, since it covers domains similar to ISDT. We reduce the treebank size in the manner in which we reduce the Twitter treebanks.

3.2 Parser

We use the graph-based neural dependency parser by Dozat and Manning (2017) as our base parser and extend it to an MTL architecture using hard parameter sharing. Our MTL parser treats the parsing of each treebank as a separate MTL task, where the tasks share the BiLSTM layers which encode the input embeddings, which are calculated by concatenating all the different types of embeddings that are used. These BiLSTM encodings are then passed through a dimension-reducing Multi-layered Perceptrons (MLP) to strip away arc and relationship information deemed irrelevant for the task at hand. We implement two MLP schemes, one in which we share the MLP layers across tasks (shared-MLP; left part of Figure 1) and the other in which each task has its own MLP layers (unshared; right part of Figure 1). Finally, in order for the model to learn task specific information, we apply task-specific biaffine attention layers to the MLP output to produce scores for both arcs and labels. A more detailed description of the parser architecture can be found in Sayyed and Dakota (2021).

Hyperparameters	**Value**
Embedding Dimensions	300
POS Tag Embedding Dimension	100
Bert Mapping Dimenstion	100
Number of BERT Layers Used	4
Number of LSTM Layers	3
LSTM Hidden Layer Dimension	400
Optimizer	Adam
Patience	50
Batch Size	20k tokens
Learning Rate	2e-3

Table 2: Hyperparameter settings for MTL parser.

We modify the PyTorch (Paszke et al., 2019) implementation of the biaffine parser provided by Zhang et al. (2020)[2], to implement our MTL parser[3]. We retain many of the default hyperparameters used in the original base parser. Table 2 lists the parameters which we have changed. Word and POS embeddings are initialized randomly. For the BERT embeddings (Devlin et al., 2019), a scalar mixture of the last four layers of BERT is passed through a linear layer to produce BERT embeddings of the specified dimension.

3.3 Loss Weighting

We train the MTL parser by having a different objective function for each task, and we optimize for each task separately. We do this by randomly choosing a task from the given tasks and then randomly choosing a batch of sentences along with their annotations from that task before calculating the loss for that batch, backpropagating the errors, and updating all the model parameters (shared and

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

[3] Our code is available from https://github.com/zeeshansayyed/multiparser

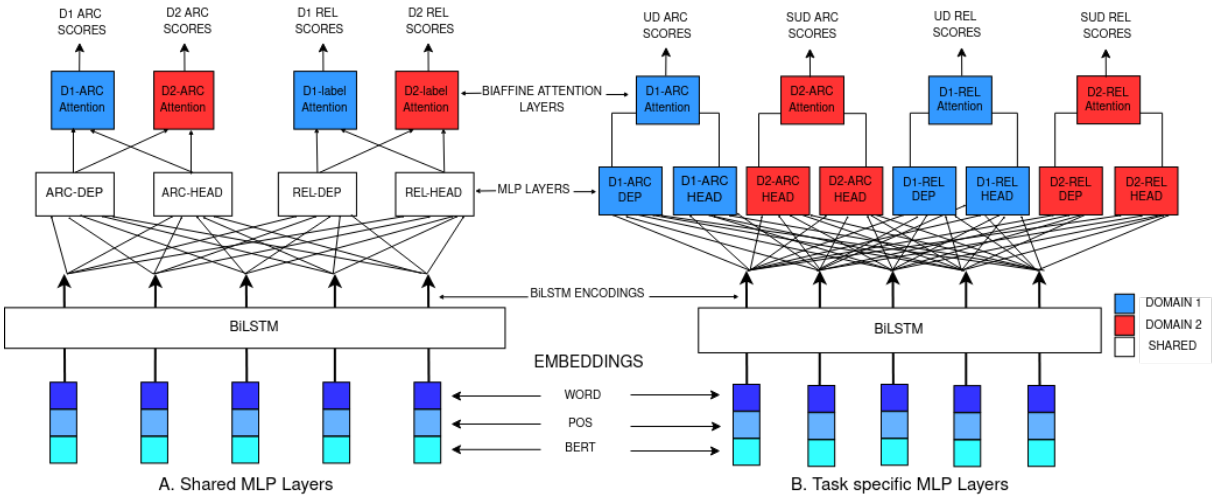

Figure 1: Multi-task model architecture.

unshared). In a given epoch, we chose sentences without replacement.

The task specific objective function is given by:

$$\mathcal{L}_t(X_t;\Theta) = \mathcal{L}_a(X_t;\Theta) + \mathcal{L}_l(X_t;\Theta) \quad (1)$$

where t is the task number, X_t represents the input batch for task t, Θ denotes all model parameters, $\mathcal{L}_a$ and $\mathcal{L}_l$ represent the cross entropy losses for arc and label scores in batch X_t respectively.

In the loss weighting experiments in section 5, in order to compensate for the varying difficulty of tasks, we weigh each objective function differently and then optimize for it, as follows:

$$\mathcal{L}_t(X_t;\Theta) = w_t \cdot (\mathcal{L}_a(X_t;\Theta) + \mathcal{L}_l(X_t;\Theta)) \quad (2)$$

where w_t is the weight for task t and $\sum w_t = 1$.

3.4 Experimental Setup

Our focus is on comparing the performance of the MTL parser against that of an STL baseline. This means that we need a better understanding of the interactions between treebank sizes, learning curves, and weighting. For this reason, we show learning curves comparing corresponding MTL and STL models or different MTL weight settings.

For the learning curves, we start with 1 000 sentences from each treebank and then double the training size for the large treebank until the max. size is reached, while the Twitter treebanks remain fixed.

We experiment with three different input embeddings: We train word and POS embeddings (using gold POS tags as input). We then use the word embeddings, POS embeddings, and the combination of word and POS embeddings plus BERT embeddings (Devlin et al., 2019). For the BERT embeddings, German and Italian language specific BERT embeddings are used[4].

We report the average LAS score over three seeds (10, 20, 30)[5] using the CoNLL 2018 shared task scorer (Zeman et al., 2018). All reported experiments are on the development set unless otherwise stated.

4 STL vs. MTL Learning Curves

Our first set of experiments concerns the question how different levels of size imbalance in the training data from the two domains affect an MTL parser.

Figure 2 presents learning curves comparing the German treebanks, Figures 3 and 4 compare the two Italian Twitter treebanks to ISDT. Across all settings, we see that using only word embeddings gives the lowest results, followed by word plus POS tag embeddings, with word plus POS tag plus BERT embeddings resulting in the highest LAS. $\mathcal{L}_t(X_t;\Theta) = w_t \cdot (\mathcal{L}_{a;t} + \mathcal{L}_{l;t})$

Also across all experiments, we see that sharing the MLP layers (in green) is slightly more beneficial to the smaller Twitter treebanks than having separate layers (in blue). The differences between the shared and unshared model are even smaller for

[4] https://github.com/stefan-it/fine-tuned-berts-seq

[5] For a small number of settings, one of the seeds produced results that were >30 points lower than for the other seeds. For those cases, we used seed 40 as a replacement.

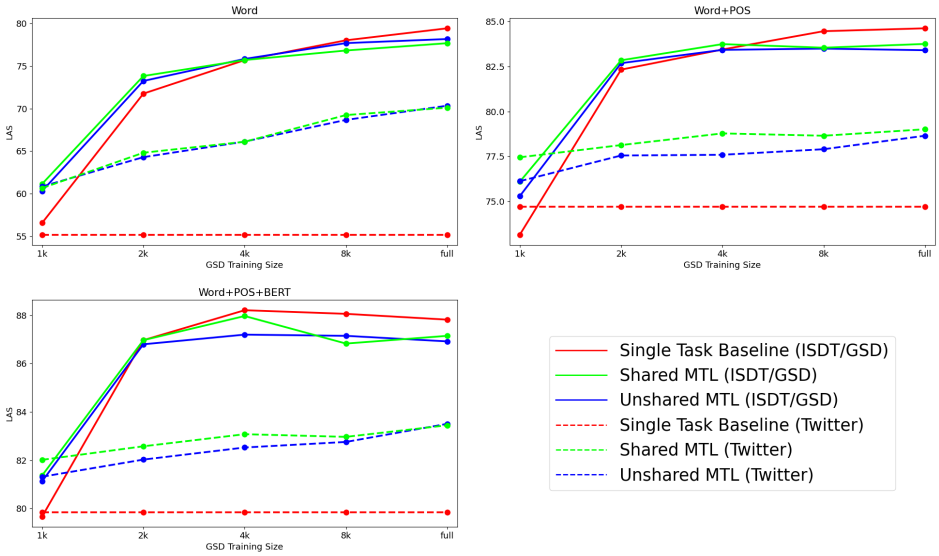

Figure 2: Results on German (GSD and tweeDe) when adding GSD training sentences (using word, word+POS, and word+POS+BERT embeddings respectively).

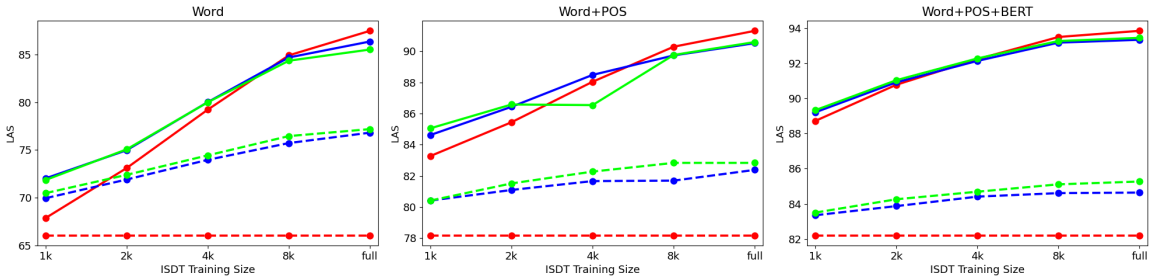

Figure 3: Results on Italian TWITTIRÒ when adding ISDT training sentences (word, word+POS, and word+POS+BERT embeddings). For legend, see Figure 2.

the large treebanks.

The two German treebanks have very similar performance at 1k while there is a considerable difference between the Italian trebanks, with the Twitter treebanks being more difficult to parse than ISDT.

When looking at the results on German in Figure 2, we see that in all settings, the MTL setup is beneficial for the tweeDe treebank, independent of the amount of out of domain training data. The same holds initially for the GSD treebank.

However, as more GSD data becomes available, the MTL setup starts to be detrimental for the GSD parser, and the results stay below the STL baseline. Additionally, the type of embeddings plays a factor in the overall improvement: While the curves flatten out quickly for word+POS and word+POS+BERT embeddings, the curve for word embeddings indicates that adding the full set still results in gains. It is also concerning that for the word+POS and word+POS+BERT embeddings, the results on GSD start decreasing after 4k GSD sentences. We attribute this to negative transfer, i.e., the MTL setting is more focused on finding an optimal solution between the non-Twitter and the Twitter task, resulting in a degradation in performance of the non-Twitter task from the equally prioritized Twitter signals. Note that while this makes sense from a MTL point of view, it is counter-intuitive from a domain adaptation perspective: For the GSD task, this means that adding more in-domain data results in lower in-domain performance.

The results for the Italian treebanks in Figures 3 and 4 exhibit similar, but not identical trends to the German experiments. One noticeable difference is that the ISDT improvements tend to be far more linear, where we only begin to see a flattening of the curve at the full training size. Another more interesting difference concerns the point where the STL on the large treebank (GSD or ISDT) improves

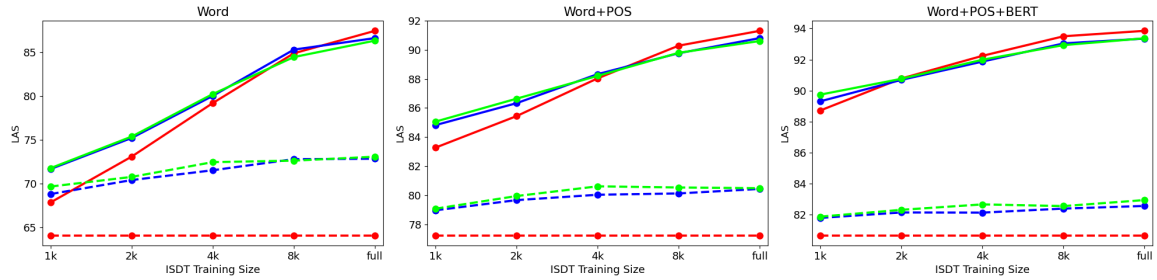

Figure 4: Results on Italian PoSTWITA when adding ISDT training sentences (word, word+POS, and word+POS+BERT embeddings). For legend, see Figure 2.

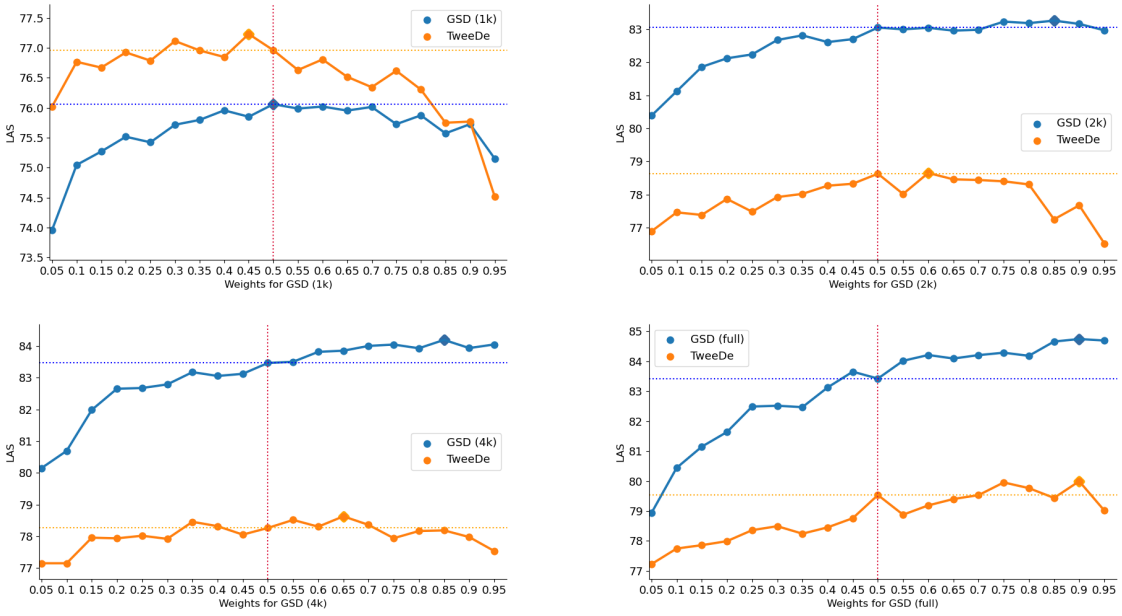

Figure 5: Effect of loss weighting in German; all experiments use 1k tweeDe and range from 1k, 2k (upper), 4k to all GSD data (lower) for training.

over the MTL model: For German, this happens systematically at 4k, for Italian, it ranges from 4-8k for TWITTIRÒ and 2-8k for PoSTWITA depending on the type of embeddings. We see that for word+POS+BERT, the STL is already on par with the MTL model at 2k, but the word embeddings model needs closer to 8k.

5 Loss Weighting

We now turn to the question of loss weighing, i.e., can we address the data imbalance problem if we assign higher weights to one or the other task in the loss function? Using loss weighting would give us a principled way of handling the data imbalance. For a description of the weighting scheme, see section 3.3. For the experiments in this section, all MTL tasks are learned using the shared MLP setting (as it performed better in most settings of section 4) and word+POS embeddings (as it performed better than word-only embeddings but does not require pretraining on external sources, such as BERT), and we vary the training size of the large treebank between 1k, 2k, 4k, and all available training data.

Figure 5 shows the results for the German treebanks, Figure 6 shows the combination of the Italian ISDT and TWITTIRÒ, and Figure 7 the combination of ISDT and PoSTWITA. Each graph shows the results for the two tasks per setting.

All experiments in section 4 are based on standard alternating loss for both tasks, i.e., the loss for each task is used as is, which is equivalent to having a weight of 1. For the experiments here, the X-axis denotes the weights applied to the large treebank task, ranging from 0 to 1. The corresponding weight applied to the second task is $1 - x$, i.e., it

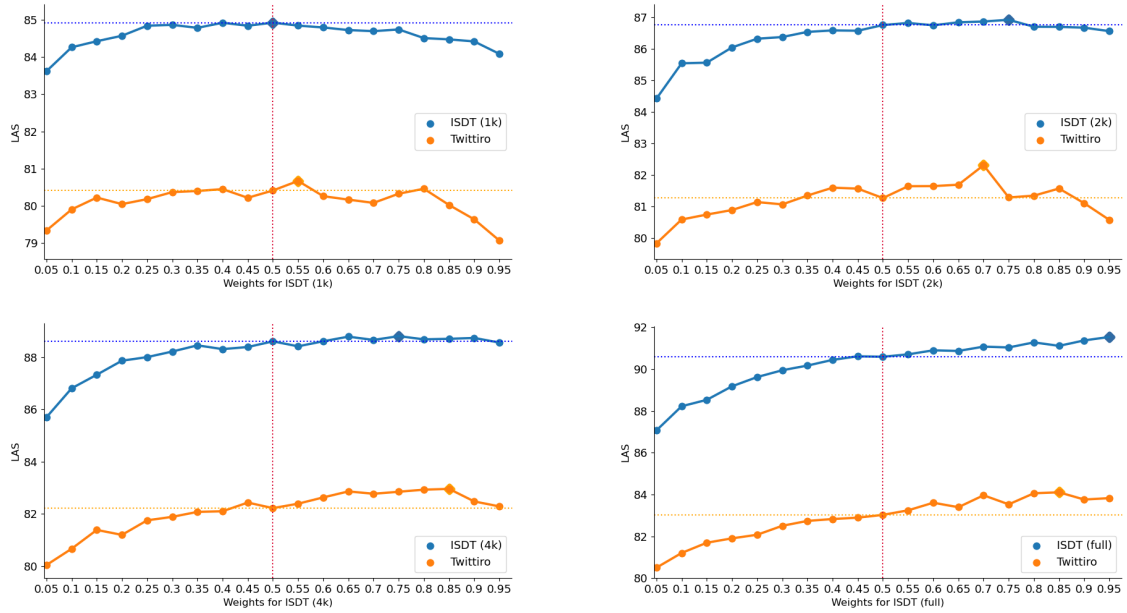

Figure 6: Effect of loss weighting in Italian; all experiments use 1k TWITTIRÒ and range from 1k, 2k (upper), 4k to all ISDT data (lower) for training.

decreases from 1 to 0. For each weight, we report the LAS for either task, blue denoting the large treebank and orange the Twitter treebank. The vertical dotted lines mark the 0.5 weight setting, where both tasks are weighted equally, and the horizontal dotted lines denote the performance of the MTL parser in this setting. The diamond markers in each line denote the best performance achieved by weighted MTL parser for that task.

Looking at the German results in Figure 5, we see that in the 1k setting, the optimal performance for GSD is reached when each task is assigned a weight of 0.5. For tweeDe, the optimal performance is reached in the same area, namely for a weight setting of 0.45:0.55 for GSD:tweeDe. In the 2k setting, even though the optimal performance shifts towards higher weights for the large treebank (0.85 for GSD and 0.4 for tweeDe), the actual increase in performance is minimal compared to the balanced weight setting. This trend continues as we add more data to GSD (for 4k and full settings): The optimal weights move closer to the maximal weight for the large treebank. This shows that as the treebank sizes become more imbalanced, the optimal performance is reached by applying a higher weight to the large treebank and consequently a lower weight to the small Twitter treebank.

It is also interesting to see that for the 1k setting, the tweeDe results surpass the GSD results. This suggests that the tweeDe data are syntactically easier than the GSD ones. We also see that the performance for tweeDe starts plateauing once it reaches the 0.5 weight setting for imbalanced scenarios (2k, 4k and full). Hence, for tweeDe, the performance gain in the optimal setting beyond 0.5 is smaller than for GSD. Moreover, while the MTL models using balanced weights (section 4) do not always improve over the STL baselines, weighted MTL improves in all settings and all tasks. For GSD, the highest gain over the STL occurs in the 1k setting. The more data we add, the smaller the improvement. For the full GSD set, the improvement is minimal.

The results for the two Italian experiments, in Figure 6 for the combination of ISDT and TWITTIRÒ, and in Figure 7 for the combination of ISDT and PoSTWITA, show similar trends with some differences. For both experiments, the Twitter task shows slightly concave curves for 1k and 2k respectively. Starting from 4k, we see an upward trend for ISDT as the weight for the large task increases. For German, this only occurs with the full GSD training set.

These results show that for smaller data sizes, i.e., when the two treebanks are closer to balanced, weighting does not matter that much: For most settings, the optimal results are close to a 0.5 weight. As the data imbalance increases, it becomes more important to slide the weights towards the larger task in order to avoid negative transfer.

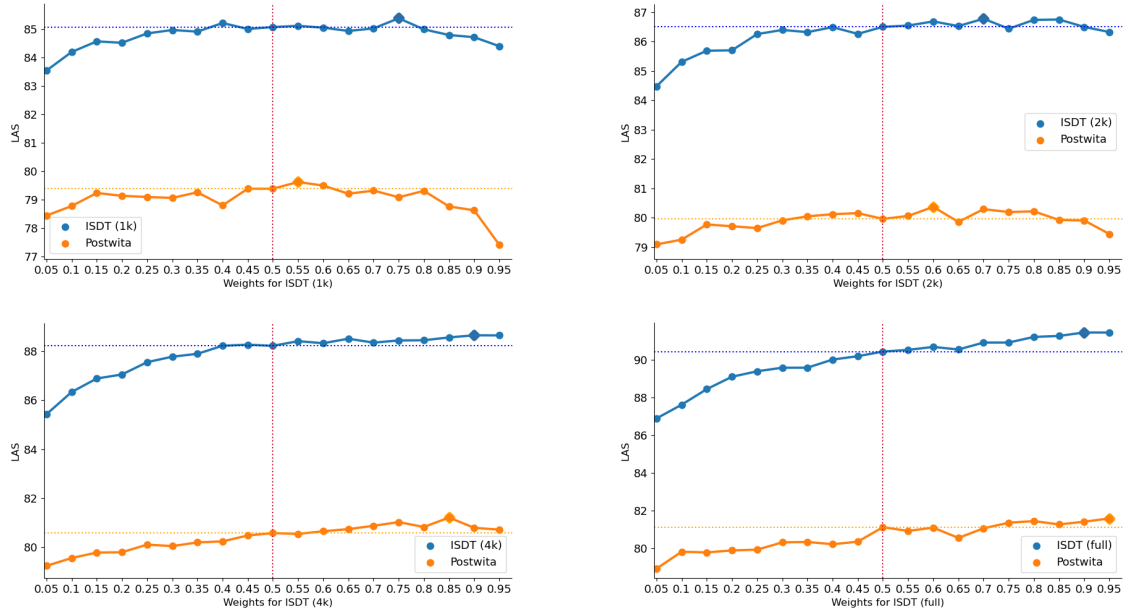

Figure 7: Effect of loss weighting in Italian; all experiments use 1k PoSTWITA and range from 1k, 2k (upper), 4k to all ISDT data (lower) for training.

The weighted MTL models allow us to use the maximum data for the larger treebank and even reach a slight gain in performance over the STL. The Twitter treebanks show even more improvement than in the standard MTL setup, suggesting that they benefit from a reduction of their own signal along with a more powerful signal from the large treebank.

Weighting the different losses is akin to having different learning rates for each task (i.e., treebank), impacting the contribution each has on the step sizes derived from the loss in finding an optimal solution (Sébastien et al., 2018). When the data sizes of the two domains are closer, the steps taken by each benefit each other. As the non-Twitter data increases however, the Twitter data have too much influence on the step sizes, resulting in a degradation on the non-Twitter treebank since standard MTL optimizes both equally. While MTL is thought to help overcome local optima that can occur in an STL (Bingel and Søgaard, 2017), in our case, we assume that the weights have a similar effect: They help both the non-Twitter and Twitter models overcome local optima encountered in a standard MTL setup.

6 Controlling Data Imbalance Vs. Domain Differences

So far, we have treated domain adaptation as a data imbalance problem. This is certainly a factor since we mostly use a large scale out-of-domain treebank to improve results in-domain. However, there are also genuine differences between the domains, and it is unclear to what degree they individually contribute to the difficulty. For this reason we conduct two additional experiments on Italian, in which we pair a larger treebank with its smaller counterpart from the same domain. In other words, we now focus on an in-domain comparison of a small and a large treebank. The more these results deviate from the previous results, the more influence the domain differences have on parser performance.

In the first experiment, we compare Twitter treebanks, i.e., we pair the smaller TWITTIRÒ treebank with the larger PoSTWITA treebank. In the second experiment, we compare the more general treebanks, i.e., we pair the smaller ParTUT treebank with the larger ISDT. For the Twitter in-domain experiments, we simply double the PoSTWITA data as we did with the ISDT reaching the full size (at about 5.3k sentences). For ISDT and ParTUT we follow the same methodology.

Figure 8 shows non-weighted MTL curves for Twitter and non-Twitter in-domain data sets respectively. The curves in both settings exhibit similar trends to the trends in Figure 3 in that the STL system begins to outperform the MTL system at the end of the curves, but that the small in-domain treebanks reach a very similar performance to their larger counterparts. This suggests that while do-

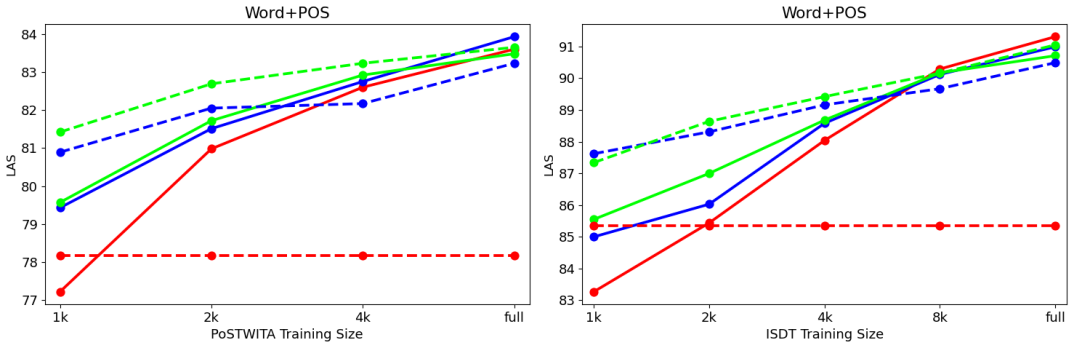

Figure 8: Results on Italian TWITTIRÒ when adding PoSTWITA training sentences on left and Italian ParTUT when adding ISDT on the right, both using word+POS.

	STL	MTL	weighted MTL	wMTL weights
PoSTWITA	83.60	83.46	**83.66**	0.65
TWITTIRÒ	78.17	83.65	**84.14**	0.35
ISDT	91.31	90.71	91.27	0.90
ParTUT	85.35	91.05	**91.34**	0.10

Table 3: LAS Results Word+POS for STL, shared MTL, and weighted shared MTL using the full data sets. ISDT paired with ParTUT.

main differences play a role in negative transfer, they appear to be less important than the data imbalance between treebanks, thus validating our decision to approach domain adaptation as an MTL task.

We conduct a single weighted experiment, using the best weights from Figures 5-7 and the best results reported in section 7. The results are shown in Table 3. We see a similar trend as in the out-of-domain experiments where the MTL setting shows a degradation for ISDT compared to its STL setting while ParTUT shows an increase at higher imbalances. For the Twitter treebanks, the differences are minimal. The weighted MTL increases the performance on ISDT over the MTL setting but does not quite match the STL setting, while the weighted MTL setting further increases the performance of the ParTUT treebank. The weighted MTL for PoSTWITA shows a slight increase over the STL and over the non-weighted MTL setting for TWITTIRÒ. Such findings suggest that even in-domain data imbalances can benefit from weighting, but may not benefit as much as treebank pairs in a domain adaptation setting.

7 Best Results

In Table 4, we show the highest LAS scores for the STL and shared MTL models, on dev and test using Word+POS embeddings. While the unweighted MTL settings result in noticeable lower LAS for both large treebanks, we see slight gains in the weighted MTL experiments, when given upwards of 0.9 loss weights. For all Twitter treebanks, the MTL setting shows considerable gains between 3-5% absolute. The weighted MTL setting improves over MTL by another 1-1.5%. We see the same developments across the dev and test sets.

8 Conclusion & Future Work

We have investigated the use of MTL for domain adaptation in parsing to address the data imbalance. The effectiveness of MTL depends on many interacting factors as laid out in section 2.3. Important factors we directly examine in our experiments are data size imbalances, difficulty of tasks, and task learning rates. Our learning curves for German and Italian show that MTL underperforms an STL when the data size imbalances become too great, due to negative transfer in optimizing for two tasks. Additionally, both our out-of-domain and in-domain experiments demonstrate that task learning difficulty affects both setups, even with balanced data. By using loss weighting, we are able to influence the learning rates of individual tasks, which helps reduce the negative transfer caused by both the data size imbalances and tasks difficulties. This allows us to train weighted MTL models where *both parsers are able to outperform both STL*

Test	Lg.	Treebank	STL	MTL	weighted MTL	wMTL weights
dev	German	GSD	84.63	83.76	**84.74**	0.90
		tweeDe	74.69	79.01	**79.99**	0.10
	Italian	ISDT	91.31	T: 90.59; P: 90.31	T: **91.54**; P: **91.45**	0.95; 0.90
		TWITTIRÒ	78.17	82.83	**84.11**	0.15
		PoSTWITA	77.22	80.48	**81.58**	0.05
test	German	GSD	81.35	80.70	**81.56**	0.90
		tweeDe	76.18	81.77	**82.52**	0.10
	Italian	ISDT	91.80	T: 91.18; P:90.92	T: **92.07**; P: 91.79	0.95; 0.90
		TWITTIRÒ	78.07	81.64	**82.28**	0.15
		PoSTWITA	75.46	78.57	**79.24**	0.05

Table 4: LAS Results Word+POS for STL, shared MTL, and weighted shared MTL using the full data sets. T: paired with TWITTIRÒ, P: with PoSTWITA.

and non-weighted MTL models on both treebanks at the same time, even when highly imbalanced, for in-domain and out-of-domain experiments for both German and Italian. We conclude that while domain differences certainly play a factor, data imbalance appears to have more influence on parser performance.

In the future, our experiments need to be extended to a wider range of languages and target domains. Additionally, we will investigate strategies for dynamic learning of weights (Guo et al., 2019; Liu et al., 2019; Ming et al., 2019; Yim and Kim, 2020) for determining optimal loss weighting automatically, as well as more complex scheduling approaches (Kiperwasser and Ballesteros, 2018; Guo et al., 2018; Sébastien et al., 2018) to further improve performance.

Acknowledgements

The authors would like to thank Ines Rehbein for providing the tweeDe treebank, 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 first author is supported by the Swedish strategic research programme eSSENCE.

References

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.

Isabelle Augenstein and Anders Søgaard. 2017. Multi-task learning of keyphrase boundary classification. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 341–346, Vancouver, Canada.

Adrian Benton, Margaret Mitchell, and Dirk Hovy. 2017. Multitask learning for mental health conditions with limited social media data. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 152–162, Valencia, Spain.

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 (EACL)*, 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.

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

Alessandra Teresa Cignarella, Cristina Bosco, and Paolo Rosso. 2019. Presenting TWITTIRÒ-UD: An Italian Twitter treebank in Universal Dependencies. In *Proceedings of the Fifth International Conference on Dependency Linguistics (Depling, SyntaxFest 2019)*, pages 190–197, Paris, France.

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 (NAACL-HLT)*, pages 4171–4186, Minneapolis, MN.

Timothy Dozat and Christopher Manning. 2017. Deep biaffine attention for neural dependency parsing. In

5h International Conference on Learning Representations (ICLR 2017), Toulon, France.

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 (ACL-IJCNLP)*, pages 845–850, Beijing, China.

Robert French. 1999. Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences*, 3:128–135.

Daniel Fried, Nikita Kitaev, and Dan Klein. 2019. Cross-domain generalization of neural constituency parsers. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 323–330, Florence, Italy.

Han Guo, Ramakanth Pasunuru, and Mohit Bansal. 2019. AutoSeM: Automatic task selection and mixing in multi-task learning. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 3520–3531, Minneapolis, MN.

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*, pages 12–22, Osaka, Japan.

Michelle Guo, Albert Haque, De-An Huang, Serena Yeung, and Li Fei-Fei. 2018. Dynamic task prioritization for multitask learning. In *Computer Vision – ECCV 2018*, pages 282–299, Munich, Germany.

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 (NAACL-HLT)*, pages 127–137, Atlanta, GA.

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 (LREC)*, pages 5219–5224, Marseille, France.

Vidur Joshi, Matthew Peters, and Mark Hopkins. 2018. Extending a parser to distant domains using a few dozen partially annotated examples. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1190–1199, Melbourne, Australia.

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 (COLING)*, pages 2497–2508, Barcelona, Spain (Online).

Alex Kendall, Yarin Gal, and Roberto Cipolla. 2018. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7482–7491, Salt Lake City, UT.

Eliyahu Kiperwasser and Miguel Ballesteros. 2018. Scheduled multi-task learning: From syntax to translation. *Transactions of the Association for Computational Linguistics*, 6:225–240.

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 (ACL)*, pages 3499–3505, Florence, Italy.

Giwoong Lee, Eunho Yang, and Sung Ju Hwang. 2016. Asymmetric multi-task learning based on task relatedness and loss. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning (ICML)*, page 230–238, New York, NY.

Hae Beom Lee, Eunho Yang, and Sung Ju Hwang. 2018. Deep asymmetric multi-task feature learning. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, pages 2956–2964, Stockholm, Sweden.

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

Ying Li, Zhenghua Li, and Min Zhang. 2020. Semi-supervised domain adaptation for dependency parsing via improved contextualized word representations. In *Proceedings of the 28th International Conference on Computational Linguistics (COLING)*, pages 3806–3817, Barcelona, Spain (Online).

Zhenghua Li, Xue Peng, Min Zhang, Rui Wang, and Luo Si. 2019. Semi-supervised domain adaptation for dependency parsing. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 2386–2395, Florence, Italy.

Jian Liang, Ziqi Liu, Jiayu Zhou, Xiaoqian Jiang, Changshui Zhang, and Fei Wang. 2020. Model-protected multi-task learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Lukas Liebel and Marco Körner. 2018. Auxiliary tasks in multi-task learning. arXiv:1805.06334v2.

Shengchao Liu, Yingyu Liang, and Anthony Gitter. 2019. Loss-balanced task weighting to reduce negative transfer in multi-task learning. *Proceedings*

of the AAAI Conference on Artificial Intelligence, 33(01):9977–9978.

Minh-Thang Luong, Quoc Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. 2015. Multi-task sequence to sequence learning. In *Proceedings of ICLR*, San Juan, Puerto Rico.

Louis Martin, Benjamin Muller, Pedro Javier Ortiz Suárez, Yoann Dupont, Laurent Romary, Éric de la Clergerie, Djamé Seddah, and Benoît Sagot. 2020. CamemBERT: A tasty French language model. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 7203–7219, Online.

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 (EACL)*, pages 44–53, Valencia, Spain.

Sara Meftah, Nasredine Semmar, Mohamed-Ayoub Tahiri, Youssef Tamaazousti, Hassane Essafi, and Fatiha Sadat. 2020. Multi-task supervised pretraining for neural domain adaptation. In *Proceedings of the Eighth International Workshop on Natural Language Processing for Social Media*, pages 61–71, Online.

Zuheng Ming, Junshi Xia, Muhammad Muzzamil Luqman, Jean-Christophe Burie, and Kaixing Zhao. 2019. Dynamic multi-task learning for face recognition with facial expression. arXiv:1911.03281.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Jan Hajič, Christopher D. Manning, Sampo Pyysalo, Sebastian Schuster, Francis Tyers, and Daniel Zeman. 2020. Universal Dependencies v2: An evergrowing multilingual treebank collection. In *Proceedings of the 12th Language Resources and Evaluation Conference (LREC)*, pages 4034–4043, Marseille, France.

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:1912.01703.

Nanyun Peng and Mark Dredze. 2017. Multi-task domain adaptation for sequence tagging. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 91–100, Vancouver, Canada.

Ines Rehbein, Josef Ruppenhofer, and Bich-Ngoc Do. 2019. tweeDe – a Universal Dependencies treebank for German tweets. In *Proceedings of the 18th International Workshop on Treebanks and Linguistic Theories (TLT, SyntaxFest 2019)*, pages 100–108, Paris, France.

Sebastian Ruder, Joachim Bingel, Isabelle Augenstein, and Anders Søgaard. 2019. Latent multi-task architecture learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):4822–4829.

Manuela Sanguinetti, Cristina Bosco, Alberto Lavelli, Alessandro Mazzei, Oronzo Antonelli, and Fabio Tamburini. 2018. PoSTWITA-UD: an Italian Twitter treebank in Universal Dependencies. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC)*, Miyazaki, Japan.

Zeeshan Ali Sayyed and Daniel Dakota. 2021. Annotations matter: Leveraging multi-task learning to parse UD and SUD. In *Findings of the ACL: ACL-IJCNLP 2021*, Online.

Fynn Schröder and Chris Biemann. 2020. Estimating the influence of auxiliary tasks for multi-task learning of sequence tagging tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 2971–2985, Online.

Claudia Schulz, Steffen Eger, Johannes Daxenberger, Tobias Kahse, and Iryna Gurevych. 2018. Multi-task learning for argumentation mining in low-resource settings. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 35–41, New Orleans, LA.

Jean Sébastien, Orhan Firat, and Melvin Johnson. 2018. Adaptive scheduling for multi-task learning. In *Proceedings of the Continual Learning Workshop at 32nd Conference on Neural Information Processing Systems (NeurIPS)*, Montréal, Canada.

Anders Søgaard. 2017. Using hyperlinks to improve multilingual partial parsers. In *Proceedings of the 15th International Conference on Parsing Technologies*, pages 67–71, Pisa, Italy.

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 (ACL)*, 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 (ACL)*, pages 619–625, Melbourne, Australia.

Antti Virtanen, Jenna Kanerva, Rami Ilo, Jouni Luoma, Juhani Luotolahti, Tapio Salakoski, Filip Ginter, and Sampo Pyysalo. 2019. Multilingual is not enough: BERT for Finnish. arXiv:1912.07076.

Genta Indra Winata, Andrea Madotto, Chien-Sheng Wu, and Pascale Fung. 2018. Code-switching language modeling using syntax-aware multi-task

learning. In *Proceedings of the Third Workshop on Computational Approaches to Linguistic Code-Switching*, pages 62–67, Melbourne, Australia.

Sen Wu, Hongyang R. Zhang, and Christopher Ré. 2020. Understanding and improving information transfer in multi-task learning. In *8th International Conference on Learning Representations*, Online.

Jonghwa Yim and Sang Hwan Kim. 2020. Learning boost by exploiting the auxiliary task in multi-task domain. arXiv:2008.02043.

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 (ACL)*, pages 3295–3305, Online.

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 (ACL)*, pages 1557–1566, Berlin, Germany.