A Little Pretraining Goes a Long Way: A Case Study on Dependency Parsing Task for Low-resource Morphologically Rich Languages

Jivnesh Sandhan[1], **Amrith Krishna**[2], **Ashim Gupta**[3], **Laxmidhar Behera**[1,4] **and Pawan Goyal**[5]

[1]Dept. of Electrical Engineering, IIT Kanpur,
[2]Dept. of Computer Science and Technology, University of Cambridge,
[3]School of Computing, University of Utah, [4]Tata Consultancy Services,
[5]Dept. of Computer Science and Engineering, IIT Kharagpur

`jivnesh@iitk.ac.in, ak2329@cam.ac.uk, pawang@cse.iitkgp.ac.in`

Abstract

Neural dependency parsing has achieved remarkable performance for many domains and languages. The bottleneck of massive labeled data limits the effectiveness of these approaches for low resource languages. In this work, we focus on dependency parsing for morphological rich languages (MRLs) in a low-resource setting. Although morphological information is essential for the dependency parsing task, the morphological disambiguation and lack of powerful analyzers pose challenges to get this information for MRLs. To address these challenges, we propose simple auxiliary tasks for pretraining. We perform experiments on 10 MRLs in low-resource settings to measure the efficacy of our proposed pretraining method and observe an average absolute gain of 2 points (UAS) and 3.6 points (LAS).[1]

1 Introduction

Dependency parsing has greatly benefited from neural network-based approaches. While these approaches simplify the parsing architecture and eliminate the need for hand-crafted feature engineering (Chen and Manning, 2014; Dyer et al., 2015; Kiperwasser and Goldberg, 2016; Dozat and Manning, 2017; Kulmizev et al., 2019), their performance has been less exciting for several morphologically rich languages (MRLs) and low-resource languages (More et al., 2019; Seeker and Çetinoğlu, 2015). In fact, the need for large labeled treebanks for such systems has adversely affected the development of parsing solutions for low-resource languages (Vania et al., 2019). Zeman et al. (2018) observe that data-driven parsing on 9 low resource treebanks resulted not only in low scores but those outputs "are hardly useful for downstream applications".

Several approaches have been suggested for improving the parsing performance of low-resource languages. This includes data augmentation strategies, cross-lingual transfer (Vania et al., 2019) and using unlabelled data with semi-supervised learning (Clark et al., 2018) and self-training (Rotman and Reichart, 2019). Further, incorporating morphological knowledge substantially improves the parsing performance for MRLs, including low-resource languages (Vania et al., 2018; Dehouck and Denis, 2018). This aligns well with the linguistic intuition of the role of morphological markers, especially that of case markers, in deciding the syntactic roles for the words involved (Wunderlich and Lakämper, 2001; Sigursson, 2003; Kittilä et al., 2011). However, obtaining the morphological tags for input sentences during run time is a challenge in itself for MRLs (More et al., 2019) and use of predicted tags from taggers, if available, often hampers the performance of these parsers. In this work, we primarily focus on one such morphologically-rich low-resource language, Sanskrit.

We propose a simple pretraining approach, where we incorporate encoders from simple auxiliary tasks by means of a gating mechanism (Sato et al., 2017). This approach outperforms multi-task training and transfer learning methods under the same low-resource data conditions (∼500 sentences). The proposed approach when applied to Dozat et al. (2017), a neural parser, not only obviates the need for providing morphological tags as input at runtime, but also outperforms its original configuration that uses gold morphological tags as input. Further, our method performs close to DCST (Rotman and Reichart, 2019), a self-training based extension of Dozat et al. (2017), which uses gold morphological tags as input for training.

To measure the efficacy of the proposed method, we further perform a series of experiments on 10 MRLs in low-resource settings and show 2 points

[1]Code and data available at: `https://github.com/jivnesh/LCM`

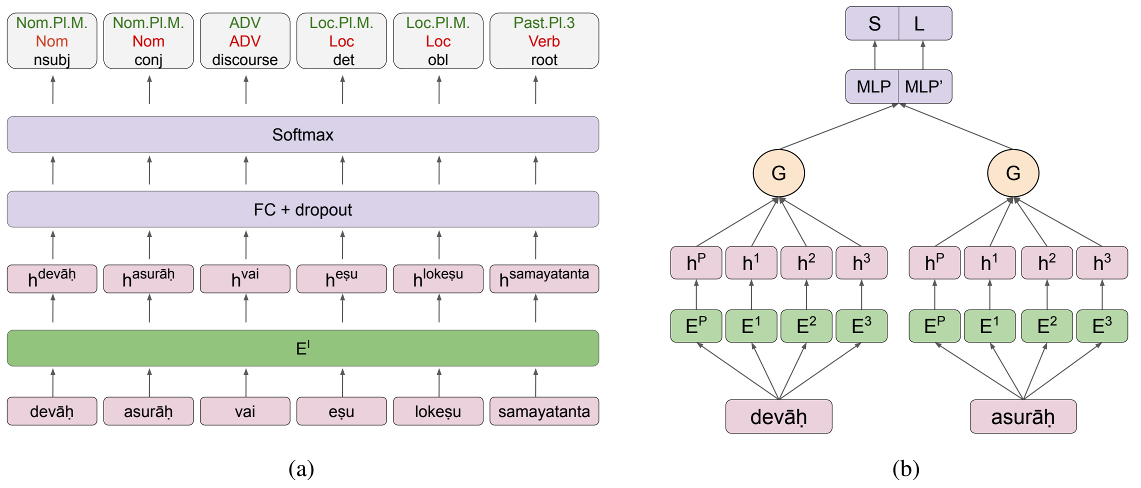

Figure 1: Illustration of proposed architecture for a Sanskrit sequence. English translation: "Demigods and demons had tried with equal effort for these planets". (a) Pretraining step: For an input word sequence, tagger predicts labels as per three proposed auxiliary tasks, namely, Morphological Tag (green), Case Tag (red) and Label Tag (black). (b) Parser with gating: $E^{(P)}$ is encoder of a neural parser like Dozat and Manning (2017) and $E^{(1)-(3)}$ are the encoders pre-trained with proposed auxiliary tasks. Gating mechanism combines representations of all the encoders which, for each word pair, is passed to two MLPs to predict the probability of arc score (S) and label (L).

and 3.6 points average absolute gain (§ 3.1) in terms of UAS and LAS, respectively. Our proposed method also outperforms multilingual BERT (Devlin et al., 2019, mBERT) based multi-task learning model (Kondratyuk and Straka, 2019, Udify) for the languages which are not covered in mBERT (§ 3.4).

2 Pretraining approach

Our proposed pretraining approach essentially attempts to combine word representations from encoders trained on multiple sequence level supervised tasks, as auxiliary tasks, with that of the default encoder of the neural dependency parser. While our approach is generic and can be used with any neural parser, we use BiAFFINE parser (Dozat and Manning, 2017), hence forth referred to as BiAFF, in our experiments.This is a graph-based neural parser that makes use of biaffine attention and a biaffine classifier.[2] Figure 1 illustrates the proposed approach using an example sequence from Sanskrit. Our pipeline-based approach consists of two steps: (1) Pretraining step (2) Integration step. Figure 1a describes the pretraining step with three auxiliary tasks to pretrain the corresponding encoders $E^{(1)-(3)}$. Finally, in the integration step, these pretrained encoders along with the encoder for the BiAFF model $E^{(P)}$ are then combined using a gating mechanism (1b) as employed in Sato et al. (2017). [3]

All the auxiliary tasks are trained independently as separate models, but using the same architecture and hyperparameter settings which differ only in terms of the output label they use. The models for the pretraining components are trained using BiLSTM encoders, similar to the encoders in Dozat and Manning (2017) and then decoded using two fully connected layers, followed by a softmax layer (Huang et al., 2015). These sequential tasks involve prediction of the morphological tag **(MT)**, dependency label (relation) that each word holds with its head (**LT**) and further we also consider task where the case information of each nominal forms the output label **(CT)**. Other grammatical categories did not show significant improvements over the case (§ 3.2). This aligns well with the linguistic paradigm that the case information plays an important role in deciding the syntactic role that a nominal can be assigned in the sentence. For words with no case-information, we predict their coarse POS tags. Here, the morphological information is automatically leveraged using the pre-trained encoders, and thus during runtime the morphological tags need not be provided as inputs. It also helps in reducing the gap between UAS and LAS (§ 3.1).

[2]More details can be found in supplemental (§ A.1).

[3]Our proposed approach is inspired from Rotman and Reichart (2019).

3 Experiments

Data and Metric: We use 500, 1,000 and 1,000 sentences from the Sanskrit Treebank Corpus (Kulkarni et al., 2010, STBC) as the training, dev and test data respectively for all the models. For the proposed auxiliary tasks, all the sequence taggers are trained with additional previously unused 1,000 sentences from STBC along with the training sentences used for the dependency parsing task. For the Label Tag (LT) prediction auxiliary task, we do not use gold dependency information; rather we use predicted tags from BiAFF parser. For the remaining auxiliary tasks, we use gold standard morphological information.

For all the models, input representation consists of FastText (Grave et al., 2018)[4] embedding of 300-dimension and convolutional neural network (CNN) based 100-dimensional character embedding (Zhang et al., 2015). For character level CNN architecture, we use following setting: 100 number of filters with kernel size equal to 3. We use standard Unlabelled and Labelled Attachment Scores (UAS, LAS) to measure the parsing performance and use t-test for statistical significance (Dror et al., 2018).

For STBC treebank, the original data does not have morphological tag entry, so the Sanskrit Heritage reader (Huet and Goyal, 2013; Goyal and Huet, 2016) is used to obtain all the possible morphological analysis and only those sentences are chosen which do not have any word showing homonymy or syncretism (Krishna et al., 2020). For other MRLs, we restrict to the same training setup as Sanskrit and use 500 annotated sentences as labeled data for training. Additionally, we use 1000 sentences with morphological information as unlabelled data for pretraining sequence taggers.[5] We use all the sentences present in original development and test split data for development and test data. For languages where multiple treebanks are available, we chose only one available treebank to avoid domain shift. Note that STBC adopts a tagging scheme based on the grammatical tradition of Sanskrit, specifically based on Kāraka (Kulkarni and Sharma, 2019; Kulkarni et al., 2010), while the other MRLs including Sanskrit-Vedic use UD.

[4]`https://fasttext.cc/docs/en/crawl-vectors.html`

[5]The predicted relations on unlabelled data by the model trained with 500 samples are used for Label Tagging task.

Hyper-parameters: We utilize the BiAFFINE parser (BiAFF) implemented by Ma et al. (2018). We employ the following hyper-parameter setting for pretraining sequence taggers and base parser BiAFF: the batch size of 16, number of epochs as 100, and a dropout rate of 0.33 with a learning rate equal to 0.002. The hidden representation generated from n-Stacked-LSTM layers of size 1,024 is passed through two fully connected layers of size 128 and 64. Note that LCM and MTL models use 2-Stacked LSTMs. We keep all the remaining parameters the same as that of Ma et al. (2018).

For all TranSeq variants, one BiLSTM layer is added on top of three augmented pretrained layers from an off-the-shelf morphological tagger (Gupta et al., 2020) to learn task-specific features. In TranSeq-FEA, the dimension of the non-linearity layer of the adaptor module is 256, and in TranSeq-UF, after every 20 epochs, one layer is unfrozen from top to down fashion. In TranSeq-DL, the learning rate is decreased from top to down by a factor of 1.2. We have used default parameters to train Hierarchical Tagger [6] and baseline models.

Models: All our experiments are performed as augmentations on two off the shelf neural parsers, BiAFF (Dozat and Manning, 2017) and Deep Contextualized Self-training (DCST), which integrates self-training with BiAFF (Rotman and Reichart, 2019).[7] Hence their default configurations become the baseline models **(Base)**. We also use a system that simultaneously trains the BiAFF (and DCST) model for dependency parsing along with the sequence level case prediction task in a multi task setting **(MTL)**. For MTL model, we also experiment with morphological tagging, as an auxiliary task. However, we do not find significant improvement in performance compared to case tagging. Hence, we consider case tagging as an auxiliary task to avoid sparsity issue due to the monolithic tag scheme for morphological tagging. As a transfer learning variant (**TranSeq**), we extract first three layers from a hierarchical multi-task morphological tagger (Gupta et al., 2020), trained on 50k examples from DCS (Hellwig, 2010). Here each layer corresponds to different grammatical categories, namely, number, gender and case. Note that number of randomly initialised encoder layers in BiAFF (and DCST) are now reduced from 3 to

[6]`https://github.com/ashim95/sanskrit-morphological-taggers`

[7]We describe the baseline models in supplemental (§ A).

1. We fine-tune these layers with default learning rate and experiment with four different fine-tuning schedules.[8] Finally, our proposed configuration (in §2) is referred to as the **LCM** model.[9] We also train a version each of the base models which expects morphological tags as input and is trained with gold morphological tags. During runtime, we report two different settings, one which uses predicted tags as input (**Predicted MI**) and other that uses gold tag as input (**Oracle MI**). We obtain the morphological tags from a Neural CRF tagger (Yang and Zhang, 2018) trained on our training data. Oracle MI will act as an upper-bound on the reported results.

3.1 Results

Table 6 presents results for dependency parsing on Sanskrit. We observe that BiAFF + LCM outperforms all corresponding BiAFF models including Oracle MI. This is indeed a serendipitous outcome as one would expect Oracle MI to be an upper bound owing to its use of gold morphological tags at runtime. The DCST variant of our pretraining approach is also the best among its peers, although the performance of Oracle MI model in this case is indeed the upper bound.

	BiAFF		DCST	
Model	UAS	LAS	UAS	LAS
Base	70.67	56.85	73.23	58.64
Predicted MI	69.02	53.11	71.15	51.75
MTL	70.85	57.93	73.04	59.12
TranSeq	71.46	60.58	74.58	62.70
LCM	**75.91**	**64.87**	**75.75**	**64.28**
Oracle MI	*74.08*	*62.48*	*76.66*	*66.35*

Table 1: Results on Sanskrit dependency parsing. Oracle MI is an upper bound and is not comparable.

On the other hand, using predicted morphological tags instead of gold tags at run time degrades results drastically, especially for LAS, possibly due to the cascading effect of incorrect morphological information (Nguyen and Verspoor, 2018). This shows that morphological information is essential in filling the UAS-LAS gap and substantiates the need for pretraining to incorporate such knowledge even when it is not available at run time. Interestingly, both MTL, and TranSeq, show improvements as compared to the base models, though do not match with that of our pretraining approach. In our experiments, the pretraining approach, even with *a little training data*, clearly outperforms the other approaches.

[8]Refer supplemental (§ B) for variations of TranSeq.

[9]LCM denotes Label, Case and Morph tagging schemes.

Training Size	BiAFF UAS/LAS	DCST UAS/LAS	BiAFF+LCM UAS/LAS
100	58.0/42.3	64.0/44.0	**70.4/59.9**
500	70.7/56.9	73.2/58.6	**75.9/64.9**
750	74.0/61.8	75.2/62.3	**77.3/66.8**
1000	74.4/62.9	76.0/64.1	**77.9/67.3**
1250	75.6/64.7	76.7/65.2	**78.5/68.3**

Table 2: Performance as a function of training set size.

Ablation: We perform further analysis on Sanskrit to study the effect of training set size as well as the impact of various tagging schemes as auxiliary tasks. First, we evaluate the impact on performance as a function of the training size (Table 2). Noticeably, for training size 100, we observe a 12 (UAS) and 17 (LAS) points increase for BiAFF+LCM over BiAFF, demonstrating the effectiveness of our approach in a very low-resource setting. This improvement is consistent for larger training sizes, though the gain reduces.

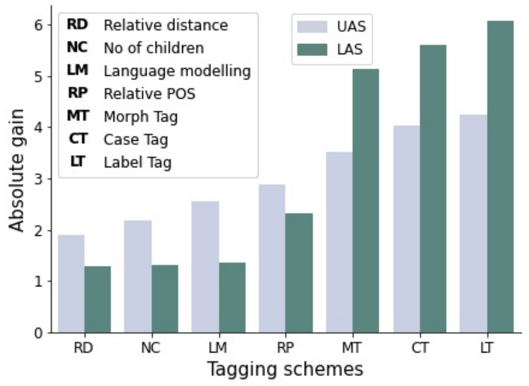

Figure 2: Comparison of proposed tagging schemes (MT, CT, LT) with those in DCST (RD, NC, LM, RP).

In Figure 2, we compare our tagging schemes with those used in self-training of DCST, namely, Relative Distance from root (RD), Number of Children for each word (NC), Language Modeling (LM) objective where task is to predict next word in sentence, and Relative POS (RP) of modifier from root word. Here, we integrate each pretrained model (corresponding to each tagging scheme) individually on top of the BiAFF baseline using the gating mechanism and report the absolute gain over the BiAFF in terms of UAS and LAS metric. Inter-

	eu		el		sv		pl		ru		avg	
Model	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS
BiAFF	63.18	54.52	79.64	75.01	71.73	64.83	78.33	70.83	73.98	67.42	73.37	66.52
DCST	69.60	60.65	83.48	78.61	77.03	69.62	81.40	73.09	78.61	72.07	78.02	70.81
DCST+MTL	70.38	61.52	83.74	79.31	76.70	69.88	81.25	73.34	78.46	72.08	78.11	71.23
DCST+TranSeq	70.70	62.96	84.69	80.37	77.30	70.85	82.84	75.02	78.95	73.18	78.90	72.48
BiAFF+LCM	**72.40**	**65.50**	**86.56**	**83.18**	77.95	72.20	**84.08**	**77.65**	79.97	74.47	80.20	74.60
DCST+LCM	72.01	65.33	85.94	82.22	**78.72**	**73.04**	83.83	77.63	**80.62**	**75.26**	**80.22**	**74.70**
BiAFF+Oracle MI	*72.16*	*66.08*	*83.05*	*79.81*	*76.50*	*71.17*	*83.27*	*77.83*	*77.83*	*73.13*	*78.56*	*73.60*
DCST+Oracle MI	*77.47*	*71.55*	*85.99*	*82.72*	*80.33*	*75.00*	*86.03*	*80.46*	*82.21*	*77.54*	*82.41*	*77.45*

	ar		hu		fi		de		cs		avg	
Model	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS
BiAFF	76.24	68.07	70.00	62.81	60.93	50.68	67.77	59.94	65.75	57.43	70.30	62.62
DCST	79.05	71.18	74.62	67.00	66.04	54.76	73.22	65.18	74.15	65.52	75.61	67.70
DCST+Predicted MI	77.17	66.63	61.55	36.18	56.48	39.67	65.31	47.12	72.03	58.37	68.72	52.61
DCST+MTL	79.35	71.37	74.49	66.70	66.30	55.29	73.98	66.05	74.66	65.95	75.84	67.99
DCST+TranSeq-FT	79.66	72.17	75.22	68.25	67.04	56.57	74.66	67.27	**75.15**	67.02	76.40	69.11
BiAFF+LCM	**79.68**	**72.55**	**76.15**	**69.53**	69.05	59.41	75.85	68.80	74.94	**67.58**	76.91	70.13
DCST+LCM	79.60	72.38	75.71	68.93	**69.15**	**60.06**	**76.12**	**69.20**	74.81	67.54	**76.99**	**70.22**
BiAFF+Oracle MI	*77.52*	*71.46*	*75.89*	*70.63*	*70.80*	*64.64*	*72.63*	*66.53*	*72.39*	*66.22*	*74.99*	*69.20*
DCST+Oracle MI	*80.43*	*74.79*	*78.43*	*73.19*	*75.30*	*68.90*	*77.70*	*71.66*	*78.54*	*72.38*	*79.09*	*73.40*

Table 3: Evaluation on 10 MRLs. Results of BiAFF+LCM and DCST+LCM are statistically significant compared to strong baseline DCST as per t-test ($p < 0.01$). Last two columns denote the average performance. Models using Oracle MI are not comparable.

estingly, our proposed tagging schemes, with an improvement of 3-4 points (UAS) and 5-6 points (LAS), outperform those of DCST and help bridge the gap between UAS-LAS.

3.2 Additional auxiliary tasks

With our proposed pretraining approach, we experiment with using the prediction of different grammatical categories as auxiliary tasks, namely, Number Tagging (NT), Person Tagging (PT), and Gender Tagging (GT). As the results in table **??** demonstrate, the improvements observed in these cases are much smaller than those for our proposed auxiliary tasks. Similar results are observed when considering other auxiliary tasks (see table **??**). We find that combining these auxiliary tasks with our proposed ones did not provide any notable improvements. One possible reason for under performance of these tagging schemes compared to the proposed ones could be that either when the training set is small, sequence taggers are not able to learn discriminative features only from surface form of words (F-score is less than 40 in all such cases in table **??**) or the learned features are not helpful for the dependency parsing task.

3.3 Experiments on other MRLs

We choose 10 additional MRLs from Universal Dependencies (UD) dataset (McDonald et al., 2013; Nivre et al., 2016), namely, Arabic (ar), Czech (cs), German (de), Basque (eu), Greek (el), Finnish (fi), Hungarian (hu), Polish (pl), Russian (ru) and Swedish (sv).[10] Then we train them in low-resource setting (500 examples) to investigate the applicability of our approach for these MRLs.

For all MRLs, the trend is similar to what is observed for Sanskrit. While all four models improve over both the baselines, BiAFF+LCM and DCST+LCM consistently turn out to be the best configurations. Note that these models are not directly comparable to Oracle MI models since Oracle MI models use gold morphological tags instead of the predicted ones. The performance of BiAFF+LCM and DCST+LCM is also comparable. Across all 11 MRLs, BiAFF+LCM shows the average absolute gain of 2 points (UAS) and 3.6 points (LAS) compared to the strong baseline DCST.

[10]We choose MRLs that have the explicit morphological information with following grammatical categories: case, number, gender, and tense.

Auxiliary Task	F-score	Gain
Relative Distance (RD)	58.71	1.9/1.3
No of children (NC)	52.82	2.2/1.3
Relative POS (RP)	46.52	2.9/2.3
Lang Model (LM)	41.54	2.6/1.4
Coarse POS (CP)	13.02	1.6/0.8
Head Word (HW)	40.12	1.5/0.4
POS Head Word (PHW)	38.98	2.0/1.2
Number Tagging (NT)	13.33	1.9/0.9
Person Tagging (PT)	12.27	1.6/0.7
Gender Tagging (GT)	0.28	1.3/0.2
Morph Tagging (MT)	62.84	3.5/5.1
Case Tagging (CT)	73.51	4.0/5.6
Label Tagging (LT)	71.51	4.2/6.0

Table 4: Comparison of different auxiliary tasks. F-score: Task performance, Gain: Absolute gain (when integrated with BiAFF) in terms of UAS/LAS score compared to BiAFF scores.

3.4 Comparison with mBERT Pretraining

We compare the proposed method with multilingual BERT (Devlin et al., 2019, mBERT) based multi-task learning model (Kondratyuk and Straka, 2019, Udify). This single model trained on 124 UD treebanks covers 75 different languages and produces state of the art results for many of them. Multilingual BERT leverages large scale pretraining on wikipedia for 104 languages.

Lang	BiAFF	BiAFF+LCM	Udify
Basque	63.2/54.5	72.4/65.5	**76.6/69.0**
German	67.7/60.0	75.8/68.8	**83.7/77.5**
Hungarian	70.0/62.8	76.2/69.5	**84.4/76.7**
Greek	69.6/75.0	86.6/83.2	**90.6/87.0**
Polish	78.3.70.8	84.1/77.7	**90.7/85.0**
Sanskrit	70.7/56.8	**75.9/64.9**	69.4/53.2
Sanskrit-Vedic	56.0/42.3	**61.6/48.0**	47.4/28.3
Wolof	75.3/67.8	**78.4/71.3**	70.9/60.6
Gothic	61.7/53.3	**69.6/61.4**	63.4/52.2
Coptic	84.3/80.2	**86.2/82.7**	32.7/14.3

Table 5: The proposed method outperforms Udify for the languages (down) not covered in mBERT and under performs for the languages (top) which are covered in mBERT.

In our experiments, we find that Udify outperforms the proposed method for languages covered during mBERT's pretraining. Notably, not only the proposed method but also a simple BiAFF parser with randomly initialized embedding outperforms Udify (Table 5) for languages which not available in mBERT. Out of 7,000 languages, only a handful of languages can take advantage of mBERT pretraining (Joshi et al., 2020) which substantiates the need of our proposed pretraining scheme.

4 Conclusion

In this work, we focused on dependency parsing for low-resource MRLs, where getting morphological information itself is a challenge. To address low-resource nature and lack of morphological information, we proposed a simple pretraining method based on sequence labeling that does not require complex architectures or massive labelled or unlabelled data. We show that little supervised pretraining goes a long way compared to transfer learning, multi-task learning, and mBERT pretraining approaches (for the languages not covered in mBERT). One primary benefit of our approach is that it does not rely on morphological information at run time; instead this information is leveraged using the pretrained encoders. Our experiments across 10 MRLs showed that proposed pretraining provides a significant boost with an average 2 points (UAS) and 3.6 points (LAS) absolute gain compared to DCST.

Acknowledgements

We thank Amba Kulkarni for providing Sanskrit dependency treebank data and the anonymous reviewers for their constructive feedback towards improving this work. The work of the first author is supported by the TCS Fellowship under the Project TCS/EE/2011191P.

References

Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 740–750, Doha, Qatar. Association for Computational Linguistics.

Kevin Clark, Minh-Thang Luong, Christopher D. Manning, and Quoc Le. 2018. Semi-supervised sequence modeling with cross-view training. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1914–1925, Brussels, Belgium. Association for Computational Linguistics.

Mathieu Dehouck and Pascal Denis. 2018. A framework for understanding the role of morphology in universal dependency parsing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2864–2870, Brussels, Belgium. Association for Computational Linguistics.

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, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Timothy Dozat and Christopher D. Manning. 2017. Deep biaffine attention for neural dependency parsing. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Timothy Dozat, Peng Qi, and Christopher D Manning. 2017. Stanford's graph-based neural dependency parser at the conll 2017 shared task. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 20–30.

Rotem Dror, Gili Baumer, Segev Shlomov, and Roi Reichart. 2018. The hitchhiker's guide to testing statistical significance in natural language processing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1383–1392, Melbourne, Australia. Association for Computational Linguistics.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 334–343, Beijing, China. Association for Computational Linguistics.

Jack Edmonds. 1967. Optimum branchings. *Journal of Research of the national Bureau of Standards B*, 71(4):233–240.

Bjarke Felbo, Alan Mislove, Anders Søgaard, Iyad Rahwan, and Sune Lehmann. 2017. Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1615–1625, Copenhagen, Denmark. Association for Computational Linguistics.

Robert M French. 1999. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3(4):128–135.

Pawan Goyal and Gérard Huet. 2016. Design and analysis of a lean interface for sanskrit corpus annotation. *Journal of Language Modelling*, 4(2):145–182.

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

Ashim Gupta, Amrith Krishna, Pawan Goyal, and Oliver Hellwig. 2020. Evaluating neural morphological taggers for sanskrit. In *Proceedings of the 17th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, Seattle, USA. Association for Computational Linguistics.

Oliver Hellwig. 2010. Dcs-the digital corpus of sanskrit. *Heidelberg (2010-2020). URL http://www.sanskrit-linguistics.org/dcs/index.php*.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for NLP. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2790–2799, Long Beach, California, USA. PMLR.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339, Melbourne, Australia. Association for Computational Linguistics.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

Gérard Huet and Pawan Goyal. 2013. Design of a lean interface for sanskrit corpus annotation. *Proceedings of ICON*, pages 177–186.

Pratik Joshi, Sebastin Santy, Amar Budhiraja, Kalika Bali, and Monojit Choudhury. 2020. The state and fate of linguistic diversity and inclusion in the NLP world. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6282–6293, Online. Association for Computational Linguistics.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional LSTM feature representations. *Transactions of the Association for Computational Linguistics*, 4:313–327.

Seppo Kittilä, Katja Västi, and Jussi Ylikoski. 2011. Introduction to case, animacy and semantic roles. *Case, animacy and semantic roles*, 99:1–26.

Dan Kondratyuk and Milan Straka. 2019. 75 languages, 1 model: Parsing Universal Dependencies universally. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2779–2795, Hong Kong, China. Association for Computational Linguistics.

Amrith Krishna, Bishal Santra, Ashim Gupta, Pavankumar Satuluri, and Pawan Goyal. 2020. A graph based framework for structured prediction tasks in sanskrit. *Computational Linguistics*, 46(4):1–63.

Amba Kulkarni, Sheetal Pokar, and Devanand Shukl. 2010. Designing a constraint based parser for sanskrit. In *International Sanskrit Computational Linguistics Symposium*, pages 70–90. Springer.

Amba Kulkarni and Dipti Sharma. 2019. Pāinian syntactico-semantic relation labels. In *Proceedings of the Fifth International Conference on Dependency Linguistics (Depling, SyntaxFest 2019)*, pages 198–208, Paris, France. Association for Computational Linguistics.

Artur Kulmizev, Miryam de Lhoneux, Johannes Gontrum, Elena Fano, and Joakim Nivre. 2019. Deep contextualized word embeddings in transition-based and graph-based dependency parsing - a tale of two parsers revisited. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2755–2768, Hong Kong, China. Association for Computational Linguistics.

Xuezhe Ma, Zecong Hu, Jingzhou Liu, Nanyun Peng, Graham Neubig, and Eduard Hovy. 2018. Stack-pointer networks for dependency parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1403–1414, Melbourne, Australia. Association for Computational Linguistics.

Michael McCloskey and Neal J. Cohen. 1989. Catastrophic interference in connectionist networks: The sequential learning problem. In Gordon H. Bower, editor, *Psychology of Learning and Motivation*, volume 24, pages 109 – 165. Academic Press.

Ryan McDonald, Joakim Nivre, Yvonne Quirmbach-Brundage, Yoav Goldberg, Dipanjan Das, Kuzman Ganchev, Keith Hall, Slav Petrov, Hao Zhang, Oscar Täckström, et al. 2013. Universal dependency annotation for multilingual parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 92–97.

Amir More, Amit Seker, Victoria Basmova, and Reut Tsarfaty. 2019. Joint transition-based models for morpho-syntactic parsing: Parsing strategies for MRLs and a case study from modern Hebrew. *Transactions of the Association for Computational Linguistics*, 7:33–48.

Dat Quoc Nguyen and Karin Verspoor. 2018. An improved neural network model for joint POS tagging and dependency parsing. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 81–91, Brussels, Belgium. Association for Computational Linguistics.

Joakim Nivre, Marie-Catherine De Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, et al. 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.

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. Association for Computational Linguistics.

Guy Rotman and Roi Reichart. 2019. Deep contextualized self-training for low resource dependency parsing. *Transactions of the Association for Computational Linguistics*, 7:695–713.

Motoki Sato, Hitoshi Manabe, Hiroshi Noji, and Yuji Matsumoto. 2017. Adversarial training for cross-domain universal dependency parsing. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 71–79, Vancouver, Canada. Association for Computational Linguistics.

Wolfgang Seeker and Özlem Çetinoğlu. 2015. A graph-based lattice dependency parser for joint morphological segmentation and syntactic analysis. *Transactions of the Association for Computational Linguistics*, 3:359–373.

Halldór Ármann Sigursson. 2003. Case: abstract vs. morphological. *New perspectives on case theory*, pages 223–268.

Asa Cooper Stickland and Iain Murray. 2019. BERT and PALs: Projected attention layers for efficient adaptation in multi-task learning. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 5986–5995, Long Beach, California, USA. PMLR.

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. Association for Computational Linguistics.

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

Dieter Wunderlich and Renate Lakämper. 2001. On the interaction of structural and semantic case. *Lingua*, 111(4-7):377–418.

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

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. Association for Computational Linguistics.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 649–657. Curran Associates, Inc.

Supplemental Material

A Baselines

A.1 BiAFFINE Parser (BiAFF)

BiAFF (Dozat and Manning, 2017) is a graph-based dependency parsing approach similar to Kiperwasser and Goldberg (2016). It uses biaffine attention instead of using a traditional MLP-based attention mechanism. For input vector $\vec{h}$, the affine classifier is expressed as $W\vec{h}+b$, while the biaffine classifier is expressed as $W'(W\vec{h}+b)+b'$. The choice of biaffine classifier facilitates the key benefit of representing the prior probability of word j to be head and the likelihood of word i getting word j as the head. In this system, during training, each modifier in the predicted tree has the highest-scoring word as the head. This predicted tree need not be valid. However, at test time, to generate a valid tree MST algorithm (Edmonds, 1967) is used on the arc scores.

A.2 Deep Contextualized Self-training (DCST)

Rotman and Reichart (2019) proposed a self-training method called Deep Contextualized Self-training (DCST).[11] In this system, the base parser BiAFF (Dozat and Manning, 2017) is trained on the labelled dataset. Then this trained base parser is applied to the unlabelled data to generate automatically labelled dependency trees. In the next step, these automatically-generated trees are transformed into one or more sequence tagging schemes. Finally, the ensembled parser is trained on manually labelled data by integrating base parser with learned representation models. The gating mechanism proposed by Sato et al. (2017) is used to integrate different tagging schemes into the ensembled parser. This approach is in line with the representation models based on language modeling related tasks (Peters et al., 2018; Devlin et al., 2019). In summary, DCST demonstrates a novel approach to transfer information learned on labelled data to unlabelled data using sequence tagging schemes such that it can be integrated into final ensembled parser via word embedding layers.

B Experiments on TranSeq Variants

In TranSeq variations, instead of pretraining with three auxiliary tasks, we use a hierarchical multi-task morphological tagger (Gupta et al., 2020) trained on 50k training data from DCS (Hellwig, 2010). In TranSeq setting, we extract the first three layers from this tagger and augment them in baseline models and experiment with five model sub-variants. To avoid catastrophic forget-

	BiAFF		DCST	
Model	UAS	LAS	UAS	LAS
Base	70.67	56.85	73.23	58.64
Base⋆	69.35	52.79	72.31	54.82
TranSeq-FE	66.54	55.46	71.65	60.10
TranSeq-FEA	69.50	58.48	73.48	61.52
TranSeq-UF	70.60	59.74	73.55	62.39
TranSeq-DL	71.40	60.58	74.52	62.73
TranSeq-FT	**71.46**	**60.58**	**74.58**	**62.70**
Oracle MI	*74.08*	*62.48*	*76.66*	*66.35*

Table 6: Ablation analysis for TranSeq variations. Oracle MI is not comparable. It can be considered as upper bound for TranSeq.

[11] https://github.com/rotmanguy/DCST

ting (McCloskey and Cohen, 1989; French, 1999), we gradually increase the capacity of adaptations for each variant. **TranSeq-FE:** Freeze the pre-trained layers and use them as Feature Extractors (FE). **TranSeq-FEA:** In the feature extractor setting, we additionally integrate adaptor modules (Houlsby et al., 2019; Stickland and Murray, 2019) in between two consecutive pre-trained layers. **TranSeq-UF:** Gradually Unfreeze (UF) these three pre-trained layers in the top to down fashion (Felbo et al., 2017; Howard and Ruder, 2018). **TranSeq-DL:** In this setting, we use discriminative learning (DL) rate (Howard and Ruder, 2018) for pre-trained layers, i.e., decreasing the learning rate as we move from top-to-bottom layers. **TranSeq-FT:** We fine-tune (FT), pre-trained layers with default learning rate used by Dozat and Manning (2017).

In the TranSeq setting, as we move down across its sub-variants in Table 6, performance improves gradually, and TranSeq-FT configuration shows the best performance with 1-2 points improvement over Base. The Base$\star$ has one additional LSTM layer compared to Base such that the number of parameters are same as that of TranSeq-FT variation. The performance of Base$\star$ decreases compared to Base but TranSeq-FT outperforms Base. This shows that transfer learning definitely helps to boost the performance.