The SLT-Interactions Parsing System at the CoNLL 2018 Shared Task

Riyaz Ahmad Bhat
Interactions LLC
Bengaluru, India
rbhat@interactions.com

Irshad Ahmad Bhat
LTRC, IIITH
Hyderabad, India
irshad.bhat@research.iiit.ac.in

Srinivas Bangalore
Interactions LLC
New Jersey, United States
sbangalore@interactions.com

Abstract

This paper describes our system (SLT-Interactions) for the CoNLL 2018 shared task: *Multilingual Parsing from Raw Text to Universal Dependencies*. Our system performs three main tasks: word segmentation (only for few treebanks), POS tagging and parsing. While segmentation is learned separately, we use neural stacking for joint learning of POS tagging and parsing tasks. For all the tasks, we employ simple neural network architectures that rely on long short-term memory (LSTM) networks for learning task-dependent features. At the basis of our parser, we use an arc-standard algorithm with Swap action for general non-projective parsing. Additionally, we use neural stacking as a knowledge transfer mechanism for cross-domain parsing of low resource domains. Our system shows substantial gains against the UDPipe baseline, with an average improvement of 4.18% in LAS across all languages. Overall, we are placed at the 12^{th} position on the official test sets.

1 Introduction

Our system for the CoNLL 2018 shared task (Zeman et al., 2018) contains the following modules: word segmentation, part-of-speech (POS) tagging and dependency parsing. In some cases, we also use a transliteration module to transcribe data into Roman form for efficient processing.

- **Segmentation** We mainly use this module to identify word boundaries in certain languages such as Chinese where space is not used as a boundary marker.
- **POS tagging** For all the languages, we only focus on universal POS tags while ignoring language specific POS tags and morphological features.
- **Dependency parsing** We use an arc-standard transition system (Nivre, 2003) with an additional Swap action for unrestricted parsing (Nivre, 2009).

We rely on UDPipe 1.2 (Straka and Straková, 2017) for tokenization for almost all the treebanks except for Chinese and Japanese where we observed that the UDPipe segmentation had an adverse effect on parsing performance as opposed to gold segmentation on the development sets. Moreover, we also observed that training a separate POS tagger was also beneficial as the UDPipe POS tagger had slightly lower performance in some languages. However, other than tokenization, we ignored other morphological features predicted by UDPipe and didn't explore their effect on parsing.

Additionally, we use knowledge transfer approaches to enhance the performance of parsers trained on smaller treebanks. We leverage related treebanks (other treebanks of the same language) using neural stacking for learning better cross-domain parsers. We also trained a generic character-based parsing system for languages that have neither in-domain nor cross-domain training data.

Upon the official evaluation on 82 test sets, our system (SLT-Interactions) obtained the 12^{th} position in the parsing task and achieved an average improvement of 4.18% in LAS over the UDPipe baseline.

Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies, pages 153–159
Brussels, Belgium, October 31 – November 1, 2018.
https://doi.org/10.18653/v1/K18-2015

2 System Architecture

2.1 Text Processing

Given the nature of the shared task, sentence and word segmentation are the two major prerequisite tasks needed for parsing the evaluation data. For most of the languages, we rely on UDPipe for both sentence segmentation and word segmentation. However, in few languages such as Chinese and Japanese which do not use white space as explicit word boundary marker, we build our own word segmentation models. Our segmentation models use a simple neural network classifier that relies on character bidirectional LSTM (Bi-LSTM) representations of a focus character to produce a probabilistic distribution over two boundary markers: **B**egining of a word and **I**nside of a word. The segmentation network is shown in Figure 1. The models are trained on the respective training data sets by merging the word forms in each sentence into a sequence of characters. At inference, the segmentation model relies on sentence segmentation from UDPipe.

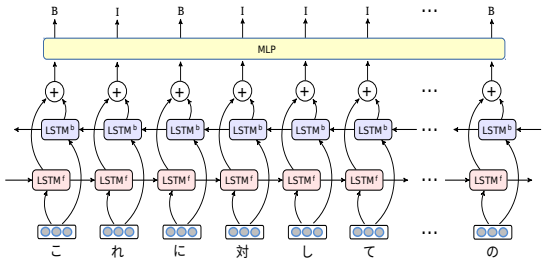

Figure 1: Word segmentation model based on character Bi-LSTM networks.

	ja_gsd			zh_gsd		
	Precision	**Recall**	**F1-score**	**Precision**	**Recall**	**F1-score**
B	97.43	97.26	97.35	96.50	96.48	96.49
I	96.35	96.58	96.46	93.94	93.96	93.95
avg	96.97	96.97	96.97	95.96	95.96	95.96

Table 1: Word segmentation results on Chinese and Japanese development sets. **B** and **I** mark the **B**egining and **I**nside of a word.

Other than word segmentation, we used Roman alphabet for Hindi. The goal is to normalize the spell variations in Hindi texts. We used an open source converter[1] that uses a deterministic mapping between Devanagari to Roman alphabet.

[1] https://github.com/ltrc/indic-wx-converter

2.2 Dependency Parsing

2.2.1 Parsing Algorithm

We employ an arc-standard transition system (Nivre, 2003) as our parsing algorithm. A typical transition-based parsing system uses the shift-reduce decoding algorithm to map a parse tree onto a sequence of transitions. Throughout the decoding process a stack and a queue data structures are maintained. The queue stores the sequence of raw input, while the stack stores the partially processed input which may be linked with the rest of the words in the queue. The parse tree is build by consuming the words in the queue from left to right by applying a set of transition actions. There are three kinds of transition actions that are performed in the parsing process: `Shift`, `Left-Arc`, `Right-Arc`. Additionally, we use a `Swap` action which reorders top node in the stack and the top node in the queue for parsing non-projective arcs (Nivre, 2009).

At training time, the transition actions are inferred from the gold parse trees and the mapping between the parser state and the transition action is learned using a simple LSTM-based neural networking architecture presented in Goldberg (2016). While training, we use the oracle presented in (Nivre et al., 2009) to restrict the number of `Swap` actions needed to parse non-projective arcs. Given that Bi-LSTMs capture global sentential context at any given time step, we use minimal set of features in our parsing model. At each parser state, we restrict our features to just two top nodes in the stack. Since `Swap` action distorts the linear order of word sequence, it renders the LSTM representations irrelevant in case of non-projective sentences. To capture this distortion, we also use the top most word in the queue as an additional feature.

2.3 Joint POS tagging and Parsing

Inspired by stack-propagation model of Zhang and Weiss (2016), we jointly model POS tagging and parsing using a stack of tagger and parser networks. The parameters of the tagger network are shared and act as a regularization on the parsing model. The overall model is trained by minimizing a joint negative log-likelihood loss for both tasks. Unlike Zhang and Weiss (2016), we compute the gradients of the log-loss function simultaneously for each training instance. While the parser network is updated given the parsing loss

only, the tagger network is updated with respect to both tagging and parsing losses. Both tagger and parser networks comprise of an input layer, a feature layer, and an output layer as shown in Figure 2. Following Zhang and Weiss (2016), we refer to this model as stack-prop.

Tagger network: The input layer of the tagger encodes each input word in a sentence by concatenating a pre-trained word embedding with its character embedding given by a character Bi-LSTM. In the feature layer, the concatenated word and character representations are passed through two stacked Bi-LSTMs to generate a sequence of hidden representations which encode the contextual information spread across the sentence. The first Bi-LSTM is shared with the parser network while the other is specific to the tagger. Finally, output layer uses the feed-forward neural network with a softmax function for a probability distribution over the Universal POS tags. We only use the forward and backward hidden representations of the focus word for classification.

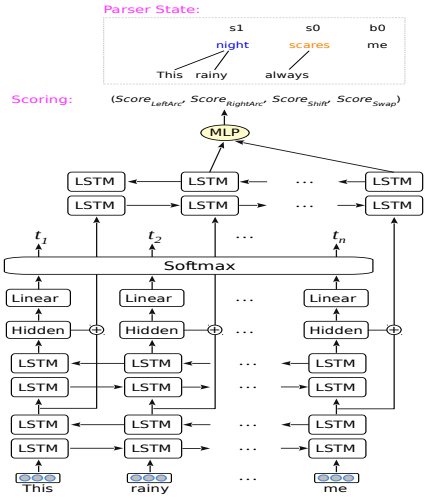

Figure 2: POS tagging and parsing network based on stack-propagation model proposed in (Zhang and Weiss, 2016).

Parser Network: Similar to the tagger network, the input layer encodes the input sentence using word and character embeddings which are then passed to the shared Bi-LSTM. The hidden representations from the shared Bi-LSTM are then concatenated with the dense representations from the feed-forward network of the tagger and passed through the Bi-LSTM specific to the parser. This ensures that the tagging network is penalized for the parsing error caused by error propagation by back-propagating the gradients to the shared tagger parameters (Zhang and Weiss, 2016). Finally, we use a non-linear feed-forward network to predict the labeled transitions for the parser configurations. From each parser configuration, we extract the top node in the stack and the first node in the buffer and use their hidden representations from the parser specific Bi-LSTM for classification.

2.4 Cross-domain Transfer

Among 57 languages, 17 languages presented in the task have multiple treebanks from different domains. From among the 17 languages, almost 5 languages have at-least one treebank which is smaller in size than the rest containing no more than 2000 sentences for training. To boost the performance of parsers trained on these smaller treebanks (target), we leverage large cross-domain treebanks (source) in the same language using neural stacking as a knowledge transfer mechanism.

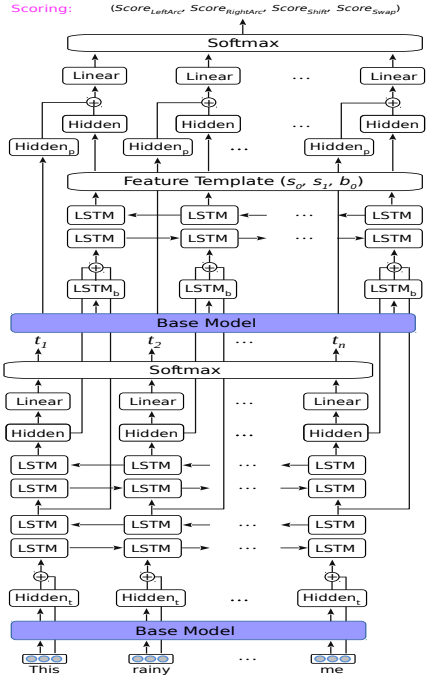

Figure 3: Knowledge transfer from resource-rich domains to resource-poor domains using neural stacking (Zhang and Weiss, 2016; Chen et al., 2016).

As we discussed above, we adapted feature-level neural stacking (Zhang and Weiss, 2016; Chen et al., 2016) for joint learning of POS tagging and parsing. Similarly, we also adapt this stacking approach for cross-domain knowledge trans-

fer by incorporating the syntactic knowledge from resource-rich domain into resource-poor domain. Recently, Wang et al. (2017); Bhat et al. (2018) showed significant improvements in parsing social media texts by injecting syntactic knowledge from large cross-domain treebanks using neural stacking.

As shown in Figure 3, we transfer both POS tagging and parsing information from the source model. For tagging, we augment the input layer of the target tagger with the hidden layer of multi-layered perceptron (MLP) of the source tagger. For transferring parsing knowledge, hidden representations from the parser specific Bi-LSTM of the source parser are augmented with the input layer of the target parser which already includes the hidden layer of the target tagger, word and character embeddings. In addition, we also add the MLP layer of the source parser to the MLP layer of the target parser. The MLP layers of the source parser are generated using raw features from target parser configurations. Apart from the addition of these learned representations from the source model, the overall target model remains similar to the base model shown in Figure 2. The tagging and parsing losses are back-propagated by traversing back the forward paths to all trainable parameters in the entire network for training and the whole network is used collectively for inference.

3 Experiments

We train three kinds of parsing models based on the availability of training data: stack-prop models trained for languages having large treebanks, ii) stacking models for languages having smaller in-domain treebanks and large out-domain treebanks, and iii) backoff character models for those languages which have neither in-domain nor out-domain training data. We will first discuss the details about the experimental setup for all these models and subsequently, we will discuss the results.

3.1 Hyperparameters

Word Representations For the stack-prop and stacking models, we include lexical features in the input layer of the neural networks using 64-dimension pre-trained word embeddings concatenated with 64-dimension character-based embeddings obtained using a Bi-LSTM over the characters of a word. For each language, we include pre-trained embeddings only for 100K most frequent words in the raw corpora.

The distributed word representations for each language are learned separately from their monolingual corpora collected from Web to Corpus (W2C) (Majliš, 2011)[2] and latest wiki dumps[3]. The word representations are learned using Skip-gram model with negative sampling which is implemented in `word2vec` toolkit (Mikolov et al., 2013). For our backoff character model we only use 64-dimension character Bi-LSTM embeddings in the input layer of the network.

Hidden dimensions The word-level Bi-LSTMs have 128 cells while the character-level Bi-LSTMs have 64 cells. The POS tagger specific MLP has 64 hidden nodes while the parser MLP has 128 hidden nodes. We use hyperbolic tangent as an activation function in all tasks.

Learning We use momentum SGD for learning with a minibatch size of 1. The initial learning rate is set to 0.1 with a momentum of 0.9. The LSTM weights are initialized with random orthonormal matrices as described in (Saxe et al., 2013). We set the dropout rate to 30% for all the hidden states in the network. All the models are trained for up to 100 epochs, with early stopping based on the development set.

All of our neural network models are implemented in DyNet (Neubig et al., 2017).

4 Results

In Table 4, we present the results of our parsing models on all the official test sets, while in Table 5, we report the average results across evaluation sets. In both tables, we also provide comparison of results on all the evaluation matrices with the UDPipe baseline models. For 74 out of 82 treebanks, we have obtained an average improvement of 5.8% in LAS over the UDPipe baseline models. Although, we ranked 12^{th} in the overall shared task, our rankings are particularly better for all those treebanks which were parsed using the stacking models or parsed after segmentation by our own segmentation models.

[2] As pointed out by one of the reviewers, W2C was not listed on the list of allowed resources. Using this data for training word embeddings might have a significant impact for resource-poor languages. Our results, therefore, might not be directly comparable with other participating teams and should be taken with a grain of salt!

[3] `https://dumps.wikimedia.org`

Our parsing system took around 1 hour 30 minutes to parse all the official test sets on TIRA virtual machine.

Impact of Word Segmentation To evaluate the impact of our segmentation models, we conducted two parsing experiments; one using the segmentation from the UDPipe models, and the other using the segmentation from our own segmentation models. We compared the performance of both segmentations on Japanese and Chinese development sets. The results are shown in Table 2. As shown in the Table, we achieved an average improvement of 3% in LAS over the UDPipe baseline. By using our segmentation models, we have achieved better ranking for these two languages than our average ranking in the official evaluation.

	UDPipe			Our model		
	UPOS	**ULAS**	**LAS**	**UPOS**	**ULAS**	**LAS**
ja_gsd	89.14	78.73	77.47	90.92	82.57	81.30
zh_gsd	84.30	65.36	61.96	86.51	68.51	64.89

Table 2: Impact of our word segmentation models on Chinese and Japanese development sets.

Impact of Domain Adaptation We also conducted experiments to evaluate the impact of neural stacking for knowledge transfer from resource-rich domains to resource-poor domains. In all the cases of neural stacking, we used the base models trained on those domains that have larger treebanks. We show the comparison of performance of stacking models with the base models trained on just the in-domain smaller treebanks. Results on development sets of multiple domains of English and French are shown in Table 3. For English domains, there is an improvement of 1% to 2% using `eng_ewt` as source domain for knowledge transfer, while for French improvements are quite high (2% to 5%) using `fr_gsd` as source domain. Similar to the impact of word segmentation, our ranking on treebanks that use neural stacking is better than our average.

	Base model			Stacking model		
	UPOS	**ULAS**	**LAS**	**UPOS**	**ULAS**	**LAS**
en_gum	96.29	86.96	83.53	96.68	88.51	85.47
en_lines	96.71	83.95	80.16	97.13	85.21	81.70
fr_sequoia	98.27	90.42	87.72	98.75	92.18	89.79
fr_spoken	95.91	82.40	74.96	97.25	86.01	79.82

Table 3: Impact of neural stacking on multiple domains of English and French.

5 Conclusion

In this paper, we have described our parsing models that we have submitted to CoNLL-2018 parsing shared task on Universal Dependencies. We have developed three types of models depending on the number of training samples. All of our models learn POS tag and parse tree information jointly using stack-propagation. For smaller treebanks, we have used neural stacking for knowledge transfer from large cross-domain treebanks. Moreover, we also developed our own segmentation models for Japanese and Chinese for improving the parsing results of these languages. We have significantly improved the baseline results from UDPipe for almost all the official test sets. Finally, we have achieved 12^{th} rank in the shared task for average LAS.

References

Irshad Bhat, Riyaz A. Bhat, Manish Shrivastava, and Dipti Sharma. 2018. Universal dependency parsing for hindi-english code-switching. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. Association for Computational Linguistics, pages 987–998. http://aclweb.org/anthology/N18-1090.

Hongshen Chen, Yue Zhang, and Qun Liu. 2016. Neural network for heterogeneous annotations. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Austin, Texas, pages 731–741.

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.

Martin Majliš. 2011. W2C web to corpus corpora. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University. http://hdl.handle.net/11858/00-097C-0000-0022-6133-9.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* .

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, et al. 2017. Dynet: The dynamic neural network toolkit. *arXiv preprint arXiv:1701.03980* .

	BASELINE UDPipe 1.2					SLT-Interactions				
	UPOS	**ULAS**	**LAS**	**MLAS**	**BLEX**	**UPOS**	**ULAS**	**LAS**	**MLAS**	**BLEX**
af_afribooms	95.12	82.02	77.88	64.48	66.60	97.42	86.05	83.07	68.66	72.89
ar_padt	89.34	71.44	66.41	55.01	57.60	90.34	75.38	71.52	58.00	63.20
bg_btb	97.72	88.88	84.91	75.30	73.78	98.54	91.84	88.64	78.63	78.00
br_keb	30.74	27.80	10.25	0.37	2.10	36.80	36.06	14.15	0.17	3.66
bxr_bdt	41.66	29.20	12.61	2.09	4.41	43.07	24.89	8.29	0.86	3.76
ca_ancora	98.00	88.66	85.61	76.74	77.27	98.66	91.77	89.42	80.79	82.21
cs_cac	98.32	87.11	83.72	70.89	77.65	99.00	91.37	89.14	74.27	83.37
cs_fictree	97.28	86.77	82.49	69.26	74.96	98.25	91.52	88.13	72.82	81.25
cs_pdt	98.21	86.88	83.94	74.32	79.39	98.70	90.74	88.44	77.07	84.14
cs_pud	96.57	85.29	80.08	66.53	73.79	96.85	88.63	83.76	68.39	77.33
cu_proiel	93.70	72.03	65.46	53.96	58.39	94.66	74.66	68.81	54.65	61.15
da_ddt	95.44	79.14	75.43	65.41	66.04	97.08	83.97	81.09	69.45	71.81
de_gsd	91.58	75.99	70.85	34.09	60.56	94.03	81.61	77.23	38.50	67.89
el_gdt	95.63	85.47	82.11	65.33	68.67	97.64	89.16	86.67	68.10	74.05
en_ewt	93.62	80.48	77.56	68.70	71.02	94.88	84.04	81.52	71.37	75.21
en_gum	93.24	78.48	74.20	62.66	62.14	95.99	85.47	82.42	67.95	70.41
en_lines	94.71	78.26	73.10	64.03	65.42	96.92	82.53	78.38	68.41	71.25
en_pud	94.15	83.05	79.56	67.59	71.14	95.66	87.36	84.64	71.41	76.89
es_ancora	98.14	87.42	84.43	76.01	76.43	98.74	90.90	88.68	80.77	81.93
et_edt	95.50	79.16	75.02	67.12	63.85	96.91	84.89	81.85	71.18	69.65
eu_bdt	92.34	75.00	70.13	57.65	63.50	95.72	83.08	79.57	63.57	72.15
fa_seraji	96.01	83.10	79.10	72.20	69.43	97.15	88.42	84.94	77.32	75.39
fi_ftb	92.28	79.86	75.64	65.22	61.76	95.02	86.90	83.64	70.66	68.89
fi_pud	95.84	83.33	80.15	73.16	65.46	86.74	65.44	54.12	48.74	50.14
fi_tdt	94.37	80.28	76.45	68.58	62.19	96.16	86.30	83.63	72.84	67.63
fo_oft	44.66	42.64	25.19	0.36	5.56	59.61	57.64	44.51	0.52	11.30
fr_gsd	95.75	84.17	81.05	72.16	74.22	96.12	87.56	84.74	75.31	78.44
fr_sequoia	95.84	83.85	81.12	71.34	74.41	97.83	89.66	88.11	77.90	82.62
fr_spoken	92.94	71.46	65.56	53.46	54.67	97.11	77.61	73.04	62.46	62.14
fro_srcmf	94.30	85.27	79.27	70.70	74.45	76.67	67.47	52.97	35.06	44.91
ga_idt	89.21	72.66	62.93	37.66	42.06	91.46	75.84	66.54	36.66	45.17
gl_ctg	96.26	79.15	76.10	62.11	65.29	96.92	82.93	80.46	67.90	70.93
gl_treegal	91.09	71.61	66.16	49.13	51.60	95.12	77.50	72.54	53.42	58.05
got_proiel	94.31	68.59	62.16	48.57	55.02	94.21	70.33	63.38	47.31	55.86
grc_perseus	82.37	64.40	57.75	31.05	38.74	90.45	71.11	65.00	39.96	43.70
grc_proiel	95.87	71.99	67.57	49.51	55.85	96.39	76.27	71.64	51.57	59.41
he_htb	80.87	62.18	57.86	44.09	46.51	82.60	65.68	62.13	47.57	51.89
hi_hdtb	95.75	91.41	87.15	69.09	79.93	97.48	94.22	91.08	72.79	63.66
hr_set	96.33	84.49	78.61	58.72	70.26	97.88	89.59	84.83	61.70	76.31
hsb_ufal	65.75	35.02	23.64	3.55	11.72	76.60	54.86	46.42	7.67	20.17
hu_szeged	90.59	72.55	66.76	52.82	56.92	95.14	79.93	74.80	58.08	64.77
hy_armtdp	65.40	36.81	21.79	6.84	11.94	43.41	33.17	11.61	1.13	5.36
id_gsd	92.99	80.63	74.37	63.42	62.50	93.79	83.98	77.66	65.76	65.99
it_isdt	97.05	88.91	86.26	77.06	77.12	97.93	91.22	89.14	79.79	81.00
it_postwita	93.94	72.34	66.81	53.64	53.99	95.43	77.21	72.54	57.89	59.96
ja_gsd	87.85	74.49	72.32	58.35	60.17	90.29	80.60	78.77	62.39	60.62
ja_modern	48.44	29.36	22.71	8.10	9.49	53.36	34.11	27.01	11.08	12.45
kk_ktb	48.94	39.45	24.21	7.62	9.79	40.98	21.92	7.10	0.95	2.65
kmr_mg	59.31	32.86	23.92	5.47	11.86	42.08	25.80	9.39	0.55	4.02
ko_gsd	93.44	69.21	61.40	54.10	50.50	95.56	85.21	81.82	76.51	68.66

Table 4: Accuracy of different parsing models on the evaluation set. POS tags are jointly predicted with parsing. LID = Language tag, TRN = Transliteration/normalization.

	BASELINE UDPipe 1.2					SLT-Interactions				
	UPOS	**ULAS**	**LAS**	**MLAS**	**BLEX**	**UPOS**	**ULAS**	**LAS**	**MLAS**	**BLEX**
All treebanks	87.32	71.64	65.80	52.42	55.80	88.12	75.46	69.98	54.52	59.68
Big treebanks	93.71	78.78	74.14	61.27	64.67	95.11	83.50	79.67	64.95	69.77
Parallel treebanks	85.23	71.22	66.63	51.75	54.87	85.07	70.96	64.73	48.47	54.90
Small treebanks	87.36	63.17	55.01	38.80	41.06	87.36	65.17	56.74	35.73	42.90
Low-resource treebanks	45.20	30.08	17.17	3.44	7.63	43.04	31.48	17.47	1.79	6.95

Table 5: POS tagging accuracies of different models on CS evaluation set. SP = stack-prop.

Joakim Nivre. 2003. An efficient algorithm for projective dependency parsing. In *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT)*. Citeseer.

Joakim Nivre. 2009. Non-projective dependency parsing in expected linear time. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*. Association for Computational Linguistics, pages 351–359.

Joakim Nivre, Marco Kuhlmann, and Johan Hall. 2009. An improved oracle for dependency parsing with online reordering. In *Proceedings of the 11th international conference on parsing technologies*. Association for Computational Linguistics, pages 73–76.

Andrew M Saxe, James L McClelland, and Surya Ganguli. 2013. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120* .

Milan Straka and Jana Straková. 2017. Tokenizing, pos tagging, lemmatizing and parsing ud 2.0 with udpipe. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*. Association for Computational Linguistics, Vancouver, Canada, pages 88–99. http://www.aclweb.org/anthology/K/K17/K17-3009.pdf.

Hongmin Wang, Yue Zhang, GuangYong Leonard Chan, Jie Yang, and Hai Leong Chieu. 2017. Universal dependencies parsing for colloquial singaporean english. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Vancouver, Canada, pages 1732–1744.

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*. Association for Computational Linguistics, Brussels, Belgium, pages 1–20.

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 (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 1557–1566.