Character Composition Model with Convolutional Neural Networks for Dependency Parsing on Morphologically Rich Languages

Xiang Yu and **Ngoc Thang Vu**
Institut für Maschinelle Sprachverarbeitung
Universität Stuttgart
{xiangyu,thangvu}@ims.uni-stuttgart.de

Abstract

We present a transition-based dependency parser that uses a convolutional neural network to compose word representations from characters. The character composition model shows great improvement over the word-lookup model, especially for parsing agglutinative languages. These improvements are even better than using pre-trained word embeddings from extra data. On the SPMRL data sets, our system outperforms the previous best greedy parser (Ballesteros et al., 2015) by a margin of 3% on average.[1]

1 Introduction

As with many other NLP tasks, dependency parsing also suffers from the out-of-vocabulary (OOV) problem, and probably more than others since training data with syntactical annotation is usually scarce. This problem is particularly severe when the target is a morphologically rich language. For example, in the SPMRL shared task data sets (Seddah et al., 2013, 2014), 4 out of 9 treebanks contain more than 40% word types in the development set that are never seen in the training set.

One way to tackle the OOV problem is to pre-train the word embeddings, *e.g.*, with word2vec (Mikolov et al., 2013), from a large set of unlabeled data. This comes with two main advantages: (1) more word types, which means that the vocabulary is extended by the unlabeled data, so that some of the OOV words now have a learned representation; (2) more word tokens per type, which means that the syntactic and semantic similarities of the words are better modeled than only using the parser training data.

[1]The parser is available at http://www.ims.uni-stuttgart.de/institut/mitarbeiter/xiangyu/index.en.html

Pre-trained word embeddings can alleviate the OOV problem by expanding the vocabulary, but it does not model the morphological information. Instead of looking up word embeddings, many researchers propose to compose the word representation from characters for various tasks, *e.g.*, part-of-speech tagging (dos Santos and Zadrozny, 2014; Plank et al., 2016), named entity recognition (dos Santos and Guimarães, 2015), language modeling (Ling et al., 2015), machine translation (Costa-jussà and Fonollosa, 2016). In particular, Ballesteros et al. (2015) use a bidirectional long short-term memory (LSTM) character model for dependency parsing. Kim et al. (2016) present a convolutional neural network (CNN) character model for language modeling, but make no comparison among the character models, and state that "it remains open as to which character composition model (*i.e.*, LSTM or CNN) performs better".

We propose to apply the CNN model by Kim et al. (2016) in a greedy transition-based dependency parser with feed-forward neural networks (Chen and Manning, 2014; Weiss et al., 2015). This model requires no extra unlabeled data but performs better than using pre-trained word embeddings. Furthermore, it can be combined with word embeddings from the lookup table since they capture different aspects of word similarities.

Experimental results show that the CNN model works especially well on agglutinative languages, where the OOV rates are high. On other morphologically rich languages, the CNN model also performs at least as good as the word-lookup model.

Furthermore, our CNN model outperforms both the original and our re-implementation of the bidirectional LSTM model by Ballesteros et al. (2015) by a large margin. It provides empirical evidence to the aforementioned open question, suggesting that the CNN is the better character composition model for dependency parsing.

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Short Papers), pages 672–678
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-2106

2 Parsing Models

2.1 Baseline Parsing Model

As the baseline parsing model, we re-implement the greedy parser in Weiss et al. (2015) with some modifications, which brings about 0.5% improvement, outlined below.[2]

Since most treebanks contain non-projective trees, we use an approximate non-projective transition system similar to Attardi (2006). It has two additional transitions (LEFT-2 and RIGHT-2) to the Arc-Standard system (Nivre, 2004) that attach the top of the stack to the third token on the stack, or *vice versa*. We also extend the feature templates in Weiss et al. (2015) by extracting the children of the third token in the stack. The complete transitions and feature templates are listed in Appendix A.

The feature templates consist of 24 tokens in the configuration, we look up the embeddings of the word forms, POS tags and dependency labels of each token.[3] We then concatenate the embeddings $\mathbf{E}_{word}(t_i)$, $\mathbf{E}_{tag}(t_i)$, $\mathbf{E}_{label}(t_i)$ for each token t_i, and use a dense layer with ReLU non-linearity to obtain a compact representation $\mathbf{f}(t_i)$ of the token:

$$\mathbf{x}(t_i) = [\mathbf{E}_{word}(t_i); \mathbf{E}_{tag}(t_i); \mathbf{E}_{label}(t_i)] \quad (1)$$
$$\mathbf{f}(t_i) = \max\{\mathbf{0}, \mathbf{W}_f \mathbf{x}(t_i) + \mathbf{b}_f\}$$

We concatenate the representations of the tokens and feed them through two hidden layers with ReLU non-linearity, and finally into the softmax layer to compute the probability of each transition:

$$\mathbf{h}_0 = [\mathbf{f}(t_1); \mathbf{f}(t_2); ...; \mathbf{f}(t_{24})]$$
$$\mathbf{h}_1 = \max\{\mathbf{0}, \mathbf{W}_1 \mathbf{h}_0 + \mathbf{b}_1\}$$
$$\mathbf{h}_2 = \max\{\mathbf{0}, \mathbf{W}_2 \mathbf{h}_1 + \mathbf{b}_2\}$$
$$p(\cdot|t_1, ..., t_{24}) = \mathrm{softmax}(\mathbf{W}_3 \mathbf{h}_2 + \mathbf{b}_3)$$

$\mathbf{E}_{word}$, $\mathbf{E}_{tag}$, $\mathbf{E}_{label}$, $\mathbf{W}_f$, $\mathbf{W}_1$, $\mathbf{W}_2$, $\mathbf{W}_3$, $\mathbf{b}_f$, $\mathbf{b}_1$, $\mathbf{b}_2$, $\mathbf{b}_3$ are all the parameters that will be learned through back propagation with averaged stochastic gradient descent in mini-batches.

Note that Weiss et al. (2015) directly concatenate the embeddings of the words, tags, and labels of all the tokens together as input to the hidden layer. Instead, we first group the embeddings of the word, tag, and label of each token and compute an intermediate representation with shared parameters, then concatenate all the representations as input to the hidden layer.

2.2 LSTM Character Composition Model

To tackle the OOV problem, we want to replace the word-lookup table with a function that composes the word representation from characters.

As a baseline character model, we re-implement the bidirectional LSTM character composition model following Ballesteros et al. (2015). We replace the lookup table $\mathbf{E}_{word}$ in the baseline parser with the final outputs of the forward and backward LSTMs $\overleftarrow{\mathbf{lstm}}$ and $\overrightarrow{\mathbf{lstm}}$. Equation (1) is then replaced with

$$\mathbf{x}(t_i) = [\overleftarrow{\mathbf{lstm}}(t_i); \overrightarrow{\mathbf{lstm}}(t_i); \mathbf{E}_{tag}(t_i); \mathbf{E}_{label}(t_i)].$$

We refer the readers to Ling et al. (2015) for the details of the bidirectional LSTM.

2.3 CNN Character Composition Model

In contrast to the LSTM model, we propose to use a "flat" CNN as the character composition model, similar to Kim et al. (2016).[4]

Equation (1) is thus replaced with

$$\mathbf{x}(t_i) = [\mathbf{cnn}^{l_1}(t_i); \mathbf{cnn}^{l_2}(t_i); ...; \mathbf{cnn}^{l_k}(t_i); \mathbf{E}_{tag}(t_i); \mathbf{E}_{label}(t_i)]. \quad (2)$$

Concretely, the input of the model is a concatenated matrix of character embeddings $\mathbf{C} \in \mathbb{R}^{d_i \times w}$, where d_i is the dimensionality of character embeddings (number of input channels) and w is the length of the padded word.[5] We apply k convolutional kernels $\mathcal{K} \in \mathbb{R}^{d_o \times d_i \times l_k}$ with ReLU non-linearity on the input, where d_o is the number of output channels and l_k is the length of the kernel. The output of the convolution operation is $\mathbf{O}_{conv} \in \mathbb{R}^{d_o \times (l-k+1)}$, and we apply a max-over-time pooling that takes the maximum activations of the kernel along each channel, obtaining the final output $\mathbf{O}_{final} \in \mathbb{R}^{d_o}$, which corresponds to the most salient n-gram representation of the word, denoted $\mathbf{cnn}^{l_k}$ in Equation (2). We then concatenate the outputs of several such CNNs with different lengths, so that the information from different n-grams are extracted and can interact with each other.

[2] We only experiment with the greedy parser, since this paper focuses on the character-level input features and is independent of the global training and inference as in Weiss et al. (2015); Andor et al. (2016).

[3] The tokens in the stack and buffer do not have labels yet, we use a special label <NOLABEL> instead.

[4] We do not use the highway networks since it did not improve the performance in preliminary experiments.

[5] The details of the padding is in Appendix B.

	Model	Ara	Baq	Fre	Ger	Heb	Hun	Kor	Pol	Swe	Avg
Int	WORD	84.50	77.87	82.20	85.35	**74.68**	76.17	84.62	80.71	79.14	80.58
	W2V	**85.11**	79.07	**82.73**	**86.60**	74.55	78.21	85.30	82.37	79.67	81.51
	LSTM	83.42	82.97	81.35	85.34	74.03	83.06	86.56	80.13	77.44	81.48
	CNN	84.65	**83.91**	82.41	85.61	74.23	**83.68**	**86.99**	**83.28**	**80.00**	**82.75**
	LSTM+WORD	**84.75**	83.43	**82.25**	85.56	74.62	83.43	86.85	82.30	79.85	82.56
	CNN+WORD	84.58	**84.22**	81.79	**85.85**	**74.79**	**83.51**	**87.21**	**83.66**	**80.52**	**82.90**
	LSTM+W2V	85.35	83.94	83.04	86.38	**75.15**	83.30	87.35	83.00	79.38	82.99
	CNN+W2V	**85.67**	**84.37**	**83.09**	**86.81**	74.95	**84.08**	**87.72**	**84.44**	**80.35**	**83.50**
Ext	B15-WORD	**83.46**	73.56	**82.03**	**84.62**	**72.70**	69.31	83.37	**79.83**	**76.40**	78.36
	B15-LSTM	83.40	**78.61**	81.08	84.49	72.26	**76.34**	**86.21**	78.24	74.47	**79.46**
	BestPub	86.21	85.70	85.66	89.65	81.65	86.13	87.27	87.07	82.75	85.79

Table 1: LAS on the test sets, the best LAS in each group is marked in bold face.

3 Experiments

3.1 Experimental Setup

We conduct our experiments on the treebanks from the SPMRL 2014 shared task (Seddah et al., 2013, 2014), which includes 9 morphologically rich languages: Arabic, Basque, French, German, Hebrew, Hungarian, Korean, Polish, and Swedish. All the treebanks are split into training, development, and test sets by the shared task organizers. We use the fine-grained predicted POS tags provided by the organizers, and evaluate the labeled attachment scores (LAS) including punctuation.

We experiment with the CNN-based character composition model (CNN) along with several baselines. The first baseline (WORD) uses the word-lookup model described in Section 2.1 with randomly initialized word embeddings. The second baseline (W2V) uses pre-trained word embeddings by word2vec (Mikolov et al., 2013) with the CBOW model and default parameters on the unlabeled texts from the shared task organizers. The third baseline (LSTM) uses a bidirectional LSTM as the character composition model following Ballesteros et al. (2015). Appendix C lists the hyper-parameters of all the models.

Further analysis suggests that combining the character composition models with word-lookup models could be beneficial since they capture different aspects of word similarities (orthographic vs. syntactic/semantic). We therefore experiment with four combined models in two groups: (1) randomly initialized word embeddings (LSTM+WORD vs. CNN+WORD), and (2) pre-trained word embeddings (LSTM+W2V vs. CNN+W2V).

The experimental results are shown in Table 1, with Int denoting internal comparisons (with three groups) and Ext denoting external comparisons, the highest LAS in each group is marked in bold face.

3.2 Internal Comparisons

In the first group, we compare the LAS of the four single models WORD, W2V, LSTM, and CNN. In macro average of all languages, the CNN model performs 2.17% higher than the WORD model, and 1.24% higher than the W2V model. The LSTM model, however, performs only 0.9% higher than the WORD model and 1.27% lower than the CNN model.

The CNN model shows large improvement in four languages: three agglutinative languages (Basque, Hungarian, Korean), and one highly inflected fusional language (Polish). They all have high OOV rate, thus difficult for the baseline parser that does not model morphological information. Also, morphemes in agglutinative languages tend to have unique, unambiguous meanings, thus easier for the convolutional kernels to capture.

In the second group, we observe that the additional word-lookup model does not significantly improve the CNN moodel (from 82.75% in CNN to 82.90% in CNN+WORD on average) while the LSTM model is improved by a much larger margin (from 81.48% in LSTM to 82.56% in LSTM+WORD on average). This suggests that the CNN model has already learned the most important information from the the word forms, while the LSTM model has not. Also, the combined CNN+WORD model is still better than the LSTM+WORD model, despite the large improvement in the latter.

In the third group where pre-trained word embeddings are used, combining CNN with W2V brings an extra 0.75% LAS over the CNN model. Combining LSTM with W2V shows similar trend but with much larger improvement, yet again, CNN+W2V outperforms LSTM+W2V both on average and individually in 8 out of 9 languages.

Model	Case	Ara	Baq	Fre	Ger	Heb	Hun	Kor	Pol	Swe	Avg
CNN	ΔIV	0.12	2.72	-0.44	0.13	-0.35	1.48	1.30	0.98	1.39	0.81
	ΔOOV	0.03	5.78	0.33	0.10	-1.04	5.04	2.17	2.34	0.95	1.74
LSTM	ΔIV	-0.58	1.98	-0.55	-0.08	-1.23	1.62	1.12	-0.49	0.21	0.22
	ΔOOV	-0.32	5.09	0.12	-0.21	-1.99	4.74	1.51	0.10	0.38	1.05

Table 2: LAS improvements by CNN and LSTM in the IV and OOV cases on the development sets.

Mod	Ara	Baq	Fre	Ger	Heb	Hun	Kor	Pol	Swe	Avg
♣bc	-1.23	-1.94	-1.35	-1.57	-0.79	-3.23	-1.22	-2.53	-1.54	-1.71
a♣c	-3.47	-3.96	-2.39	-2.54	-1.24	-4.52	-3.21	-4.47	-4.19	-3.33
ab♣	-1.52	-15.31	-0.72	-1.23	-0.26	-13.97	-10.22	-3.52	-2.61	-5.48
a♣♣	-3.73	-19.29	-3.30	-3.49	-1.21	-17.89	-12.95	-6.22	-6.01	-8.23
♣b♣	-3.02	-18.06	-2.60	-3.54	-1.42	-18.43	-11.69	-6.22	-3.85	-7.65
♣♣c	-5.11	-7.90	-4.05	-4.86	-2.50	-9.75	-4.56	-6.71	-6.74	-5.80

Table 3: Degradation of LAS of the CNN model on the masked development sets.

3.3 External Comparisons

We also report the results of the two models from Ballesteros et al. (2015): B15-WORD with randomly initialized word embeddings and B15-LSTM as their proposed model. Finally, we report the best published results (BestPub) on this data set (Björkelund et al., 2013, 2014).

On average, the B15-LSTM model improves their own baseline by 1.1%, similar to the 0.9% improvement of our LSTM model, which is much smaller than the 2.17% improvement of the CNN model. Furthermore, the CNN model is improved from a strong baseline: our WORD model performs already 2.22% higher than the B15-WORD model.

Comparing the individual performances on each language, we observe that the CNN model almost always outperforms the WORD model except for Hebrew. However, both LSTM and B15-LSTM perform higher than baseline only on the three agglutinative languages (Basque, Hungarian, and Korean), and lower than baseline on the other six.

Ballesteros et al. (2015) do not compare the effect of adding a word-lookup model to the LSTM model as in our second group of internal comparisons. However, Plank et al. (2016) show that combining the same LSTM character composition model with word-lookup model improves the performance of POS tagging by a very large margin. This partially confirms our hypothesis that the LSTM model does not learn sufficient information from the word forms.

Considering both internal and external comparisons in both average and individual performances, we argue that CNN is more suitable than LSTM as character composition model for parsing.

While comparing to the best published results (Björkelund et al., 2013, 2014), we have to note that their approach uses explicit morphological features, ensemble, ranking, *etc.*, which all can boost parsing performance. We only use a greedy parser with much fewer features, but bridge the 6 points gap between the previous best greedy parser and the best published result by more than one half.

3.4 Discussion on CNN and LSTM

We conjecture that the main reason for the better performance of CNN over LSTM is its flexibility in processing sub-word information. The CNN model uses different kernels to capture n-grams of different lengths. In our setting, a kernel with a minimum length of 3 can capture short morphemes; and with a maximum length of 9, it can practically capture a normal word. With the flexibility of capturing patterns from morphemes up to words, the CNN model almost always outperforms the word-lookup model.

In theory, LSTM has the ability to model much longer sequences, however, it is composed step by step with recurrence. For such deep network architectures, more data would be required to learn the same sequence, in comparison to CNN which can directly use a large kernel to match the pattern. For dependency parsing, training data is usually scarce, this could be the reason that the LSTM has not utilized its full potential.

3.5 Analyses on OOV and Morphology

The motivation for using character composition models is based on the hypothesis that it can address the OOV problem. To verify the hypothesis, we analyze the LAS improvements of the CNN and LSTM model on the development sets in two cases: (1) both the head and the dependent are in vocabu-

lary or (2) at least one of them is out of vocabulary. Table 2 shows the results, where the two cases are denoted as ΔIV and ΔOOV. The general trend in the results is that the improvements of both models in the OOV case are larger than in the IV case, which means that the character composition models indeed alleviates the OOV problem. Also, CNN improves on seven languages in the IV case and eight languages in the OOV case, and it performs consistently better than LSTM in both cases.

To analyze the informativeness of the morphemes at different positions, we conduct an ablation experiment. We split each word equally into three thirds, approximating the prefix, stem, and suffix. Based on that, we construct six modified versions of the development sets, in which we mask one or two third(s) of the characters in each word. Then we parse them with the CNN models trained on normal data. Table 3 shows the degradations of LAS on the six modified data sets compared to parsing the original data, where the position of ♣ signifies the location of the masks. The three agglutinative languages Basque, Hungarian, and Korean suffer the most with masked words. In particular, the suffixes are the most informative for parsing in these three languages, since they cause the most loss while masked, and the least loss while unmasked. The pattern is quite different on the other languages, in which the distinction of informativeness among the three parts is much smaller.

4 Conclusion

In this paper, we propose to use a CNN to compose word representations from characters for dependency parsing. Experiments show that the CNN model consistently improves the parsing accuracy, especially for agglutinative languages. In an external comparison on the SPMRL data sets, our system outperforms the previous best greedy parser.

We also provide empirical evidence and analysis, showing that the CNN model indeed alleviates the OOV problem and that it is better suited than the LSTM in dependency parsing.

Acknowledgements

This work was supported by the German Research Foundation (DFG) in project D8 of SFB 732. We also thank our collegues in the IMS, especially Anders Björkelund, for valuable discussions, and the anonymous reviewers for the suggestions.

References

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally normalized transition-based neural networks. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 2442–2452. https://doi.org/10.18653/v1/P16-1231.

Giuseppe Attardi. 2006. Experiments with a multilanguage non-projective dependency parser. In *Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL-X)*. Association for Computational Linguistics, pages 166–170. http://aclweb.org/anthology/W06-2922.

Miguel Ballesteros, Chris Dyer, and A. Noah Smith. 2015. Improved transition-based parsing by modeling characters instead of words with lstms. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 349–359. https://doi.org/10.18653/v1/D15-1041.

Anders Björkelund, Özlem Çetinoğlu, Agnieszka Faleńska, Richárd Farkas, Thomas Müller, Wolfgang Seeker, and Zsolt Szántó. 2014. The ims-wrocław-szeged-cis entry at the spmrl 2014 shared task: Reranking and morphosyntax meet unlabeled data. *Notes of the SPMRL* .

Anders Björkelund, Özlem Çetinoğlu, Richárd Farkas, Thomas Mueller, and Wolfgang Seeker. 2013. (re)ranking meets morphosyntax: State-of-the-art results from the spmrl 2013 shared task. In *Proceedings of the Fourth Workshop on Statistical Parsing of Morphologically-Rich Languages*. Association for Computational Linguistics, pages 135–145. http://aclweb.org/anthology/W13-4916.

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)*. Association for Computational Linguistics, pages 740–750. https://doi.org/10.3115/v1/D14-1082.

R. Marta Costa-jussà and R. José A. Fonollosa. 2016. Character-based neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, pages 357–361. https://doi.org/10.18653/v1/P16-2058.

Cicero dos Santos and Victor Guimarães. 2015. Boosting named entity recognition with neural character embeddings. In *Proceedings of the Fifth Named Entity Workshop*. Association for Computational Linguistics, pages 25–33. https://doi.org/10.18653/v1/W15-3904.

Cicero dos Santos and Bianca Zadrozny. 2014. Learning character-level representations for part-of-speech tagging. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*. pages 1818–1826.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*. pages 1026–1034.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. 2016. Character-aware neural language models. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA.*. AAAI Press, pages 2741–2749.

Wang Ling, Chris Dyer, W. Alan Black, Isabel Trancoso, Ramon Fermandez, Silvio Amir, Luis Marujo, and Tiago Luis. 2015. Finding function in form: Compositional character models for open vocabulary word representation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 1520–1530. https://doi.org/10.18653/v1/D15-1176.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, Curran Associates, Inc., pages 3111–3119.

Joakim Nivre. 2004. Incrementality in deterministic dependency parsing. In *Proceedings of the Workshop on Incremental Parsing: Bringing Engineering and Cognition Together*. http://aclweb.org/anthology/W04-0308.

Barbara Plank, Anders Søgaard, and Yoav Goldberg. 2016. Multilingual part-of-speech tagging with bidirectional long short-term memory models and auxiliary loss. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, pages 412–418. https://doi.org/10.18653/v1/P16-2067.

Djamé Seddah, Sandra Kübler, and Reut Tsarfaty. 2014. Introducing the spmrl 2014 shared task on parsing morphologically-rich languages. In *Proceedings of the First Joint Workshop on Statistical Parsing of Morphologically Rich Languages and Syntactic Analysis of Non-Canonical Languages*. Dublin City University, pages 103–109. http://aclweb.org/anthology/W14-6111.

Djamé Seddah, Reut Tsarfaty, Sandra Kübler, Marie Candito, D. Jinho Choi, Richárd Farkas, Jennifer Foster, Iakes Goenaga, Koldo Gojenola Galletebeitia, Yoav Goldberg, Spence Green, Nizar Habash, Marco Kuhlmann, Wolfgang Maier, Joakim Nivre, Adam Przepiórkowski, Ryan Roth, Wolfgang Seeker, Yannick Versley, Veronika Vincze, Marcin Woliński, Alina Wróblewska, and Villemonte Eric de la Clergerie. 2013. Overview of the spmrl 2013 shared task: A cross-framework evaluation of parsing morphologically rich languages. In *Proceedings of the Fourth Workshop on Statistical Parsing of Morphologically-Rich Languages*. Association for Computational Linguistics, pages 146–182. http://aclweb.org/anthology/W13-4917.

David Weiss, Chris Alberti, Michael Collins, and Slav Petrov. 2015. Structured training for neural network transition-based parsing. 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)*. Association for Computational Linguistics, pages 323–333. https://doi.org/10.3115/v1/P15-1032.

A Transitions and Feature Templates

Transition	Configurations (Before ⇒ After)
SHIFT	$(\sigma, [w_i\|\beta], A) \Rightarrow ([\sigma\|w_i], \beta, A)$
LEFT	$([\sigma\|w_i, w_j], \beta, A)$
	$\Rightarrow ([\sigma\|w_j], \beta, A \cup (w_j \rightarrow w_i))$
RIGHT	$([\sigma\|w_i, w_j], \beta, A)$
	$\Rightarrow ([\sigma\|w_i], \beta, A \cup (w_i \rightarrow w_j))$
LEFT-2	$([\sigma\|w_i, w_j, w_k], \beta, A)$
	$\Rightarrow ([\sigma\|w_j, w_k], \beta, A \cup (w_k \rightarrow w_i))$
RIGHT-2	$([\sigma\|w_i, w_j, w_k], \beta, A)$
	$\Rightarrow ([\sigma\|w_i, w_j], \beta, A \cup (w_i \rightarrow w_k))$

Table 4: The transition system in our experiments, where the configuration is a tuple of (stack, buffer, arcs).

s_1, s_2, s_3, s_4, b_1, b_2, b_3, b_4,
$s_1.lc_1$, $s_1.lc_2$, $s_1.rc_1$, $s_1.rc_2$,
$s_2.lc_1$, $s_2.lc_2$, $s_2.rc_1$, $s_2.rc_2$,
$s_3.lc_1$, $s_3.lc_2$, $s_3.rc_1$, $s_3.rc_2$,
$s_1.lc_1.lc_1$, $s_1.lc_1.rc_1$, $s_1.rc_1.lc_1$, $s_1.rc_1.rc_1$

Table 5: The list of tokens to extract feature templates, where s_i denotes the i-th token in the stack, b_i the i-th token in the buffer, lc_i denotes the i-th leftmost child, rc_i the i-th rightmost child.

B Character Input Preprocessing

For the CNN input, we use a list of characters with fixed length to for batch processing. We add some special symbols apart from the normal alphabets, digits, and punctuations: <SOW> as the start of the word, <EOW> as the end of the word, <MUL> as multiple characters in the middle of the word squeezed into one symbol, <PAD> as padding equally on both sides, and <UNK> as characters unseen in the training data.

For example, if we limit the input length to 9, a short word *ein* will be converted into <PAD>-<PAD>-<SOW>-e-i-n-<EOW>-<PAD>-<PAD>; a long word *prächtiger* will be <SOW>-p-r-ä-<MUL>-g-e-r-<EOW>. In practice, we set the length as 32, which is long enough for almost all the words.

C Hyper-Parameters

The common hyper-parameters of all the models are tuned on the development set in favor of the WORD model:

- 100,000 training steps with random sampling of mini-batches of size 100;
- test on the development set every 2,000 steps;
- early stop if the LAS on the development does not improve for 3 times in a row;
- learning rate of 0.1, with exponential decay rate of 0.95 for every 2,000 steps;
- L2-regularization rate of 10^{-4};
- averaged SGD with momentum of 0.9;
- parameters are initialized following He et al. (2015);
- dimensionality of the embeddings of each word, tag, and label are 256, 32, 32, respectively;
- dimensionality of the hidden layers are 512, 256;
- dropout on both hidden layers with rate of 0.1;
- total norm constraint of the gradients is 10.

The hyper-parameters for the CNN model are:

- dimensionality of the character embedding is 32;
- 4 convolutional kernels of lengths 3, 5, 7, 9;
- number of output channels of each kernel is 64;
- fixed length for the character input is 32.

The hyper-parameters for the LSTM model are:

- 128 hidden units for both LSTMs;
- all the gates use orthogonal initialization;
- gradient clipping of 10;
- no L2-regularization on the parameters.