Improved Graph-based Dependency Parsing via Hierarchical LSTM Networks

Wenhui Wang and Baobao Chang

Key Laboratory of Computational Linguistics, Ministry of Education.
School of Electronics Engineering and Computer Science, Peking University,
No.5 Yiheyuan Road, Haidian District, Beijing, 100871, China
Collaborative Innovation Center for Language Ability, Xuzhou, 221009, China.
{wangwenhui,chbb}@pku.edu.cn

Abstract. In this paper, we propose a neural graph-based dependency parsing model which utilizes hierarchical LSTM networks on character level and word level to learn word representations, allowing our model to avoid the problem of limited-vocabulary and capture both distributional and compositional semantic information. Our model achieves state-of-the-art accuracy on Chinese Penn Treebank and competitive accuracy on English Penn Treebank with only first-order features. Moreover, our model shows effectiveness in recovering dependencies involving out-of-vocabulary words.

Keywords: Graph-based dependency parsing, Hierarchical LSTM

1 Introduction

Dependency parsing is a fundamental task for language processing which has been investigated for decades. Among a variety of dependency parsing models, graph-based models are attractive for their ability of scoring the parsing decisions on a whole-tree basis. Recently, neural network models have been successfully introduced into graph-based dependency parsing and obtained state-of-the-art results. [16] presented a simple feed-forward network model which uses only atomic features such as word unigrams and POS tag unigrams. [18] proposed utilizing word representations learned by Bidirectional LSTM network to support parsing decisions and further improved their model by segment embeddings.

Effective as these models are, above models have a strong limitation in vocabulary due to its standard lookup-based word representations, which lead to difficulty in recovering dependencies involving out-of-vocabulary words. On the other hand, this standard lookup-based word representations capture only distributional semantics of words, which in essence encodes useful information in the surrounding contexts of the concerned word. However, for a more complete word representation, compositional semantics of a word would be necessary as well, which can be derived by combining meaning of word parts. This is especially important for languages like Chinese. The characters making up words bear

meanings of their own and usually determine the meanings of words to a certain extent.

In this paper, we propose to utilize a hierarchical LSTM-based network model to graph-based dependency parsing. Our model includes two levels of Bidirectional LSTM network: one at the character level and one at the word level. The character-level LSTM aims to capture compositional semantics of word, which is then combined with the standard lookup-based word embedding to produce a more complete representation of word. The introduction of compositional representation also makes our model more robust to OOV words which are usually not well represented by the lookup-based word embeddings. The word-level LSTM aims to enrich the word representation and capture potential long range contextual information to support parsing decisions.

Character-level information has already been explored in transition-based dependency parsing. [5] proposed replacing lookup-based word representations with character-based representations which obtained by Bidirectional LSTM. They show that character-level information is helpful for morphologically rich language. However, simply using character-based representations results in their model performing poorly for English and Chinese. Different from their work, lookup-based word representations and character-based representations are combined and a word-level Bidirectional LSTM is further utilized to capture richer contextual information in our work. The combined word representation used in our work captures both distributional and compositional semantic information.

We evaluate our model on the English Penn Treebank and Chinese Penn Treebank, our model achieves state-of-the-art accuracy on Chinese Penn Treebank and competitive accuracy on English Penn Treebank.

2 Neural Network Model

In this section, we describe the architecture and the training of our neural network model in detail.

2.1 Word Representation

Given an input sentence $s = w_1, \ldots, w_n$ together with the corresponding POS tags $p_1, \ldots, p_n$, a hierarchical LSTM network is utilized to learn word representations, which is summarized in figure 1. A character-level Bidirectional LSTM is used to compute character-based embeddings of words. We then concatenate four vectors: the forward character-level LSTM hidden vector ($\overrightarrow{c}$); the backward character-level LSTM hidden vector ($\overleftarrow{c}$); the word embedding ($e(w_i)$) and the POS tag embedding ($e(p_i)$). A linear transformation w_e is performed and passed though an element-wise activation function g (ReLU is used as our activation function):

$$x_i = g(w_e[\overrightarrow{c}; \overleftarrow{c}; e(w_i); e(p_i)] + b_e) \tag{1}$$

A word-level Bidirectional LSTM is utilized to enrich word vector representations. Each output vector (v_i) of word-level Bidirectional LSTM is used to

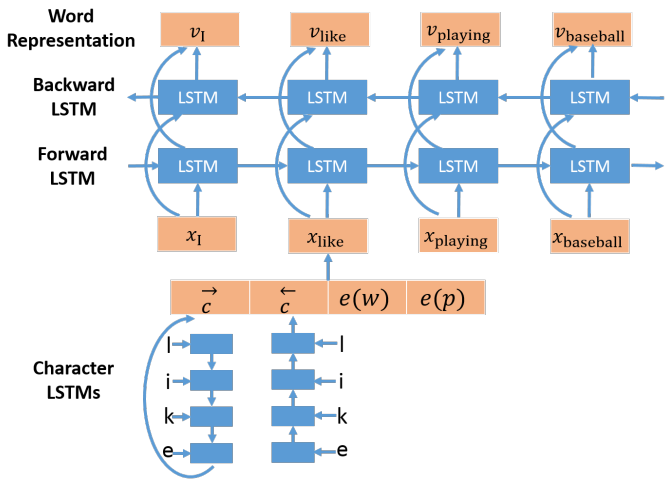

Fig. 1. Illustration for learning word representations based on hierarchical LSTM networks. $e(w)$ and $e(p)$ stand for the word embedding and POS tag embedding for this word.

represent words in sentence. Hierarchical LSTM networks allow each word representation v_i to capture information regarding the character level, the word form and POS tag, as well as the sentential context it appears in. In addition, combining character-level information allows our model to avoid the problem of out-of-vocabulary words.

2.2 Score Model

A neural network model is utilized to score dependency arcs. We use the same architecture and the same features as [18].

For a dependency pair (h, m), [18] utilize feature embeddings including the word representations for head word h and the modifier word m, distance between them (distance features are encoded as randomly initialized embeddings), and segment embeddings for the dependency pair (h, m). A sentence is divided into three parts (*prefix*, *infix* and *suffix*) by head word h and modifier word m, these parts are called segments. An extra forward LSTM layer is placed on top of the word representations and segment embeddings are learned by using subtraction between the forward LSTM hidden vectors.

All feature embeddings are mapped to the hidden layer. Direction-specific transformation is utilized to model edge direction:

$$h = g\Big(\sum_i W^d_{h_i} a_i + b^d_h\Big) \tag{2}$$

where a_i is the feature embedding, $W^d_{h_i}$ and b^d_h are bound with index $d \in \{0, 1\}$ which indicates the direction between head and modifier.

A output layer is finally added on the top of the hidden layer for scoring dependency arcs:

$$Score(h, m) = W_o^d h + b_o^d \tag{3}$$

Where $Score(h, m) \in \mathbb{R}^L$ is the output vector, L is the number of dependency types. Each dimension of the output vector is the score for each kind of dependency type of head-modifier pair.

2.3 Neural Training

We use the Max-Margin criterion to train our model. Parameter optimization is performed with the diagonal variant of AdaGrad [9] with minibatchs (batch size = 20). To mitigate overfitting, we apply dropout [12] on the hidden layer of the score model with 0.2 rate.

The following hyper-parameters are used in all experiments: word embedding size = 100, POS tag embedding size = 50, character embedding size = 50, hidden layer size = 200, character-level LSTM hidden vector size = 50, word-level LSTM hidden vector size = 100, character-level LSTM layers = 1, word-level LSTM layers = 2, regularization parameter $\lambda = 10^{-4}$.

We initialized the parameters using pretrained word embeddings[1]. Following [10] and [18], we use a variant of the skip n-gram model introduced by [13] on Gigaword corpus [11]. We also experimented with randomly initialized embeddings, where embeddings are uniformly sampled from range $[-0.3, 0.3]$. All other parameters are uniformly sampled from range $[-0.05, 0.05]$.

3 Experiments

In this section, we present our experimental setup and the main results of our work.

3.1 Experiments Setup

We conduct our experiments on the English Penn Treebank (PTB) [1] and the Chinese Penn Treebank (CTB) [2] datasets.

For English, we evaluated on the standard Wall Street Journal (WSJ) part of the Penn Treebank. Dependencies generated from version 3.3.0[2] of the Stanford converter [15], we call it Penn-SD In the following section. We followed standard practice and used sections 2-21 for training, section 22 for development, and section 23 for testing. The Stanford POS Tagger [17] with ten-way jackknifing of the training data is used for assigning POS tags (accuracy $\approx$ 97.2%).

For Chinese, we adopt the same split of CTB5 as described in [21]. Dependencies are converted using the Penn2Malt[3] tool with the head-finding rules of [21]. And following [21, 23], we use gold segmentation and POS tags for the input.

[1] In our experiments, all words occurring less than 10 times in the corpus are treated as unknown words.

[2] `http://nlp.stanford.edu/software/lex-parser.shtml`

[3] `http://stp.lingfil.uu.se/nivre/research/Penn2Malt.html`

3.2 Experiments Results

Method	CTB5	
	UAS	LAS
Zhang and Nivre (2011)[23]	86.0	84.4
Bernd Bohnet (2012)[6]	87.5	85.9
Zhang and McDonald (2014)[22]	**87.96**	86.34
Dyer et al. (2015)[10]	87.2	85.7
Ballesteros et al. (2015b)[5]	85.30	83.72
Wang and Chang (2016)[18]	87.55	86.23
Our model	87.77	**86.42**

Table 1. Comparison with previous state-of-the-art models on CTB5.

We first compare our model with previous state-of-the-art models on Chinese. Following previous work, UAS (unlabeled attachment scores) and LAS (labeled attachment scores) are calculated by excluding punctuation[4]. Table 1 lists the performances of our model as well as previous state-of-the-art models on CTB5. As we can see, word representations combining character-level information do improve model's performance compared with [18]. Moreover, the LAS of our model achieves state-of-the-art accuracy.

Method	Penn-SD	
	UAS	LAS
Zhang and McDonald (2014)[22]	93.01	90.64
Dyer et al. (2015)[10]	93.1	90.9
Weiss et al. (2015)[19]	93.99	92.05
Ballesteros et al. (2015b)[5]	91.63	89.44
Andor et al. (2016)[3]	**94.41**	**92.55**
Wang and Chang (2016)[18]	94.08	91.82
Our model	94.13	91.85

Table 2. Comparison with previous state-of-the-art models on Penn-SD.

We then compare our model with previous state-of-the-art models on English. Table 2 lists the performances of our model as well as previous state-of-the-art models on Penn-SD. As we can see, the improvement on Penn-SD is lower than CTB5. On one hand, the OOV rate of Penn-SD (2.2%) is much lower than CTB5 (10.4%). On the other hand, character-level information of English reflects more morphological information which is also encoded in treebank POS tags, while character-level information of Chinese supplements semantic information within words to support parsing decisions. Moreover, our model outperforms [5] by a

[4] Following previous work, a token is a punctuation if its POS tag is {“ ” : , .}

substantial margin on both Chinese and English since our word representations capture richer information rather than simple character-level information.

	HiLSTM	BiLSTM	Significant Test
UAS	84.42	80.46	$t = 1.96$, $p < 0.05$
LAS	81.72	75.84	$t = 3.91$, $p < 10^{-4}$

Table 3. Accuracy and significant test in recovering dependencies involving out-of-vocabulary words.

To show the effectiveness of our model in recovering dependencies involving out-of-vocabulary words, we compare the UAS and LAS of out-of-vocabulary words between our hierarchical LSTM network model and bidirecitonal LSTM network model proposed by [18] on CTB5. As shown in table 3, incorporating character-level information makes our model achieve better UAS and LAS of out-of-vocabulary words. We observed a 3.96% rise in UAS and 5.88% rise in LAS respectively on recovering dependencies involving out-of-vocabulary words. A t-test on the difference shows the improvement is statistically significant.

Method	Penn-SD		CTB5	
	UAS	LAS	UAS	LAS
Basic model	92.81	90.18	79.33	75.55
+character	93.63	91.12	82.99	79.64
+POS	94.08	91.82	87.55	86.23
+POS+character	94.13	91.85	87.77	86.42

Table 4. The impact of POS tags and character-level information on CTB5 and Penn-SD.

We further examine POS tags and character-level information that account for the performance of our parser. As shown in table 4, our basic model uses only lookup-based word representations and shows the worst performance. Using character-level information and POS tags do improve our basic model and lets our model achieve competitive accuracy. Moreover, we find that character-level information makes greater improvement when POS tags are not provided. Again we observe a much bigger improvement in Chinese than in English when introducing the character-level representation of words. In addition, although using character-level information could improve model to a certain extent, POS tags is still necessary for dependency parsing. It seems that the introduction of POS information contributes more in Chinese than in English as well. This is, however, not strictly comparable, since we use gold standard POS tag for Chinese and automatically generated POS tag for English as most previous work did.

We also test out model without pre-trained word embeddings, our model achieves 93.52% UAS / 91.23% LAS on Penn-SD and 86.62% UAS / 85.11% LAS

on CTB5. Using pre-trained word embeddings can obtain around 0.6%∼1.1% improvement.

4 Conclusion

In this paper, we propose a hierarchical LSTM network model for graph-based dependency parsing. Word-level and character-level LSTM networks are utilized to capture information regarding the character level, the word form and POS tag, as well as the sentential context it appears in, which allows for a improvement on Chinese and lets our model achieve state-of-the-art accuracy. Moreover, our model is still a first-order model using standard Eisner algorithm for decoding, the computational cost remains at a lowest level among graph-based models.

As further work, we will improve our model to generate nonprojective trees and test our model on morphologically rich languages which are often nonprojective dependencies.

Acknowledgments

This work is supported by National Key Basic Research Program of China under Grant No.2014CB340504 and National Natural Science Foundation of China under Grant No.61273318. The Corresponding author of this paper is Baobao Chang.

References

1. Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330.
2. Naiwen Xue, Fei Xia, Fu-Dong Chiou, and Marta Palmer. 2005. The penn chinese treebank: Phrase structure annotation of a large corpus. *Natural language engineering*, 11(02):207–238.
3. 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 54rd Annual Meeting of the Association for Computational Linguistics*.
4. Miguel Ballesteros, Chris Dyer, and Noah A. Smith. 2015a. Improved transition-based parsing by modeling characters instead of words with lstms. *Computer Science*.
5. Miguel Ballesteros, Chris Dyer, and Noah A. Smith. 2015b. 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, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 349–359.
6. Jonas Kuhn Bernd Bohnet. 2012. The best of both worlds: a graph-based completion model for transition-based parsers. *Conference of the European Chapter of the Association for Computational Linguistics*.

7. Danqi Chen and Christopher D. Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 740–750.
8. James Cross and Liang Huang. 2016. Incremental parsing with minimal features using bi-directional lstm. In *Proceedings of the 54rd Annual Meeting of the Association for Computational Linguistics*.
9. John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, pages 2121–2159.
10. 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*, pages 334–343.
11. David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. 2003. English gigaword. *Linguistic Data Consortium, Philadelphia*.
12. Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.
13. Wang Ling, Chris Dyer, Alan Black, and Isabel Trancoso. 2015a. Two/too simple adaptations of word2vec for syntax problems. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.
14. Wang Ling, Chris Dyer, Alan W. Black, Isabel Trancoso, Ramon Fermandez, Silvio Amir, Luís Marujo, and Tiago Luís. 2015b. 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, EMNLP 2015*, pages 1520–1530.
15. Marie Catherine De Marneffe, Bill Maccartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. *Lrec*, pages 449–454.
16. Wenzhe Pei, Tao Ge, and Baobao Chang. 2015. An effective neural network model for graph-based dependency parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*, pages 313–322.
17. Kristina Toutanova, Dan Klein, Christopher D Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 173–180. Association for Computational Linguistics.
18. Wenhui Wang and Baobao Chang. 2016. Graph-based dependency parsing with bidirectional lstm. In *Proceedings of the 54rd Annual Meeting of the Association for Computational Linguistics*.
19. 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*.
20. Hiroyasu Yamada and Yuji Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proceedings of IWPT*, volume 3, pages 195–206.
21. Yue Zhang and Stephen Clark. 2008. A tale of two parsers: Investigating and combining graph-based and transition-based dependency parsing. In *2008 Conference on Empirical Methods in Natural Language Processing*, pages 562–571.

22. Hao Zhang and Ryan T. McDonald. 2014. Enforcing structural diversity in cube-pruned dependency parsing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 656–661.
23. Yue Zhang and Joakim Nivre. 2011. Transition-based dependency parsing with rich non-local features. In *The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 188–193.
24. Yuan Zhang and David Weiss. 2016. Stack-propagation: Improved representation learning for syntax. In *Proceedings of the 54rd Annual Meeting of the Association for Computational Linguistics*.