A Transition-based System for Universal Dependency Parsing

Hao Wang[1,2], **Hai Zhao**[1,2,*], **Zhisong Zhang**[1,2]
[1]Department of Computer Science and Engineering, Shanghai Jiao Tong University
[2]Key Laboratory of Shanghai Education Commission for Intelligent Interaction and Cognitive Engineering, Shanghai Jiao Tong University, Shanghai, 200240, China
{wanghao.ftd, zzs2011}@sjtu.edu.cn, zhaohai@cs.sjtu.edu.cn,

Abstract

This paper describes the system for our participation of team *Wanghao-ftd-SJTU* in the *CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*. In this work, we design a system based on UDPipe[1] for universal dependency parsing, where transition-based models are trained for different treebanks. Our system directly takes raw texts as input, performing several intermediate steps like tokenizing and tagging, and finally generates the corresponding dependency trees. For the special surprise languages for this task, we adopt a delexicalized strategy and predict based on transfer learning from other related languages. In the final evaluation of the shared task, our system achieves a result of 66.53% in macro-averaged LAS F1-score.

1 Introduction

Universal Dependencies (UD) (Nivre et al., 2016, 2017b) and universal dependency parsing take efforts to build cross-linguistically treebank annotation and develop cross-lingual learning to parse many languages even low-resource languages. Universal Dependencies release 2.0[2] (Nivre et al., 2017b) includes rich languages and treebanks resources and the parsing task in CoNLL 2017 is based on this dataset. In fact, dependency parsing has been adopted as topic of the shared task in CoNLL-X and CoNLL-2007 (Buchholz and Marsi, 2006; Nivre et al., 2007), which have been the milestones for the researching field of parsing. This time, the task is taking a universal annotation version and trying to exploit cross-linguistic similarities between various languages.

In this paper, we describe the system of team *Wanghao-ftd-SJTU* for the *CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies* (Zeman et al., 2017). For this task, we only use provided treebanks to train models without any other resources including pretrained embeddings.

For dependency parsing, there have been two major parsing methods: graph-based and transition-based. The former searches for the final tree through graph algorithms by decomposing trees into factors, utilizing ingenious dynamic programming algorithms (Eisner, 1996; McDonald et al., 2005; McDonald and Pereira, 2006); while the latter parses sentences by making a series of shift-reduce decisions (Yamada and Matsumoto, 2003; Nivre, 2003). In our system, we will utilize the transition-based system for its simplicity and relatively lower computation cost.

Transition-based dependency parsing takes linear time complexity and utilizes rich features to make structural prediction (Zhang and Clark, 2008; Zhang and Nivre, 2011). Specifically, a buffer for input words, a stack for partially built structure and shift-reduce actions are basic elements in a transition-based dependency parsing. For the transition systems of dependency parsing, there have been two major ones: *arc-standard* and *arc-eager* (Nivre, 2008). Our system adopts the former, whose basic algorithm can be described as

[*]Correspondence author. This paper was partially supported by Cai Yuanpei Program (CSC No. 201304490199 and No. 201304490171), National Natural Science Foundation of China (No. 61170114, No. 61672343 and No. 61272248), National Basic Research Program of China (No. 2013CB329401), Major Basic Research Program of Shanghai Science and Technology Committee (No. 15JC1400103), Art and Science Interdisciplinary Funds of Shanghai Jiao Tong University (No. 14JCRZ04), Key Project of National Society Science Foundation of China (No. 15-ZDA041).

[1]http://ufal.mff.cuni.cz/udpipe

[2]http://universaldependencies.org/

following:

Start : $\sigma = [ROOT], \beta = w_1, ..., w_n, A = \emptyset$
1. Shift :
$\sigma, w_i \mid \beta, A \rightarrow \sigma \mid w_i, \beta, A$
2. Left-Arc$_r$:
$\sigma \mid w_i \mid w_j, \beta, A \rightarrow \sigma \mid w_j, \beta, A \cup r(w_j, w_i)$
3. Right-Arc$_r$:
$\sigma \mid w_i \mid w_j, \beta, A \rightarrow \sigma \mid w_i, \beta, A \cup r(w_i, w_j)$
Finish : $\sigma = [w], \beta = \emptyset$

where σ, β, A represent the stack, queue and the actions respectively.

One major difference for parsing between the situation of current and that of ten years ago is that recently we have seen a rising of neural network based methods in the field of Natural Language Processing and parsing has also been greatly changed by the neural methods. With distributed representation for words and sentences and the powerful non-linear calculation ability of the neural networks, we could explore deeper syntactic and maybe semantic meaning in text analysis, and both graph-based (Pei et al., 2015; Wang and Chang, 2016) and transition-based (Chen and Manning, 2014; Weiss et al., 2015; Dyer et al., 2015; Andor et al., 2016) parsing have benefited a lot from neural representation learnings. In our system, the model, which is trained by *UDPipe*, for the transition action predictor is also based on neural network, which is similar to the one of Chen and Manning (2014).

For this shared task, our system is built based on UDpipe (Straka et al., 2016), which provides a pipeline from raw text to dependency structures, including a tokenizer, taggers and the dependency predictor. We trained and tuned the models on different treebanks, and in the final evaluation, a score of 66.53% in macro-averaged LAS F1-score measurement is achieved. In the task, there are several surprise languages which lack of annotated resources, which means it is hard to train specified models for those languages. To tackle this problem, we exploit the universal part-of-speech (POS) tags, which could be represented as cross-lingual knowledge to avoid language-specific information, and adopting a delexicalized and cross-lingual method, which relies solely on universal POS tags and annotated data in close-related languages.

The rest of the paper is organized as follows: Section 2 describes our system overview, Section 3 elaborates the components of the system, Section 4 shows the experiments and results for our participation in the shared task, and Section 5 concludes this paper.

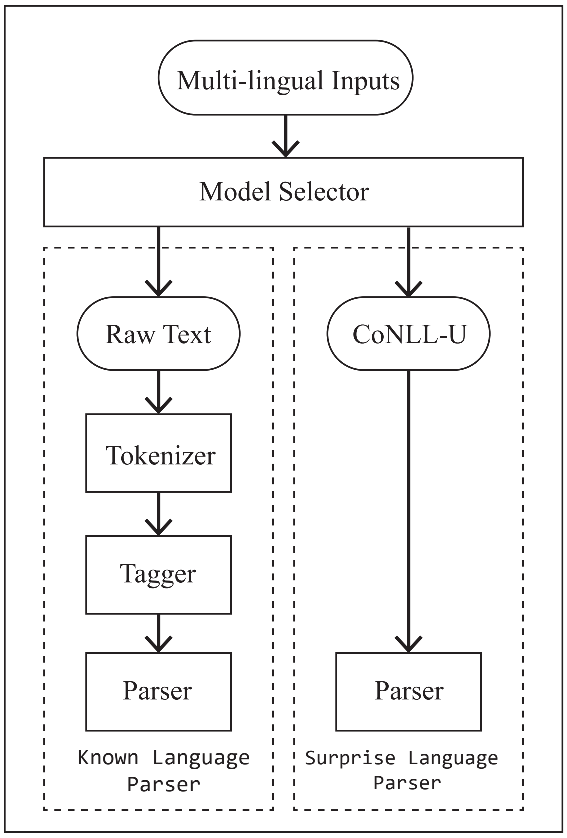

Figure 1: System overview.

2 System Overview

The overall architecture of our universal dependency parser is shown in Figure 1. The whole system can be divided into two parts: *Known Language Parser* and *Surprise Language Parser*. The former deals with known languages, including rich resource treebanks and low resource treebanks, whose annotations as the training data are accessible, while the latter disposes of the ones without dependency annotations. When the text to be processed by the system is inputed, it is first discriminated as rich-resource or low-resource and then dispatched to the corresponding sub-systems, which will be described as follows.

For the **Known Language Parser**, the related pipeline contains three steps as follow.

(1) **Tokenizer** The raw texts are split into basic units for the latter processing of dependency anal-

ysis, which is the main task of the tokenizer. For all rich resource languages, we train tokenziers using provided training data, including the languages which can be easily tokenized by specific delimiters.

(2) **Tagger** The tokenized texts are labeled by taggers, which provides them with the tags which will be utilized in the later dependency analysis, such as POS and morphological features. Like the previous step, we train taggers for all the rich resource languages.

(3) **Dependency Parser** Tokens and linguistic features generated by taggers are put into the dependency parser to generate the final dependency structures.

For **Surprise Language Parser**, only *Dependency Parser* is needed. We directly take the provided CoNLL-U files which already include the tokens and features as inputs and predicts the results. Without annotated training data, we could not train the tokenizers and taggers for these languages; Meanwhile for the parsing, we adopt a delexicalized and cross-lingual strategy, which will be described later in Section 3.3.

3 System Components

3.1 Model Selector

In the final testing phase of the shared task, there are mainly three types of test data (Nivre et al., 2017a), *Ordinary Provided Resource Test Set* which have corresponding training datasets , *Parallel Test Set* which concerns selected known languages but may have different domain from their training data and *Surprise Languages* whose training annotations are not available in the provided dataset. The model selector aims to discriminate these different input types, and dispatch the inputs to different sub-systems. Specifically, for the first two types which we refer to as Known Language, they will be dealt by the *Known Language Parser*, while the *Surprise Language Parser* will dispose with the surprise languages.

As for *Parallel Test Set*, we use its corresponding treebank without specific domain in treebank name.[3]

[3]It may be better if we use the whole treebanks of corresponding language

Parameter Name	Value
tokenize_url	1
allow_spaces	1
iterations	20
batch_size	50
learning_rate	0.005
dropout	0.1

Table 1: Parameters for the training of tokenizers.

Parameter Name	Value
guesser_suffix_rules	8
guesser_prefixes_max	4
guesser_prefix_min_count	10
guesser_enrich_dictionary	0
iterations	20
dimension of upostag	20
dimension of feats	20
dimension of xpostag	20
dimension of form	50
dimension of deprel	20

Table 2: Parameters for the training of taggers.

3.2 Known Language Parser

3.2.1 Tokenizer

In the *Known Language Parser*, the first step is to tokenize the input raw text, generating the basic units for later processing. We train tokenizers for all the languages using *UDPipe*, including those ones which are quite easy to separate using simple rules, like identifying the blank spaces in English. Considering there are some languages that could not be simply tokenized by blank spaces, we adopt this unified treatment for this step. The tokenizers are trained mainly using the *SpaceAfter* features provided in the CoNLL-U files and the parameters of *UDPipe Tokenizer* are shown in Table 1.

3.2.2 Tagger

In the pipeline of dealing known languages, the second step is to provide several light-weighted syntactical and morphological features for the tokenized texts, which will be utilized as the input features in the final parsing step. In our system, we adopt the tagger in *UDPipe*, whose tagging method is based on *MorphoDita* (Straková et al., 2014) and the training method is the classical Averaged Perceptron (Collins, 2002), and the training parameters of *UDPipe Tagger* are provided in Table 2. In this step, the tagger will provide the following outputs:

1. *Lemma*: Lemma or stem of word forms.
2. *UPOS*: Universal POS tags.
3. *XPOS*: Language-specific POS tags.
4. *FEATS*: Morphological features from the universal feature inventory or from a defined language-specific extension.

These features will be used as inputs in the final parsing step for Rich Resource Languages.

3.2.3 Dependency Parser

For the final step, we generate the final dependency outputs with the tokens and features generated by the pre-trained POS taggers. The parser uses *Parsito* (Straka et al., 2015b). *Parsito*[4] is a transition-based parser with neural network classifier, which is similar to the one of (Chen and Manning, 2014). The inputs to the model represent the current configuration of the stack and buffer, including features of the top three nodes on both of them and child nodes of the nodes on the stack. After we projected features to embeddings and concatenated the generated embeddings to representations of features, the vector representations of the input are fed to a hidden layer activated with *tanh*, and the output layer is softmax indicating the probabilities of each possible transition actions.

The parser supports projective and non-projective dependency parsing, which is configured by the option *transition_system*. In Universal Dependencies release 2.0, only UD_Japanese and UD_Galician have no non-projective dependency trees; while UD_Chinese, UD_Polish and UD_Hebrew have a few non-projective trees, around 1% in the treebanks. According to the projective tree quantities of the whole treebanks[5], we train non-projective parsing for most treebanks except UD_Japanese and UD_Galician. In projective parsing, we use dynamic oracle which usually performs better but more slowly. In non-projective parsing, we use static_lazy and search-based oracle (Straka et al., 2015a).

Except *transition_system* option, other configurations of *Parsito* are the same in all the training of different treebanks. For the structured_interval option, we kept the default value 8. To make sure that there is a only single root when parsing, *single_root* option is set to 1. Transition-based method could employ rich features effectively. In our system, we use the linguistic features generated by previous taggers, including lemma, POS tags and morphological features as described in Section 3.2.2. The parameters for the parser training are shown in Table 3.

[4]http://ufal.mff.cuni.cz/parsito

[5]Projective tree extraction script is from https://github.com/ftyers/ud-scripts

Parameter Name	Value
iteration	20
hidden_layer	200
batch_size	10
learning_rate	0.1
dimension of upostag	20
dimension of feats	20
dimension of xpostag	20
dimension of form	50
dimension of deprel	20

Table 3: Parameters for the training of parsers.

Surprise Language	Source Language
Buryat	Turkish
Kurmanji	Persian
North Sami	Finnish
Upper Sorbian	Czech

Table 4: Surprise languages and corresponding source languages.

3.3 Surprise Language Parser

This sub-system deals with the surprise languages without enough training data. We use a simple delexicalized and cross-lingual method, that is, parsing these low resource languages based on the models learned from other languages. This follows the method of (Zeman and Resnik, 2008), which shows that transfer learning for another language based on delexicalized parser can perform well. Although different languages may have different word forms, the underlying syntactic information could overlap and the universal POS tags could be utilized to explore the correlations. To achieve this, we train a dependency parser in a close-relation language (source language) for a surprise language, and then feed the delexicalized POS tag sequence of the surprise language to the source language parser. We consider language family and close area to find the source language for surprise language. Table 4 shows surprise languages and their corresponding source languages we found.

We use delexicalized source language parser to

Treebank's Type	LAS
Big treebanks	71.20
PUD treebanks	65.55
Small treebanks	52.13
Surprise languages	34.49

Table 5: Results of main types of treebanks.

Surprise Language	LAS	UAS
Buryat	28.11	49.67
Kurmanji	19.85	30.49
North Sami	33.39	45.60
Upper Sorbian	56.60	64.33
Macro-average	34.49	47.52

Table 6: Final LAS scores for the surprise languages.

predict the surprise language's dependency structures. The input is CoNLL-U file in which we filter other linguistic features except Universal POS tags. Blind test results of surprise language are showed in Section 4.

Results	Our System	Best
LAS	66.53	76.30
UAS	72.69	81.30
CLAS	60.85	72.57
UPOS	90.63	93.09
XPOS	79.64	82.27
Morphological Features	82.27	82.58
Morphological Tags	73.81	73.92
Lemmas	81.63	83.74
Sentence segmentation	88.40	89.10
Word Segmentation	98.55	98.81
Tokenization	98.81	98.95

Table 7: Final results for the task.

4 Results

Evaluation process of this shared task is deployed in TIRA [6] (Potthast et al., 2014). LAS is the main scoring metric and we show performances of our system in several types of treebanks in Table 5 using the same groups as the official results. What's more, LAS of our system in Surprise Languages are shown in Table 6. We show several official evaluation results such as LAS, UAS and other results and compared with best results in Table 7.

[6]http://www.tira.io/tasks/conll

5 Conclusion

In this paper, we describe the universal dependency parser for our participation in the CoNLL 2017 shared task. The official evaluation shows that our system achieves 66.53% in macro-averaged LAS F1-score measurement on the official blind test. Further improvements could be obtained by more carefully fine-tuning models and adopting more sophisticated neural models.

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)*. Berlin, Germany, pages 2442–2452.

Sabine Buchholz and Erwin Marsi. 2006. Conll-x shared task on multilingual dependency parsing. In *Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL-X)*. Association for Computational Linguistics, New York City, pages 149–164.

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)*. Doha, Qatar, pages 740–750.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*. pages 1–8.

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)*. Beijing, China, pages 334–343.

Jason M. Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of the 16th International Conference on Computational Linguistics*. Copenhagen, pages 340–345.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Online large-margin training of dependency parsers. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*. Ann Arbor, Michigan, pages 91–98.

Ryan McDonald and Fernando Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *Proceedings of the 11th Conference of the European Chapter of the ACL (EACL 2006)*. pages 81–88.

Joakim Nivre. 2003. An efficient algorithm for projective dependency parsing. In *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT)*. pages 149–160.

Joakim Nivre. 2008. Algorithms for deterministic incremental dependency parsing. *Computational Linguistics* 34(4):513–553.

Joakim Nivre, Željko Agić, Lars Ahrenberg, et al. 2017a. Universal dependencies 2.0 CoNLL 2017 shared task development and test data. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics, Charles University. http://hdl.handle.net/11234/1-2184.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajič, Christopher Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. 2016. Universal Dependencies v1: A multilingual treebank collection. In *Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC 2016)*. Portoro, Slovenia, pages 1659–1666.

Joakim Nivre, Johan Hall, Sandra Kübler, Ryan McDonald, Jens Nilsson, Sebastian Riedel, and Deniz Yuret. 2007. The CoNLL 2007 shared task on dependency parsing. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*. Association for Computational Linguistics, Prague, Czech Republic, pages 915–932.

Joakim Nivre et al. 2017b. Universal Dependencies 2.0. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics, Charles University, Prague. http://hdl.handle.net/11234/1-1983.

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 and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Beijing, China, pages 313–322.

Martin Potthast, Tim Gollub, Francisco Rangel, Paolo Rosso, Efstathios Stamatatos, and Benno Stein. 2014. Improving the reproducibility of PAN's shared tasks: Plagiarism detection, author identification, and author profiling. In Evangelos Kanoulas, Mihai Lupu, Paul Clough, Mark Sanderson, Mark Hall, Allan Hanbury, and Elaine Toms, editors, *Information Access Evaluation meets Multilinguality, Multimodality, and Visualization. 5th International Conference of the CLEF Initiative (CLEF 14)*. Berlin Heidelberg New York, pages 268–299.

Milan Straka, Jan Hajic, Jana Straková, and Jan Hajic jr. 2015a. Parsing universal dependency treebanks using neural networks and search-based oracle. In *International Workshop on Treebanks and Linguistic Theories (TLT14)*. page 208.

Milan Straka, Jan Hajič, and Jana Straková. 2016. UDPipe: trainable pipeline for processing CoNLL-U files performing tokenization, morphological analysis, pos tagging and parsing. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*. Paris, France.

Milan Straka, Jan Hajič, Jana Straková, and Jan Hajič jr. 2015b. Parsing universal dependency treebanks using neural networks and search-based oracle. In *Proceedings of Fourteenth International Workshop on Treebanks and Linguistic Theories (TLT 14)*.

Jana Straková, Milan Straka, and Jan Hajič. 2014. Open-Source Tools for Morphology, Lemmatization, POS Tagging and Named Entity Recognition. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. Baltimore, Maryland, pages 13–18.

Wenhui Wang and Baobao Chang. 2016. Graph-based dependency parsing with bidirectional lstm. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany, pages 2306–2315.

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)*. Beijing, China, pages 323–333.

Hiroyasu Yamada and Yuji Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT)*. pages 195–206.

Daniel Zeman, Martin Popel, Milan Straka, Jan Hajič, Joakim Nivre, Filip Ginter, Juhani Luotolahti, Sampo Pyysalo, Slav Petrov, Martin Potthast, Francis Tyers, Elena Badmaeva, Memduh Gökırmak, Anna Nedoluzhko, Silvie Cinková, Jan Hajič jr., Jaroslava Hlaváčová, Václava Kettnerová, Zdeňka Urešová, Jenna Kanerva, Stina Ojala, Anna Missilä, Christopher Manning, Sebastian Schuster, Siva Reddy, Dima Taji, Nizar Habash, Herman Leung, Marie-Catherine de Marneffe, Manuela Sanguinetti, Maria Simi, Hiroshi Kanayama, Valeria de Paiva, Kira Droganova, Hěctor Martínez Alonso, Hans Uszkoreit, Vivien Macketanz, Aljoscha Burchardt, Kim Harris, Katrin Marheinecke, Georg Rehm, Tolga Kayadelen, Mohammed Attia, Ali Elkahky, Zhuoran Yu, Emily Pitler, Saran Lertpradit, Michael

Mandl, Jesse Kirchner, Hector Fernandez Alcalde, Jana Strnadova, Esha Banerjee, Ruli Manurung, Antonio Stella, Atsuko Shimada, Sookyoung Kwak, Gustavo Mendonça, Tatiana Lando, Rattima Nitisaroj, and Josie Li. 2017. CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*. Association for Computational Linguistics.

Daniel Zeman and Philip Resnik. 2008. Cross-language parser adaptation between related languages. In *IJCNLP*. pages 35–42.

Yue Zhang and Stephen Clark. 2008. A tale of two parsers: Investigating and combining graph-based and transition-based dependency parsing. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*. Honolulu, Hawaii, pages 562–571.

Yue Zhang and Joakim Nivre. 2011. Transition-based dependency parsing with rich non-local features. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Portland, Oregon, USA, pages 188–193.