How Important is POS to Dependency Parsing? Joint POS Tagging and Dependency Parsing Neural Networks

Hsuehkuan Lu, Lei Hou, and Juanzi Li

[1] DCST, Tsinghua University, Beijing 100084, China
[2] KIRC, Institute for Artificial Intelligence, Tsinghua University
[3] Beijing National Research Center for Information Science and Technology
s810142000@gmail.com; {houlei,lijuanzi}@tsinghua.edu.cn

Abstract. It is widely accepted that part-of-speech (POS) tagging and dependency parsing are highly related. Most state-of-the-art dependency parsing methods still rely on the results of POS tagging, though the tagger is not perfect yet. Inevitably, dependency parsing model will encounter performance degradation due to the error propagation problems. And it still remains uncertain about how important POS tagging is to dependency parsing. In this work, we propose a method to jointly learn POS tagging and dependency parsing so as to alleviate the error propagation problems. Our proposed method is based on transition system, which is capable to produce dependency tree efficiently and accurately. The results reported in the experiments support our idea that POS tagging is a crucial syntactic component for dependency parsing.

Keywords: Dependency parsing · Part-of-speech tagging · Joint learning.

1 Introduction

Both POS tagging and dependency parsing are fundamental tasks of natural language processing (NLP), which are beneficial to various downstream applications such as relation extraction [8, 5], text summarization [14, 23], machine translation [7], and named entity recognition [11, 13]. These two tasks are highly related, and many dependency parsing methods use POS tags as essential features. However, using automatically predicted POS tags triggers the error propagation problem, which degrades the performance of parser. In order to relieve the error propagation problem, we propose a method to jointly learn dependency parsing and POS tagging.

Either traditional or neural network methods use the information of POS tags, which are generated automatically by taggers. Currently, POS taggers are not perfect yet, and the taggers may be irrelevant to the sentences to be parsed. Some works try to avoid using POS tags for dependency parsing, instead they explore the usage of lexical information [2, 10]. However, the parsers achieving the highest performance still rely on the usage of POS tags [1, 16, 9]. Therefore, joint modeling POS tagging and dependency parsing is then proposed to improve both tagging and parsing results [17, 12, 4, 26, 27, 25, 19]. The work of Li [17] reports that dependency accuracy drops around 6% on Chinese applying automatically labelled POS tags instead of correct POS tags.

Typically, dependency parsing models can be classified into graph-based [18] and transition-based [20] methods. Graph-based method is able to traverse all the potential solutions, while transition-based method restricts the search of solutions. Overall, graph-based method is able to achieve higher performance than transition-based method in most cases, but with extra cost of computation. Traditional graph-based or transition-based models define a set of features such as lexical features, POS tagging features, dependency label features, and word correlation features [18, 22, 3, 28]. Nevertheless, defining feature template is time-consuming and requires linguistic expertise's efforts. In the work reported by Bohet et.al [3], their parser spends 99% of its time processing feature extraction, though they use standard efficient ways. Recent state-of-the-art models adopt neural architectures to replace feature-engineering [6, 24, 29, 1, 10, 16, 9].

In this work, we propose a neural networks architecture to jointly learn POS tagging and dependency parsing based on transition-based algorithm. We first treat POS tagging as downstream task, and stack dependency parsing on the results of POS tagging. Our method extends BIST transition-based dependency parser [16] with extra POS tagging component. Our proposed method is relatively lightweight which uses shallow bidirectional-LSTMs (BiLSTMs) and applies greedy strategy to generate dependency parsing. Despite the simplicity of our neural architecture, our method is capable to capture both lexical and syntactic information. In the experimental results, our method can achieve comparable results (-1.4% in LAS) with automatically generated POS tags on Universal Dependency (UD) 1.2, and outperforms baseline on UD 2.0 by 3.3% (LAS). Moreover, our model obtains the state-of-the-art scores on UD 1.2 with ground-truth POS tags by leading 2% (LAS), and nearly close to the overall state-of-the-art results on UD 2.0 (-0.5% in LAS). At last, the ablation tests reveal the importance of each component (n-gram, character, and POS) to dependency parsing, and validate our idea of joint learning.

2 Method

In this section, we present our joint model of POS tagging and transition-based dependency parsing. Given a sentence $T = \langle w_1, w_2, \ldots, w_K \rangle$ of length K, where w is word token, our goal is to predict a sequence of POS tags $P = \langle p_1, p_2, \ldots, p_K \rangle$ and dependency labels $L = \langle l_1, l_2, \ldots, l_K \rangle$, as shown in Fig. 1.

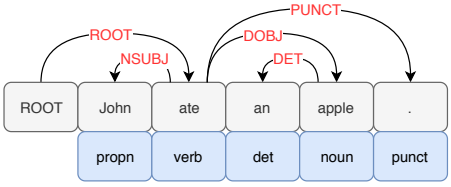

Fig. 1. The example sentence "John ate an apple." including Part-of-speech tags and dependency relations.

Our proposed neural architecture is shown in Fig. 2, which is composed of three stages to generate the word representations used in dependency parsing, i.e., **word representations**, **POS tagging** and **joint representations**.

2.1 Joint Model of POS Tagging and Dependency Parsing

The joint model starts with a BiLSTM layer to learn vectors representing word tokens for POS tagging purpose. In the stage of tagging, we adopt multi-layer perceptron (MLP) as classifier to predict POS tags. Then we integrate POS information to lexical information. Based on the idea of multi-task learning, we hypothesize that sharing the bottom neural layer is beneficial for model to learn more general vector representations across multiple tasks, which are POS tagging and dependency parsing in our case.

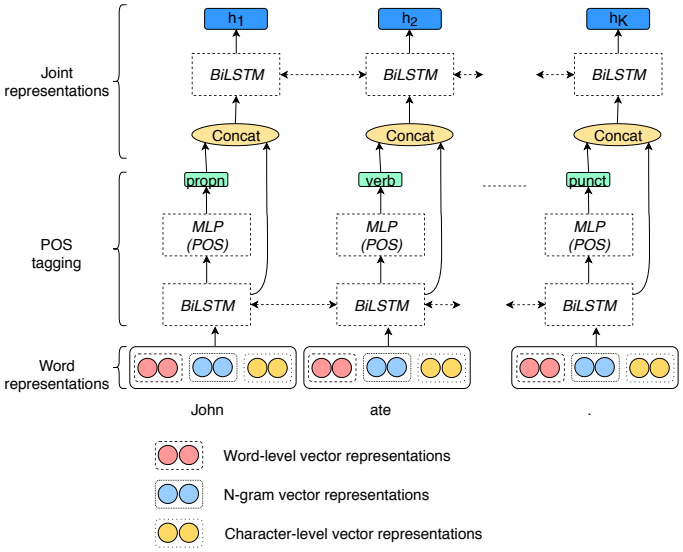

Fig. 2. The overall architecture of our joint model to generate word representations for dependency parsing.

Word representation. The first stage generates word features, including word token, n-gram, and characters. As shown in Fig. 2, we follow Kiperwasser [16] to learn $v_{w_i}(i = 1, 2, \ldots)$ using BiLSTM, where v is the vector representations of word token w_i. Additionally, we attempt to extend lexical information by adding n-gram features. In the inputs we use the concatenations of word embeddings, n-gram representations, and character-level representations. For instance, given a word "John", we extract three types of features, including word features [John], n-gram features [$\langle$Jo, Joh, ohn, hn$\rangle$], and character features [J, o, h, n]. In this example, we use 3-gram as illustration, and we add a start token $\langle$ and an end token $\rangle$ in n-gram features as to better capture the po-

sitional information. In the experiments, all the parameters are initialized from uniform distribution randomly, and we do not use pre-trained embeddings.

We apply respective BiLSTMs to learn the representations of n-gram and character, and we simply adopt the concatenation of last hidden states in forward and backward LSTM as representations. Afterwards, the designed word representations are applied in POS tagging task to capture the syntactic information during training.

POS tagging. We consider POS tagging task as downstream task, and stack dependency parsing upon the results of tagging. As shown in Fig. 2, we treat POS tagging as a sequence labeling task, and each process of tagging is an N-way classification. In order to simplify the task, we use MLP as classifier. BiLSTM cell is capable to model neighboring information, and suppose to obtain reliable results.

Given a sentence $T = \langle w_1, w_2, \ldots, w_K \rangle$, POS tagging task is to label a sequence of POS tags $P = \langle p_1, p_2, \ldots, p_K \rangle$. The prediction is formally defined in Equation 1

$$\hat{p}_i = \text{MLP}_{pos}(v_{w_i}), \hat{p}_i \in \mathbb{R}^{N_{pos}} \tag{1}$$

where $\hat{p}_i$ is the predicted POS tag of word w_i, and MLP_{pos} is the MLP classifier for POS tagging. Here $\hat{p}_i$ is represented by probability distribution of all possible categories of POS tags. The last layer of MLP output uses softmax function to produce the probability distribution. Based on the Equation 1, we define the loss of POS tagging by cross-entropy loss in Equation 2

$$\mathcal{L}_{pos} = \text{cross-entropy}(p_i, \hat{p}) = -\sum p_i \log \hat{p} \tag{2}$$

where p_i is the golden POS tag.

Joint representation. Based on the POS tagging results, we design a method to jointly represent lexical and syntactic information. As stated by Zhang [27], stack-propagation is beneficial to strongly-connected tasks, even without directly using the results of downstream task. In addition to share the bottom neural layer for multiple tasks, we further integrate the results of downstream task (POS tagging) to upstream task (dependency parsing). Similar to POS tagging, we stack another BiLSTM layer for dependency parsing, where the inputs are concatenations of outputs of BiLSTM for POS tagging and predicted POS tags embeddings. The pipeline of joint representations is illustrated in Fig. 2.

Given a sentence $T = \langle w_1, w_2, \ldots, w_K \rangle$, and corresponding predicted POS tags $\hat{P} = (\hat{p}_1, \hat{p}_2, \ldots, \hat{p}_K)$. The formal definition of joint representations is in Equation 3

$$h_i = \text{BiLSTM}_{dep}(v_{w_i}, v_{\hat{p}_i}), i \in \{1, 2, \ldots, K\} \tag{3}$$

where h_i is the concatenation of BiLSTM_{dep} outputs of word w_i, BiLSTM_{dep} is BiLSTM for dependency parsing, $v_{\hat{p}_i}$ is embedding of predicted tag $\hat{p}_i$.

2.2 Transition-based Dependency Parsing

Transition-based dependency parsing aims to predict a transition sequence from an initial configuration to certain terminal configuration, and generates a target dependency parsing tree. During the process of transition, the action is determined by the current

configuration. In this paper, we implement our parsing process in a greedy fashion, which is much more efficient and achieves remarkable accuracy as well.

We apply **arc-standard** transition system [21] as dependency parsing method. Arc-standard is a simple and efficient method to generate dependency parsing tree, though it is incapable to handle non-projective trees which are not discussed in this work. There are three main components in arc-standard system to represent transition configuration. The formal definition of *configuration* is $C = (S, B, A)$, where

1. S: a *stack* holding words to parse dependency labels.
2. B: a *buffer* holding words to be processed.
3. A: a set of *dependency arcs* to record the parsed dependency labels.

The initial configuration for a sentence $T = \langle w_1, w_2, ..., w_K \rangle$ is $S = [\text{ROOT}], B = [w_1, w_2, ..., w_K], A = \varnothing$. The terminal configuration is that stack contains only ROOT node and buffer is empty, then transition system collects all the dependency labels stored in A to generate parsing tree. Let $s_i (i = 1, 2, ...)$, $b_i (i = 1, 2, ...)$ denote the i^{th} element in S and B respectively, the arc-standard system defines three types of transitions:

1. *Left-arc(l)*: add an arc $s_1 \rightarrow s_2$ with dependency label *l*, and pops s_2 out from S. Precondition: $|S| > 2$.
2. *Right-arc(l)*: add an arc $s_2 \rightarrow s_1$ with dependency label *l*, and pops s_1 out from S. Precondition: $|S| \geq 2$.
3. *Shift*: pops b_1 out from B and add to the top of S. Precondition: $|B| \geq 1$.

The whole parsing process costs computations in linear time complexity. Each word requires 2 transitions *(Shift, Left- or Right-arc)*, while the last word with ROOT label requires only *Shift* transition. In Table 1, a complete process of transition sequence is illustrated.

Table 1. An example of the process of transition-based dependency parsing. Use the illustrated sentence "John ate an apple." from Fig. 1.

Transition	Stack	Buffer	A
	[ROOT]	[John, ate, an, apple, .]	$\varnothing$
Shift	[ROOT, John]	[ate, an, apple, .]	
Shift	[ROOT, John, ate]	[an, apple, .]	
Left-arc	[ROOT, ate]	[an, apple, .]	A $\cup$ (ate, NSUBJ, John)
Shift	[ROOT, ate, an]	[apple, .]	
Shift	[ROOT, ate, an, apple]	[.]	
Left-arc	[ROOT, ate, apple]	[.]	A $\cup$ (apple, DET, an)
Right-arc	[ROOT, ate]	[.]	A $\cup$ (ate, DOBJ, apple)
Shift	[ROOT, ate, .]	[]	
Right-arc	[ROOT, ate]	[]	A $\cup$ (ate, PUNCT, .)
Right-arc	[ROOT]	[]	A $\cup$ (ROOT, ROOT, ate)

As to the representations of configuration C, we uses the joint representations of word tokens defined in Equation 3.

Transition actions. The parsing process can be deemed as a sequence of transitions, and determined by current configuration C. In this paper, we define a simple feature template to represent C. There are 4 features in the template, including 3 from S and 1 from B, and the formal definition is in Equation 4

$$\begin{aligned} v_{arc} &= [v_{s_1}; v_{s_2}; v_{s_3}; v_{b_1}] \\ \hat{a} &= \text{MLP}_{arc}(v_{arc}), \hat{a} \in \mathbb{R}^{N_{arc}} \\ \mathcal{L}_{arc} &= \text{cross-entropy}(a_i, \hat{a}) = -\sum a_i \log \hat{a} \end{aligned} \tag{4}$$

where v_{arc} stands for representations of transition action, $\hat{a}$ is the probability distribution of categories of transition action, MLP_{arc} is MLP classifier for transition action, and $\mathcal{L}_{arc}$ is cross-entropy loss of transition action.

Dependency labels. As to the prediction of dependency labels, we hypothesize that label is strongly-connected to the pair of words containing dependency relation. Therefore, we define a feature template so as to better explore the correlation of words. There are 4 features in the template, including first and second words from S, absolute difference of words, and element-wise production of words. The formal definition is in Equation 5

$$\begin{aligned} v_{dep} &= [v_{s_1}; v_{s_2}; |v_{s_1} - v_{s_2}|; v_{s_1} \odot v_{s_2}] \\ \hat{l} &= \text{MLP}_{dep}(v_{dep}), \hat{l} \in \mathbb{R}^{N_{dep}} \\ \mathcal{L}_{dep} &= \text{cross-entropy}(l, \hat{l}) = -\sum l \log \hat{l} \end{aligned} \tag{5}$$

which is similar to Equation 4, the only difference comes from the definition of feature v_{dep}.

2.3 Training

In the training stage, we apply Adam optimizer [15] as our gradient stochastic optimizer. Based on the loss occurred in the classifications of POS tagging from Equation 2, transition action from Equation 4, and dependency label from Equation 5, the final loss is defined in Equation 6

$$\mathcal{L}_{total} = \mathcal{L}_{pos} + \mathcal{L}_{arc} + \mathcal{L}_{dep} + \frac{\lambda}{2}||\theta||^2 \tag{6}$$

where $\frac{\lambda}{2}||\theta||^2$ is the L2-regularization term for all the parameters in the model. In order to accelerate training, we do not tune our model on development dataset, instead we simply use the model training 30 epochs as final model. The size of mini-batch is 150 and learning rate is 0.001.

All embeddings (words, POS tags, n-gram tokens, characters) are initialized from uniform distribution randomly with 50 dimensions. The lengths of character, n-gram are both restricted to 20, and the maximal length of sentence is 100. We use 3-gram as n-gram feature. Dropout is 0.33 for all layers. Each LSTM cell contains 256 units. Each MLP classifier is composed of 2 layers, first layer is fully-connected layer (512 units) with ReLU activation function, and second layer is corresponding categories for each task with softmax function.

3 Experiments

In this section, we first introduce the experiment settings, including datasets, baseline methods and comparison metrics, then report the overall comparison results, and finally analyze the feature contributions via ablation studies.

3.1 Experiment Settings

Datasets. We evaluate our approach on two datasets: Universal Dependencies[1] 1.2, and 2.0, which both contain abundant multi-lingual dependency trees.
Baselines. For UD 1.2 dataset, we report the results from Yang [25], and use seven languages including German (de), English (en), Spanish (es), French (fr), Italian (it), Portuguese (pt) and Swedish (sv). For UD 2.0 dataset, we report the results of 37 datasets from big treebank, and compare with the results reported in CoNLL 2017[2], including baseline method UDPipe 1.1.
Metrics. For POS tagging, we use accuracy based on words as evaluation metric. As to dependency parsing, two evaluation metrics are adopted, which are unlabeled attachment score (UAS) and labeled attachment score (LAS). UAS only counts the correctness of head words, while LAS considers both head words and dependency labels.

3.2 Results

For UD 1.2 dataset, we follow the experimental settings in [27], and report results in Table 2. Compared with baseline methods, our proposed method with automatically generated POS tags performs slightly worse (-1.4%) than the state-of-the-art method. Nevertheless, with the aid of golden POS tags, it achieves a significant improvement by 3.6%, and obtains the state-of-the-art LAS score with over 2% leading. Besides, our proposed method is able to achieve the highest performance in all languages.

Table 2. Dependency parsing results on UD 1.2 dataset with LAS metric. All values are reported in %, and the table includes other 3 results from Yang [25].

Method	de	en	es	fr	it	pt	sv	AVG
Ballesteros et al. [2]	73.00	77.90	77.80	78.00	84.20	80.40	74.50	77.97
Zhang and Weiss [27]	74.20	80.70	80.70	80.00	85.80	80.40	77.50	79.90
Yang et al. [25]	77.10	82.50	82.50	81.20	87.00	83.10	80.40	81.97
Our method (*auto-POS*)	74.60	81.71	81.98	79.82	86.37	81.34	77.77	80.51
Our method (*gold-POS*)	**79.21**	**85.69**	**85.76**	**82.17**	**88.77**	**84.25**	**83.35**	**84.17**

The results for UD 2.0 dataset are reported in Table 3, from which we have the following observations:

[1] http://universaldependencies.org
[2] http://universaldependencies.org/conll17/results.html

– Our method with automatically generated POS tags is better than baseline UDPipe 1.1, with about 3.3% leading in average.
– Because the results from CoNLL 2017 vary from languages (i.e., the leading teams are not always the same), we do not compute the overall best performance. Through comparison on each language, our proposed method can achieve comparable, even better performance. The results show the good generalization capacity of the proposed method.
– The method with golden POS taggers outperforms those of auto-predicted ones by an average increase of 5.7% on all languages, which validates our hypothesis that POS information is beneficial for dependency parsing, and the quality of POS taggers largely influences the performance.

Table 3. Dependency parsing results on UD 2.0 dataset with LAS metric. All values are reported in %, and in each language we report the top 3 results from CoNLL 2017 competition. The first result is marked bold and underlined, while the second result is marked bold.

Method	ar	bg	ca	cs	cu	da	de	el	en	es	et	eu	fa
1st	72.90	**89.81**	**90.70**	**90.17**	**76.84**	**82.97**	**80.71**	**87.38**	**82.23**	**87.29**	**71.65**	**81.44**	**86.31**
2nd	71.96	88.39	88.27	86.52	72.35	**81.55**	77.17	**86.90**	79.94	85.22	**69.71**	**79.61**	**84.90**
3rd	70.70	87.65	88.09	86.50	71.84	79.52	75.47	84.96	79.64	85.16	67.60	77.97	83.34
UDPipe1.1	65.30	83.64	85.39	82.87	62.76	73.38	69.11	79.26	75.84	81.47	58.79	69.15	79.24
auto-POS	**75.18**	85.38	85.16	85.84	72.27	72.96	73.26	77.40	80.88	84.01	54.37	66.76	78.83
gold-POS	**77.71**	**88.64**	**89.89**	**87.35**	**77.05**	79.85	**79.24**	82.81	**85.57**	**86.52**	69.36	74.94	84.68

Method	fi	fr	gl	got	he	hi	hr	id	it	ja	ko	lv	nl
1st	**85.64**	**85.51**	**83.23**	**71.36**	68.16	**91.59**	**85.25**	**79.19**	**90.68**	91.13	**82.49**	**74.01**	**80.48**
2nd	82.38	84.36	**83.22**	68.34	63.94	90.41	**83.15**	78.55	89.08	80.85	**81.10**	**71.35**	75.50
3rd	81.21	83.82	81.60	66.82	63.72	90.40	82.51	77.70	87.85	80.01	79.51	68.03	75.07
UDPipe1.1	73.75	80.75	77.31	59.81	57.23	86.77	77.18	74.61	85.28	72.21	59.09	59.95	68.90
auto-POS	78.64	82.61	78.81	66.58	**78.82**	87.36	76.94	72.68	86.76	**92.67**	68.17	58.06	69.54
gold-POS	**83.17**	**86.50**	82.24	**72.68**	**82.11**	**90.98**	80.03	**79.73**	**90.11**	**95.98**	77.46	67.96	**77.31**

Method	pl	pt	ro	ru	sk	sl	sv	tr	ur	vi	zh	AVG
1st	**90.32**	**87.65**	**85.92**	**83.65**	**86.04**	**91.51**	**85.87**	**62.79**	**82.28**	47.51	68.56	-
2nd	**87.15**	85.11	**84.40**	**83.50**	**81.75**	88.24	**84.98**	**62.66**	81.06	42.52	65.88	-
3rd	86.75	85.01	83.50	81.49	80.53	87.08	82.28	62.39	80.93	42.13	65.15	-
UDPipe1.1	78.78	82.11	79.88	74.03	72.75	81.15	76.73	53.19	76.69	37.47	57.40	72.14
auto-POS	80.82	81.90	79.76	74.00	75.27	81.78	77.25	54.73	76.35	**49.46**	**71.23**	75.47
gold-POS	86.32	**86.64**	82.27	78.50	81.17	**88.29**	83.85	57.52	**84.36**	**63.02**	**80.19**	**81.14**

3.3 Ablation Test

In order to analyze the feature contributions to the joint model, we conduct ablation tests on UD 1.2 dataset. Taking the results in Table 2 as baseline, we remove features respectively, including n-gram, character and POS tags features. The results of ablation tests are shown in Table 4.

Table 4. Ablation tests on UD 1.2 dataset. All values are reported in %, "all lexical" stands for removing both character and n-gram features.

Method	de			en			es			fr		
	UAS	LAS	POS	UAS	LAS	POS	UAS	LAS	POS	UAS	LAS	POS
gold-POS	83.78	79.21	-	88.26	85.69	-	88.64	85.76	-	85.77	82.17	-
auto-POS	80.92	74.60	92.08	85.68	81.71	93.38	85.82	81.98	95.90	84.31	79.82	95.07
- POS	77.17	70.45	-	83.63	78.91	-	84.24	74.18	-	82.43	77.04	-
- character	80.02	73.60	91.58	84.69	80.56	92.98	84.79	80.97	95.51	84.16	79.24	94.90
- n-gram	79.39	73.29	91.46	85.08	80.88	93.23	85.11	80.89	95.17	84.24	79.38	94.67
- all lexical	74.28	66.88	86.98	82.40	77.30	89.56	82.17	77.21	92.66	81.75	75.86	92.72
- all	70.91	62.77	-	80.43	74.86	-	74.18	74.18	-	79.79	73.74	-

Method	it			pt			sv			AVG		
	UAS	LAS	POS	UAS	LAS	POS	UAS	LAS	POS	UAS	LAS	POS
gold-POS	90.91	88.77	-	86.72	84.25	-	86.91	83.35	-	87.28	84.17	-
auto-POS	89.37	86.37	96.36	85.43	81.34	95.74	82.18	77.77	95.01	84.82	80.51	94.79
- POS	86.54	83.20	-	82.19	77.55	-	77.38	71.85	-	81.94	76.17	-
- character	88.36	85.21	96.43	85.09	80.72	95.62	81.60	76.74	94.80	84.10	79.58	94.55
- n-gram	88.29	85.18	96.24	85.21	80.79	95.36	80.66	75.82	94.24	84.00	79.46	94.34
- all lexical	85.49	81.20	94.10	80.02	73.70	90.32	73.46	65.95	86.50	79.94	74.01	90.41
- all	82.28	77.84	-	76.56	69.76	-	69.69	60.86	-	76.26	70.57	-

In this paper, our goal is to analyze the importance of POS tagging to dependency parsing, and it can be observed from the table that after removing the POS tags, the overall performance decreases 3% in UAS, and 3.6% in LAS. The results suggest that lexical features are important to dependency parsing. Specifically, n-gram and character features performs quite similarly, and the overall performance decreases 5% in UAS, 5.5% in LAS, and 4.4% in POS accuracy after removing both features. Although n-gram and character features are close, using them jointly can slightly improve the performance for 1% in both UAS and LAS, and 0.4% in POS accuracy.

The influence of POS tagging varies across different languages. As the languages with weakest POS taggers, de (92.08%) and en (93.38%) are affected largely due to the inadequacy of POS tagger for about 3% drop in UAS and LAS. Furthermore, the importance of POS information can be observed from the table, model with perfect POS tagger (100%) is able to reach 87.28% in UAS and 84.17% in LAS, while without tagger model obtains 81.94% in UAS and 76.17% in LAS. The performance of model declines 5.2% in UAS and 7% in LAS with only the replacement of POS information. The final results validate our hypothesis that POS tagging is vital to dependency parsing.

4 Related Work

In this section, we review the literatures related to our work, i.e., joint learning of POS tagging and dependency parsing and transition-based dependency parsing

Most traditional joint models are implemented with feature templates, and are difficult to analyze the contribution of individual feature [4, 12, 17]. There is another line of works to avoid using POS tagging in dependency parsing [2, 10], e.g., Ballesteros et. al introduce character features to better capture lexical information [2], but it is not contradictive to explore lexical and POS information simultaneously, and in our work we revealed the contributions of these features with extensive experiments.

Our work is inspired by neural networks methods of dependency parsing [6, 24, 29, 1, 10, 16, 9]. The work from Li et.al [17] is the pioneer to jointly learn POS tagging and dependency parsing, but in this work about 6% performance decrease is reported in Chinese. Most recently, Zhang et.al [27], Yang et.al [25], and Nguyen et. al [19] propose joint models to improve both POS tagging and dependency parsing. However, in their research, the impact of POS information to dependency parsing is not explained sufficiently.

5 Conclusion

In this work, we propose a method to jointly learn POS tagging and dependency parsing so as to alleviate the impact of error propagation problem. The experimental results indicate that POS information is significantly influential to dependency parsing, with about 5.2% and 7% differences in UAS and LAS respectively across languages. We also explore the impact of lexical features, and achieve a slight improvement by introducing n-gram feature. Our proposed method is capable to be applied in multiple languages, and is lightweight with shallow neural networks but still obtains high accuracy. Our code are publicly available for further research [4].

Acknowledgement

The work is supported by NSFC key projects (U1736204, 61533018, 61661146007), Ministry of Education and China Mobile Joint Fund (MCM20170301), a research fund supported by Alibaba Group, and THUNUS NExT Co-Lab.

References

1. Andor, D., Alberti, C., Weiss, D., Severyn, A., Presta, A., Ganchev, K., Petrov, S., Collins, M.: Globally normalized transition-based neural networks. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 2442–2452. Association for Computational Linguistics (2016)

[4] https://github.com/hsuehkuan-lu/JointParser

2. Ballesteros, M., Dyer, C., Smith, N.A.: 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. pp. 349–359. Association for Computational Linguistics (2015)
3. Bohnet, B.: Top accuracy and fast dependency parsing is not a contradiction. In: Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010). pp. 89–97. Coling 2010 Organizing Committee (2010)
4. Bohnet, B., Nivre, J.: A transition-based system for joint part-of-speech tagging and labeled non-projective dependency parsing. In: Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning. pp. 1455–1465. Association for Computational Linguistics (2012)
5. Bunescu, R.C., Mooney, R.J.: A shortest path dependency kernel for relation extraction. In: Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing. pp. 724–731. Association for Computational Linguistics (2005)
6. Chen, D., Manning, C.D.: A fast and accurate dependency parser using neural networks. In: Proceedings of the 2014 conference on empirical methods in natural language processing. pp. 740–750 (2014)
7. Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y.: Learning phrase representations using RNN encoder–decoder for statistical machine translation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing. pp. 1724–1734. Association for Computational Linguistics (2014)
8. Culotta, A., Sorensen, J.: Dependency tree kernels for relation extraction. In: Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics. Association for Computational Linguistics (2004)
9. Dozat, T., Manning, C.D.: Deep biaffine attention for neural dependency parsing. In: 5th International Conference on Learning Representations (2017)
10. Dyer, C., Ballesteros, M., Ling, W., Matthews, A., Smith, N.A.: 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). pp. 334–343. Association for Computational Linguistics (2015)
11. Finkel, J.R., Manning, C.D.: Joint parsing and named entity recognition. In: Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics. pp. 326–334. Association for Computational Linguistics (2009)
12. Hatori, J., Matsuzaki, T., Miyao, Y., Tsujii, J.: Incremental joint POS tagging and dependency parsing in Chinese. In: Proceedings of 5th International Joint Conference on Natural Language Processing. pp. 1216–1224. Asian Federation of Natural Language Processing (2011)
13. Jie, Z., Muis, A.O., Lu, W.: Efficient dependency-guided named entity recognition. In: Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence. pp. 3457–3465. AAAI Press (2017)
14. Kikuchi, Y., Hirao, T., Takamura, H., Okumura, M., Nagata, M.: Single document summarization based on nested tree structure. In: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers). pp. 315–320. Association for Computational Linguistics (2014)
15. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv:1412.6980 (2014)
16. Kiperwasser, E., Goldberg, Y.: Simple and accurate dependency parsing using bidirectional LSTM feature representations. Transactions of the Association for Computational Linguistics **4**, 313–327 (2016)

17. Li, Z., Zhang, M., Che, W., Liu, T., Chen, W., Li, H.: Joint models for chinese pos tagging and dependency parsing. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing. pp. 1180–1191. Association for Computational Linguistics (2011)
18. McDonald, R., Crammer, K., Pereira, F.: Online large-margin training of dependency parsers. In: Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05). pp. 91–98. Association for Computational Linguistics (2005)
19. Nguyen, D.Q., Verspoor, K.: An improved neural network model for joint POS tagging and dependency parsing. In: Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies. pp. 81–91. Association for Computational Linguistics (2018)
20. Nivre, J.: An efficient algorithm for projective dependency parsing. In: Proceedings of the Eighth International Workshop on Parsing Technologies. pp. 149–160 (2003)
21. Nivre, J.: Incrementality in deterministic dependency parsing. In: Proceedings of the Workshop on Incremental Parsing: Bringing Engineering and Cognition Together. pp. 50–57. Association for Computational Linguistics (2004)
22. Nivre, J., Hall, J., Nilsson, J., Chanev, A., Eryigit, G., Kübler, S., Marinov, S., Marsi, E.: Maltparser: A language-independent system for data-driven dependency parsing. Natural Language Engineering **13**(2), 95–135 (2007)
23. Rush, A.M., Chopra, S., Weston, J.: A neural attention model for abstractive sentence summarization. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing. pp. 379–389. Association for Computational Linguistics (2015)
24. Weiss, D., Alberti, C., Collins, M., Petrov, S.: 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). pp. 323–333. Association for Computational Linguistics (2015)
25. Yang, L., Zhang, M., Liu, Y., Sun, M., Yu, N., Fu, G.: Joint pos tagging and dependence parsing with transition-based neural networks. IEEE/ACM Trans. Audio, Speech and Lang. Proc. **26**(8), 1352–1358 (2018)
26. Zhang, Y., Li, C., Barzilay, R., Darwish, K.: Randomized greedy inference for joint segmentation, POS tagging and dependency parsing. In: Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. pp. 42–52. Association for Computational Linguistics (2015)
27. Zhang, Y., Weiss, D.: Stack-propagation: Improved representation learning for syntax. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 1557–1566. Association for Computational Linguistics (2016)
28. Zhang, Y., Nivre, J.: 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: Short Papers - Volume 2. pp. 188–193. Association for Computational Linguistics (2011)
29. Zhou, H., Zhang, Y., Huang, S., Chen, J.: A neural probabilistic structured-prediction model for transition-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). pp. 1213–1222. Association for Computational Linguistics (2015)