Is POS Tagging Necessary or Even Helpful for Neural Dependency Parsing?

Houquan Zhou⋆, Yu Zhang⋆, Zhenghua Li, and Min Zhang

Institute of Artificial Intelligence, School of Computer Science and Technology,
Soochow University, Suzhou, China
hqzhou@stu.suda.edu.cn, yzhang.cs@foxmail.com,
{zhli13,minzhang}@suda.edu.cn

Abstract. In the pre deep learning era, part-of-speech tags have been considered as indispensable ingredients for feature engineering in dependency parsing. But quite a few works focus on joint tagging and parsing models to avoid error propagation. In contrast, recent studies suggest that POS tagging becomes much less important or even useless for neural parsing, especially when using character-based word representations. Yet there are not enough investigations focusing on this issue, both empirically and linguistically. To answer this, we design and compare three typical multi-task learning framework, i.e., *Share-Loose*, *Share-Tight*, and *Stack*, for joint tagging and parsing based on the state-of-the-art biaffine parser. Considering that it is much cheaper to annotate POS tags than parse trees, we also investigate the utilization of large-scale heterogeneous POS tag data. We conduct experiments on both English and Chinese datasets, and the results clearly show that POS tagging (both homogeneous and heterogeneous) can still significantly improve parsing performance when using the *Stack* joint framework. We conduct detailed analysis and gain more insights from the linguistic aspect.

1 Introduction

Among different NLP tasks, syntactic parsing is the first to convert sequential utterances into full tree structures. Due to its simplicity and multi-lingual applicability, dependency parsing has attracted extensive research interest as a main-stream syntactic formalism [21, 24], and been widely used in semantic parsing [7], information extraction [22], machine translation [25], etc.

Given an input sentence $S = w_0 w_1 \ldots w_n$, dependency parsing constructs a tree $T = \{(h, d, l), 0 \leq h \leq n, 1 \leq d \leq n, l \in \mathcal{L}\}$, as depicted in Figure 1, where (h, d, l) is a dependency from the head w_h to the dependent w_d with the relation label l, and w_0 is a pseudo root node.

In the pre deep learning (DL) era, part-of-speech (POS) tags are considered as indispensable ingredients for feature engineering in dependency parsing. POS tags function
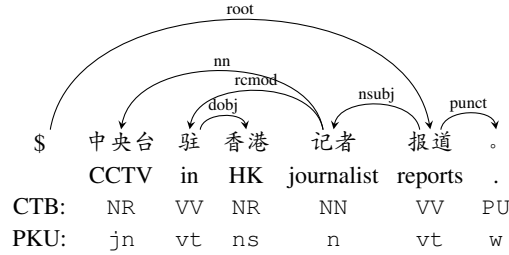

Fig. 1: An example dependency tree with both homogeneous (CTB) and heterogeneous (PKU) POS tags.

as word classes in the sense that the same POS tags usually play similar syntactic roles in language utterances. For example, ordinary verbs are tagged as VV and ordinary nouns as NN in Penn Chinese Treebank (CTB). Parsing performance drops dramatically when removing POS-related features, since POS tags play a key role in reducing the data sparseness problem of using pure word-based lexical features. Meanwhile, to alleviate error propagation in the first-tagging-then-parsing pipeline, researchers propose to jointly model POS tagging and dependency parsing under both graph-based [18] and transition-based [9] frameworks.

In the past five years, dependency parsing has achieved tremendous progress thanks to the strong capability of deep neural networks in representing word and long-range contexts [1, 2, 4, 6, 12, 28]. Yet all those works hold the assumption of POS tags being important and concatenate word and POS tag embeddings as input.

Moreover, researchers show that using Character-level Long Short-term Memory (CHARLSTM) based word representations is helpful for named entity recognition [15], dependency parsing [5], and constituency parsing [13]. The idea is first to perform Long Short-term Memory (LSTM) over word characters and then to add (or concatenate) together word embeddings and CHARLSTM word representations as model inputs. In particular, Kitaev and Klein (2018) [13] show that with CHARLSTM word representations, POS tags are useless for constituency parsing. We believe the reason may be two-fold. First, word embeddings, unlike lexical features, suffer from much less data sparseness, since syntactically similar words can be associated via similar dense vectors. Second, CHARLSTM word representation can effectively capture morphological inflections by looking at lemma/prefix/suffix, which provide similar information as POS tags. However, there still lacks a full and systematic study on the usefulness of POS tags for dependency parsing.

In this work, we try to answer the question whether POS tagging is necessary or even helpful for neural dependency parsing and make the following contributions[1].

---

[1] We release our code at https://github.com/Jacob-Zhou/stack-parser.

- We design three typical multi-task learning (MTL) frameworks (i.e., *Share-Loose*, *Share-Tight*, *Stack*), for joint POS tagging and dependency parsing based on the state-of-the-art biaffine parser.
- Considering that there exist large-scale heterogeneous POS-tag data for Chinese partly because it is much cheaper to annotate POS tags than parse trees, we also investigate the helpfulness of such heterogeneous data, besides homogeneous POS-tag data that are annotated together with parse trees.
- We conduct experiments on both English and Chinese benchmark datasets, and the results show that POS tagging, both homogeneous and heterogeneous, can still significantly improve parsing accuracy when using the *Stack* joint framework. Detailed analysis sheds light on the reasons behind helpfulness of POS tagging.

2 Basic Tagging and Parsing Models

This section presents the basic POS tagging and dependency parsing models separately in a pipeline architecture. In order to make fair comparison with the joint models, we make the encoder-decoder architectures of the tagging and parsing models as similar as possible. The input representation contains both word embeddings and CHARLSTM word representations. The encoder part adopts three BiLSTM layers.

2.1 The Encoder Part

The input layer. Given a sentence $S = w_0 w_1 \ldots w_n$, the input layer maps each word w_i into a dense vector $\mathbf{x}_i$.

$$\mathbf{x}_i = \mathbf{e}_i^w \oplus \mathbf{e}_i^c \tag{1}$$

where $\mathbf{e}_i^w$ is the word embedding, $\mathbf{e}_i^c$ is the CHARLSTM word representation vector, and $\oplus$ means vector concatenation.[2] CHARLSTM word representations $\mathbf{e}_i^c$ are obtained by applying BiLSTM to the word characters and concatenating the final hidden output vectors. Following Dozat and Manning (2017) [4], the word embeddings $\mathbf{e}_i^w$ is the sum of a fixed pretrained word embedding and a trainable word embedding initialized as zero. Infrequent words in the training data (less than 2 times) are treated as a special OOV-token to learn its embedding.

Under the pipeline framework, the parsing model may use extra POS tags as input.

$$\mathbf{x}_i = \mathbf{e}_i^w \oplus \mathbf{e}_i^c \oplus \mathbf{e}_i^p \oplus \mathbf{e}_i^{p\prime} \tag{2}$$

where $\mathbf{e}_i^p$ and $\mathbf{e}_i^{p\prime}$ are the embeddings of the homogeneous and heterogeneous POS tags, respectively. For dropouts, we follow Dozat and Manning (2017) [4] and drop the different components of the input vector $\mathbf{x}_i$ independently.

The BiLSTM encoder. We employ the same $N = 3$ BiLSTM layers over the input layer to obtain context-aware word representations for both tagging and parsing. We follow the dropout strategy of Dozat and Manning (2017) [4] and share the same dropout masks at all time steps of the same unidirectional LSTM. The hidden outputs of the top-layer BiLSTM are used as the encoded word representations, denoted as $\mathbf{h}_i$.

---

[2] We have also tried the sum of $\mathbf{e}_i^w$ and $\mathbf{e}_i^c$, leading to slightly inferior performance.

2.2   The Tagging Decoder

For the POS tagging task, we use two MLP layers to compute the score vector for different tags and get the optimal tag via softmax. The first MLP layer uses leaky ReLU [19] activation, while the second MLP layer is linear without activation. During training, we take the local cross-entropy loss.

2.3   The Parsing Decoder

We adopt the state-of-the-art biaffine parser of Dozat and Manning (2017) [4]. We apply an MLP layer with leaky ReLU activation to obtain the representations of each word as a head ($\mathbf{r}_i^h$) and as a dependent ($\mathbf{r}_i^d$).

$$\mathbf{r}_i^h; \mathbf{r}_i^d = \mathrm{MLP}\left(\mathbf{h}_i\right) \tag{3}$$

As discussed in Dozat and Manning (2017) [4], this MLP layer on the one hand reduces the dimensionality of $\mathbf{h}_i$, and more importantly on the other hand strips away syntax-unrelated information and thus avoids the risk of over-fitting.

Then a biaffine layer is used to compute scores of all dependencies.

$$\mathrm{score}\left(i \leftarrow j\right) = \begin{bmatrix}\mathbf{r}_i^d \\ 1\end{bmatrix}^{\mathrm{T}} W\mathbf{r}_j^h \tag{4}$$

where $\mathrm{score}\left(i \leftarrow j\right)$ is the score of the dependency $i \leftarrow j$, and W is a weight matrix. During training, supposing the gold-standard head of w_i is w_j, we use the cross-entropy loss to maximize the probability of w_j being the head against all words, i.e., $\frac{e^{\mathrm{score}(i \leftarrow j)}}{\sum_{0 \le k \le n} e^{\mathrm{score}(i \leftarrow k)}}$.

For dependency labels, we use extra MLP and Biaffine layers to compute the scores and also adopt cross-entropy classification loss. We omit the details due to space limitation.

3   Joint Tagging and Parsing Models

The pipeline framework suffers from the error propagation problem, meaning that POS tagging mistakes badly influence parsing performance. In the pre-DL era, researchers propose joint tagging and parsing models under both graph-based and transition-based parsing architectures [9, 18]. The key idea is to define the joint score of a tag sequence and a parse tree and to find the optimal joint result in the enlarged search space. In the neural network era, jointly modeling two tasks becomes much easier thanks to the commonly used encoder-decoder architecture and the MTL framework.

In this work, we design and compare three typical MTL frameworks for joint POS tagging and dependency parsing, i.e., *Share-Loose*, *Share-Tight*, and *Stack*, as illustrated in Figure 2. The *Share-Loose* and *Share-Tight* methods treat tagging and parsing as two parallel tasks, whereas the *Stack* method consider parsing as the main task and derive POS tag-related information as the inputs of the parsing component. For all joint models, the inputs only include the word embeddings and CHARLSTM word representations, as shown in Equation 1.
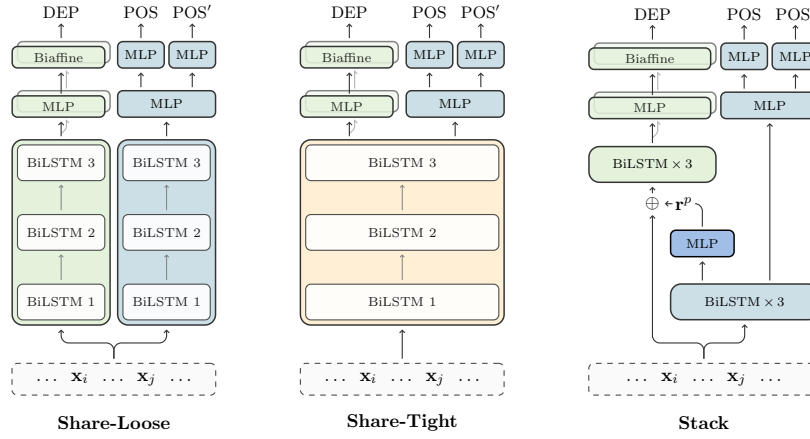
Fig. 2: The framework of three variants of the joint model.

Share-Loose. The tagging and parsing tasks use nearly separate networks, and only share the word and char embeddings. To incorporate heterogeneous POS tagging data, we add another scoring MLP at the top to compute scores of different heterogeneous POS tags. Under such architecture, the loosely connected tagging and parsing components can only influence each other in very limited manner.

Share-Tight. This is the most commonly used MTL framework, in which the tagging and parsing components share not only the embeddings, but also the BiLSTM encoder. Different decoders are then used for different tasks. In this tightly joint model, the tagging and parsing components can interact with and mutually help each other to a large extent. The shared parameters are trained to capture the commonalities of the two tasks.

Stack. The *Stack* takes BiLSTM hidden outputs of the tagger, denoted as $\mathbf{r}_i^p$, as the extra input of the parser. In this way, the error propagation problem can be better handled.

$$\mathbf{x}_i^{parse} = \mathbf{x}_i \oplus \mathbf{r}_i^p \tag{5}$$

The idea of the *Stack* joint method is mainly borrowed from Zhang and Weiss (2016) [26]. They propose the *stack-propagation* approach to avoid using explicit POS tags in dependency parsers. They employ the simple feed-forward networks for both tagging and parsing [2]. Without BiLSTM encoders, they use the hidden outputs of a single-layer MLP of the tagging component as extra inputs of the parsing component.

Training loss. During training, we directly add together all losses of different tasks, i.e., the parsing loss, the homogeneous POS tagging loss, and the heterogeneous POS tagging loss.

$$\mathcal{L} = \mathcal{L}_{\text{DEP}} + \mathcal{L}_{\text{POS}} + \mathcal{L}_{\text{POS}'} \tag{6}$$

4   Experiments

In this section, we conduct experiments and detailed analysis to make full investigation on the usefulness of POS tagging for dependency parsing.

4.1   Experimental Settings

Data. We conduct experiments on the English Penn Treebank (PTB), the Chinese dataset at the CoNLL-2009 shared task (CoNLL09) [7], and the larger-scale Chinese Penn Treebank 7 (CTB7). For PTB, we adopt the same settings such as data split and Stanford dependencies of Chen and Manning (2014) [2]. We follow the official settings for CoNLL09.

We use the Stanford Parser v3.0 to obtain Stanford dependencies for CTB7.[3] For Chinese, besides the homogeneous POS tags, we also incorporate the large-scale People Daily corpus of Peking University (PKU) as heterogeneous POS tagging data.

Evaluation metrics. We use POS tagging accuracy (TA), unlabeled attachment score (UAS), and labeled attachment scores (LAS) for dependency parsing. For UAS and LAS computation, We follow Dozat and Manning (2017) [4] and ignore all punctuation marks for PTB.

Hyper-parameters. We follow most hyper-parameter settings of Dozat and Manning (2017) [4] for all our models. For CHARLSTM word representations, we set the dimension of the character embeddings to 50, and the dimension of CHARLSTM outputs to 100. We train each model for at most 1,000 iterations, and stop training if the peak performance on the dev data does not increase in 100 (50 for models with BERT) consecutive iterations.

4.2   Results on the Dev Data

Results of the pipeline framework. Table 1 shows the influence of using homogeneous and heterogeneous POS tags in the pipeline framework. More results are also presented to understand the contributions of each of the four components in the input layer. The homogeneous tagging accuracy is 97.58, 96.59, 96.72, 97.85, on the dev data of PTB, CoNLL09, and CTB7, and PKU, respectively. We perform 5-fold jack-knifing to obtain the automatic homogeneous POS tags on the training data to avoid closed testing, and use a POS tagger trained on PKU data to produce heterogeneous POS tags for sentences of CoNLL09 and CTB7.

The results of using only one component clearly show that lexical information (i.e., $\mathbf{e}^w$ and $\mathbf{e}^c$) is most crucial for parsing, and only using POS tag embeddings leads to very large accuracy drop.

When using two components at the same time, using CHARLSTM word representations ($\mathbf{e}^c$) is slightly yet consistently better than using POS tag embeddings ($\mathbf{e}^p$), both substantially outperforming the model using only word embeddings ($\mathbf{e}^w$) by more than 0.5 on all three datasets.

---

[3] https://nlp.stanford.edu/software/stanford-dependencies.shtml

	PTB	CoNLL09	CTB7
$\mathbf{e}^p$	87.79	75.94	75.72
$\mathbf{e}^w$	93.42	85.30	84.43
$\mathbf{e}^c$	93.34	84.42	83.73
$\mathbf{e}^w \oplus \mathbf{e}^p$	93.92	85.94	85.12
$\mathbf{e}^w \oplus \mathbf{e}^c$	**93.97**	86.09	85.23
$\mathbf{e}^w \oplus \mathbf{e}^c \oplus \mathbf{e}^p$	93.88	**86.17**	**85.32**
$\mathbf{e}^w \oplus \mathbf{e}^c \oplus \mathbf{e}^p \oplus \mathbf{e}^{p'}$	-	86.01	85.23

Table 1: Parsing performance (LAS) on dev data under the pipeline framework.

		PTB	CoNLL09	CTB7
homo	*Share-Loose*	93.95	86.28	85.56
	Share-Tight	93.93	86.17	85.56
	Stack	**94.09**	**86.26**	**85.79**
hetero	*Share-Loose*	-	86.05	85.62
	Share-Tight	-	**86.25**	85.76
	Stack	-	86.16	**85.86**
homo + hetero	*Share-Loose*	-	86.30	85.57
	Share-Tight	-	86.62	85.86
	Stack	-	**86.69**	**85.88**

Table 2: Parsing performance (LAS) comparison on dev data for the three joint methods.

Moreover, using three components leads to slight improvement on both CoNLL09 and CTB7, but hurts performance on PTB. Further using heterogeneous tag embeddings slightly degrades the performance. 用了额外的词性信息没有提升性能

All those results indicate that under the pipeline framework, POS tags become unnecessary and can be well replaced by the CHARLSTM word representations. We believe the reasons are two-fold. First, CHARLSTM can effectively capture morphological inflections by looking at lemma/prefix/suffix, and thus plays a similar role as POS tags in terms of alleviating the data sparseness problem of words. Second, the error propagation issue makes predicted POS tags less reliable.

Results of the joint methods. Table 2 presents the results of the three joint tagging and parsing methods without or with heterogeneous POS tagging.

When using only homogeneous POS tagging, we find that the performance gaps between different joint methods are very small. The best joint methods outperform the basic model by 0.1, 0.3, and 0.6 respectively. A similar situation arises when using heterogeneous tagging only.

When using both homogeneous and heterogeneous tagging , aka (w/ hetero) setting, we can see that the overall performance is further improved by large margin. The best *Stack* method outperforms the basic model by 0.6 on CoNLL09 and 0.7 on CTB7, showing that heterogeneous labeled data can inject useful knowledge into the model.

Overall, we can see that the *Stack* method is more stable and superior compared with the other three methods, and is adopted for the following experiments and analysis.

4.3 Final Results on the Test Data

Table 3 shows the results on the test data. For the scenario of not using BERT, the pipeline method using homogeneous POS tags is slightly yet consistently inferior to the basic model. The *Stack* method using only homogeneous POS tags significantly outperforms the basic method by 0.2 ($p < 0.005$), 0.4 ($p < 0.0005$), and 0.5 ($p < 0.0005$) in LAS on the three datasets respectively. Utilizing heterogeneous POS tags on

	PTB			CoNLL09			CTB7		
	TA	UAS	LAS	TA	UAS	LAS	TA	UAS	LAS
w/o BERT Andor et al. (2016) [1]	97.44	94.61	92.79	-	84.72	80.85	-	-	-
Dozat and Manning (2017) [4]	97.3	95.74	94.08	-	88.90	85.38	-	-	-
Ji et al. (2019) [11]	97.3	95.97	94.31	-	-	-	-	-	-
Li et al. (2019) [17]	97.3	95.93	94.19	-	88.77	85.58	-	-	-
Basic ($\mathbf{e}^w \oplus \mathbf{e}^c$)	97.50	95.97	94.34	96.42	89.12	86.00	96.48	88.58	85.40
Pipeline ($\mathbf{e}^w \oplus \mathbf{e}^c \oplus \mathbf{e}^p$)	97.50	95.88	94.27	96.42	89.12	85.98	96.48	88.42	85.28
Stack	**97.91**	**96.13**	**94.53**	96.55	89.46	86.44	96.62	88.86	85.88
Stack w/ hetero	-	-	-	**96.66**	**89.85**	**86.85**	**96.72**	**89.26**	**86.27**
w/ BERT Li et al. (2019) [17]	-	96.67	95.03	-	92.24	89.29	-	-	-
Basic ($\mathbf{e}^w \oplus \mathbf{e}^c$)	97.42	96.85	95.14	97.29	92.21	89.42	97.22	91.66	88.75
Stack	**97.57**	**96.85**	**95.25**	97.36	92.44	89.68	97.32	91.67	88.84
Stack w/ hetero	-	-	-	**97.39**	**92.46**	**89.76**	**97.40**	**91.81**	**89.04**

Table 3: Final results on the test data. It is noteworthy that we produce our experiments with single run for each model on each dataset, since our preliminary experiments that we train *Stack* w/ hetero and Basic on CTB7 for four times show the variance of performances is small ($\sigma^2 < 0.01$).

Chinese further boosts parsing performance, leading to large overall improvements of 0.9 ($p < 0.0001$) on both datasets.

When using BERT, parsing accuracy of the basic method increases by very large margin   Overall, using both homogeneous and heterogeneous POS tagging, the *Stack* method significantly outperforms the basic method by 0.3 ($p < 0.005$) on both CoNLL09 and CTB7.

For POS tagging, the trend of performance change is similar. First, the joint method can also improve tagging accuracy, especially when with heterogeneous POS tagging. Using Bert can substantially improve TA on both Chinese datasets. However, it is surprising to see a slight decrease in TA when using BERT, which is possibly due to overfitting considering the TA is already very high on English.

We also list the results of recent previous works. We can see that our final joint models achieve competitive parsing accuracy on PTB and CoNLL09 w/ or w/o BERT.

4.4   Detailed Analysis

In the following, we conduct detailed analysis on the CoNLL09 test data, in order to understand or gain more insights on the interactions and mutual influence between POS tagging and dependency parsing. For the joint method, we adopt the *Stack* model with both homogeneous and heterogeneous POS tagging without using BERT, to jointly produce automatic POS tags and parse trees. For the pipeline method, we use the two basic tagging and parsing models separately to produce automatic results.

Correlation of performance changes between tagging and parsing. Overall, the joint method outperforms the pipeline method by 0.2 in TA, and 0.7/0.9 in UAS/LAS,
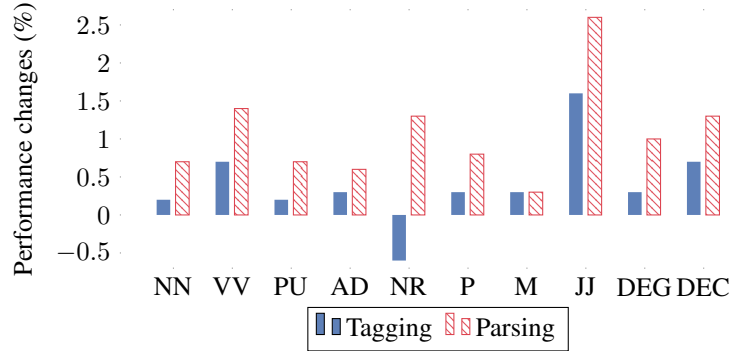
Fig. 3: Changes of tagging accuracy and parsing accuracy (LAS) on the CoNLL09 test set for words of different POS tags. Words/arcs are categorized according to their/their dependent's gold-standard POS tags.

as shown in Table 3. To gain more insights, we categorize all words according to their gold-standard POS tags and compare the accuracy changes for each set. Figure 3 shows the most frequent tags. We can see that there is clear positive correlation of absolute performance changes between tagging and parsing. For instance, as the most frequent tags NN and VV, their tagging accuracy increases by 0.2 and 0.7, and parsing accuracy increases by 0.7 and 1.4, respectively. The most notable exception is NR with opposite changes in tagging and parsing accuracy (-0.6 vs. +1.3), which can be explained from two aspects. First, we find that most of NR mistakes are due to the {NR, NN} ambiguous pair, which is syntax-insensitive and thus has very small impact on parsing decisions. Second, the *Stack* model may be more robust to tagging errors.

Overall, we conclude that tagging and parsing performance is highly correlated due to the close relationship between the two tasks.

Influence of tagging errors on parsing. In the *Stack* method, the hidden representations from the tagging encoder is fed into the parsing encoder as extra inputs. We would like to understand how tagging decisions influence parsing. Overall, UAS/LAS are 90.42/88.38 for words getting correct POS tags, whereas 73.40/42.59 for wrongly tagged words. We can observe dramatic drop of 17.0/45.8, indicating that POS tags has much larger influence on LAS than UAS.

Looking deeper into this issue, Table 4 shows the parsing accuracy for words of different POS tagging patterns. A tagging pattern X → Y represents the set of words whose correct tag is X and are tagged as Y.

We can see that higher parsing accuracy are usually achieved by correct tagging patterns X → X than wrong pattern X → NOT-X, except DEC → DEG in UAS.[4]

---

[4] DEG and DEC are two tags for the frequently used auxiliary word "的" (dē, translated as "of" or "that") in Chinese. "的" is tagged as DEG in phrase "土地/land 的面积/area (area of the land)", while as DEC in "他/he 提出/proposed 的方法/method (method that he proposed)".

	UAS	LAS			UAS	LAS
NN → NN	91.73	89.69	NR → NR		91.73	86.96
→ VV	67.25	44.98	→ NN		86.39	83.67
→ NR	90.43	86.96	JJ → JJ		95.40	94.33
→ JJ	91.96	20.54	→ NN		92.82	14.92
VV → VV	85.92	84.12	DEG → DEG		96.75	95.91
→ NN	65.60	40.07	→ DEC		92.06	26.56
→ VA	84.75	83.05	DEC → DEC		94.28	92.39
→ AD	55.32	25.53	→ DEG		96.88	22.22

Table 4: The impact of specific POS tagging error patterns on parsing.

The tagging ambiguites can be classified into three types. First, the syntax-sensitive ambiguous pairs such as {NN, VV} and {VV, AD} lead to large performance decrease in both UAS and LAS if wrongly tagged. Second, the syntax-insensitive ambiguous pairs such as {NN, NR} and {VV, VA} have very small influence on parsing accuracy. Finally, some ambiguous pairs only greatly influence LAS but have little effect on UAS, such as {NN, JJ} and {DEC, DEG}.

5   Related Works

Previous studies [16, 20] show POS tags are indispensable ingredients for composing different features in the traditional pre-DL dependency parsers [14, 27]. Meanwhile, Li et al. (2011) [18] show that the error propagation problem introduced by predicted POS tags degrades parsing accuracy by about 6 (UAS) on different Chinese datasets. Therefore, researchers propose to jointly model the POS tagging and dependency tasks in the graph-based [18] and transition-based [9] frameworks, leading to promising results. The key challenge is to define scoring functions on the joint results, and design effective search algorithms to determine optimal joint answers in the enlarged search space. Furthermore, joint models of word segmentation, POS tagging, and parsing are also proposed [10].

In the DL era, joint modeling of multiple related tasks becomes much easier under the MTL framework [3]. In fact, MTL has become an extensively used and powerful technique for many problems. The basic idea is sharing the encoder part while using separate decoders for different tasks. The major advantages of employing MTL are two-fold, i.e., 1) exploiting the correlation and mutual helpfulness among related tasks, and 2) making direct use of all (usually non-overlapping) labeled data of different tasks. The Share-Light and Share-Tight methods are both typical MTL frameworks, and the main difference lies in the amount of shared parameters. Actually, there are still many other variants due to the flexibility of MTL. For example, Straka (2018) [23] stacks task-specific private BiLSTMs over shared BiLSTMs for joint tagging and parsing. Based on the current results, we expect that such variants may achieve very similar performance.

The *Stack* method is similar to the *stack-propagation* method of Zhang and Weiss (2016) [26]. Their basic idea is to use the hidden outputs of the POS tagging components as extra inputs of the parsing components, forming a stacked structure. During

training, parsing loss is directly propagated into the full tagging component whereas tagging loss only indirectly influences the parsing components via their shared parts. Their pioneer work employ a simple feed-forward network for both tagging and parsing [2], and only achieves an LAS of 91.41 on PTB. Another inspiring work related with the *Stack* method is Hashimoto et al. (2017) [8], who propose to jointly train many tasks of different complexity in a very deep and cascaded network architecture, where higher levels are used for more complex tasks.

6   Conclusions

Unlike the findings in traditional pre-DL dependency parsing, recent studies indicate that POS tagging becomes much less important and can be replaced by CHARLSTM word representations in neural dependency parsers. However, there lacks a full and systematic investigation on this interesting issue, from both empirical and linguistic perspectives. In this paper, we try to investigate the role of POS tagging for neural dependency parsing in both pipeline and joint frameworks. We design and compare three typical joint methods based on the state-of-the-art biaffine parser. We try to accommodate both homogeneous and heterogeneous POS tagging, considering it is much cheaper to annotate POS tags than parse trees and there exist large-scale heterogeneous POS tag datasets for Chinese. Based on the experiments and analysis on three English and Chinese benchmark datasets, we can draw the following conclusions.

- For the pipeline method, both homogeneous and heterogeneous POS tags provide little help to the basic parser with both word embeddings and CHARLSTM, due to error propagation and the overlapping role in reducing data sparseness.
- The three joint methods investigated in this work perform better than the pipeline method. Among them, the *Stack* is more stable and superior compared with the other three, leading to significant improvement over the basic model on all datasets.
- POS tagging is still helpful for dependency parsing under the joint framework even if the parser is enhanced with BERT, especially when with heterogeneous POS tagging.
- Detailed analysis shows that POS tagging and dependency parsing are two closely correlated tasks. In particular, If the joint model fails to resolve syntax-sensitive POS tagging ambiguities, it usually makes wrong parsing decisions as well.

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 ACL (2016)
2. Chen, D., Manning, C.: A fast and accurate dependency parser using neural networks. In: Proceedings of EMNLP (2014)
3. Collobert, R., Weston, J.: A unified architecture for natural language processing: Deep neural networks with multitask learning. In: Proceedings of ICML (2008)
4. Dozat, T., Manning, C.D.: Deep biaffine attention for neural dependency parsing. In: Proceedings of ICLR (2017)

5. Dozat, T., Qi, P., Manning, C.D.: Stanford's graph-based neural dependency parser at the CoNLL 2017 shared task. In: Proceedings of CoNLL (2017)
6. Dyer, C., Ballesteros, M., Ling, W., Matthews, A., Smith, N.A.: Transition-based dependency parsing with stack long short-term memory. In: Proceedings of ACL (2015)
7. Hajič, J., Ciaramita, M., Johansson, R., Kawahara, D., Martí, M.A., Màrquez, L., Meyers, A., Nivre, J., Padó, S., Štěpánek, J., Straňák, P., Surdeanu, M., Xue, N., Zhang, Y.: The CoNLL-2009 shared task: Syntactic and semantic dependencies in multiple languages. In: Proceedings of CoNLL (2009)
8. Hashimoto, K., Xiong, C., Tsuruoka, Y., Socher, R.: A joint many-task model: Growing a neural network for multiple NLP tasks. In: Proceedings of EMNLP (2017)
9. Hatori, J., Matsuzaki, T., Miyao, Y., Tsujii, J.: Incremental joint POS tagging and dependency parsing in Chinese. In: Proceedings of IJCNLP (2011)
10. Hatori, J., Matsuzaki, T., Miyao, Y., Tsujii, J.: Incremental joint approach to word segmentation, POS tagging, and dependency parsing in Chinese. In: Proceedings of ACL (2012)
11. Ji, T., Wu, Y., Lan, M.: Graph-based dependency parsing with graph neural networks. In: Proceedings of ACL (2019)
12. Kiperwasser, E., Goldberg, Y.: Simple and accurate dependency parsing using bidirectional LSTM feature representations. Transactions of ACL (2016)
13. Kitaev, N., Klein, D.: Constituency parsing with a self-attentive encoder. In: Proceedings of ACL (2018)
14. Koo, T., Collins, M.: Efficient third-order dependency parsers. In: ACL (2010)
15. Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K., Dyer, C.: Neural architectures for named entity recognition. In: Proceedings of NAACL (2016)
16. Lei, T., Xin, Y., Zhang, Y., Barzilay, R., Jaakkola, T.: Low-rank tensors for scoring dependency structures. In: Proceedings of ACL (2014)
17. Li, Y., Li, Z., Zhang, M., Wang, R., Li, S., Si, L.: Self-attentive biaffine dependency parsing. In: Proceedings of IJCAI (2019)
18. Li, Z., Zhang, M., Che, W., Liu, T., Chen, W., Li, H.: Joint models for Chinese POS tagging and dependency parsing. In: Proceedings of EMNLP (2011)
19. Maas, A.L., Hannun, A.Y., Ng, A.Y.: Rectifier nonlinearities improve neural network acoustic models. In: Proceedings of ICML. p. 3 (2013)
20. McDonald, R., Petrov, S., Hall, K.: Multi-source transfer of delexicalized dependency parsers. In: Proceedings of EMNLP (2011)
21. Nivre, J., de Marneffe, M.C., Ginter, F., Goldberg, Y., Hajic, J., Manning, C.D., McDonald, R., Petrov, S., Pyysalo, S., Silveira, N., Tsarfaty, R., Zeman, D.: Universal dependencies v1: A multilingual treebank collection. In: Proceedings of LREC (2016)
22. Roller, S., Kiela, D., Nickel, M.: Hearst patterns revisited: Automatic hypernym detection from large text corpora. In: Proceedings of ACL (2018)
23. Straka, M.: UDPipe 2.0 prototype at CoNLL 2018 UD shared task. In: Proceedings of CoNLL (2018)
24. Zeman, D., Hajič, J., Popel, M., Potthast, M., Straka, M., Ginter, F., Nivre, J., Petrov, S.: CoNLL 2018 shared task: Multilingual parsing from raw text to universal dependencies. In: Proceedings of CoNLL (2018)
25. Zhang, M., Li, Z., Fu, G., Zhang, M.: Syntax-enhanced neural machine translation with syntax-aware word representations. In: Proceedings of NAACL (2019)
26. Zhang, Y., Weiss, D.: Stack-propagation: Improved representation learning for syntax. In: Proceedings of ACL (2016)
27. Zhang, Y., Nivre, J.: Transition-based dependency parsing with rich non-local features. In: Proceedings of ACL (2011)
28. Zhou, H., Zhang, Y., Huang, S., Chen, J.: A neural probabilistic structured-prediction model for transition-based dependency parsing. In: Proceedings of ACL (2015)