Joint POS Tagging and Dependence Parsing With Transition-Based Neural Networks

Liner Yang, Meishan Zhang, Yang Liu, Maosong Sun, Nan Yu, and Guohong Fu

***Abstract*—While part-of-speech (POS) tagging and dependency parsing are observed to be closely related, existing work on joint modeling with manually crafted feature templates suffers from the feature sparsity and incompleteness problems. In this paper, we propose an approach to joint POS tagging and dependency parsing using transition-based neural networks. Three neural network based classifiers are designed to resolve shift/reduce, tagging, and labeling conflicts. Experiments show that our approach significantly outperforms previous methods for joint POS tagging and dependency parsing across a variety of natural languages.**

***Index Terms*—Dependency parsing, joint model, neural networks, part-of-speech tagging.**

I. Introduction

PART-OF-SPEECH (POS) tagging [1]–[4] and dependency parsing [5]–[9] are two fundamental tasks for understanding natural languages. While POS tagging aims to assign parts of speech to words in a text to indicate their word categories, the goal of dependency parsing is to analyze the syntactic structure of sentences by establishing relationships between words.

It is widely accepted that POS tagging and dependency parsing are closely related. On one hand, POS tagging often requires long-distance syntactic information for resolving tagging ambiguity [10]. Hatori *et al.* [11] indicate that the disambiguation between POS tags "DEG" (a genitive marker) and "DEC" (a complementizer) for a Chinese word *de* often depends on global context. On the other hand, as a preprocessing step, POS tagging directly influences the accuracy of dependency parsing significantly. For example, determining the head word of a two-word phrase "closed door" directly depends on the POS tag of "closed" (adjective or verb in past tense). Li *et al.* [12] report that dependency accuracy drops by around 6% on Chinese when automatic POS tagging results instead of ground-truth tags are used.

Therefore, joint POS tagging and dependency parsing has attracted intensive attention in the NLP community. Previous work has focused on jointly modeling POS tagging and dependency parsing using linear models that combine both tagging and parsing features [11]–[15]. Allowing lexicality and syntax to interact in a unified framework, joint POS tagging and dependency parsing improves both tagging and parsing performance over independent modeling significantly [11]–[13].

However, existing work on joint POS tagging and dependency parsing suffers from the feature sparsity and incompleteness problems. Chen and Manning [7] indicate that lexicalized indicator features indispensable for discriminative dependency parsing are usually highly sparse. The situation in joint POS tagging and dependency parsing is much more severe because tagging and parsing features are concatenated in joint models [12]. Moreover, due to the complexity of tagging and parsing natural languages, it is hard for manually-designed features to cover all regularities. As a result, the incompleteness of feature design is considered as an unavoidable issue in conventional discriminative models [7].

In this paper, we propose an approach to joint POS tagging and dependency parsing with neural networks by extending from a transition-based dependency parsing model. Three neural network based classifiers are designed to resolve the conflicts of transition actions, respectively for shift/reduce (dependency parse tree skeletons), tagging (POS tagging), and labeling (dependency label) disambiguations. Experiments show that our approach significantly outperforms previous methods for joint POS tagging and dependency parsing on three treebanks across eight natural languages.

II. Approach

A. Problem Statement

As shown in Fig. 1, given an English sentence "He won the game", the corresponding tag sequence is "PRP VBD DT NN". These tags indicate the part of speech of each word: "He" is a personal pronoun, "won" is a verb in past tense, "the" is a determiner, and "game" is a noun.

Fig. 1 also shows a dependency tree, which is a collection of dependency arcs. The leftmost arc between the first two words indicates that "won" is a head word, "He" is a modifier, and the syntactic label "*nsubj*" suggests that "He" is a nominal subject.

Manuscript received July 16, 2017; revised November 20, 2017; accepted December 11, 2017. Date of publication December 29, 2017; date of current version May 8, 2018. This work was supported in part by the National Basic Research 973 Program of China under Grant 2014CB340501 and in part by the National Natural Science Foundation of China under Grants 61602160 and 61672211. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Ani Nenkova. *(Corresponding author: Yang Liu.)*

L. Yang, Y. Liu, and M. Sun are with the Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China (e-mail: lineryang@gmail.com; liuyang2011@tsinghua.edu.cn; sms@tsinghua.edu.cn).

M. Zhang, N. Yu, and G. Fu are with the School of Computer Science and Technology, Heilongjiang University, Harbin 150080, China (e-mail: mason.zms@gmail.com; yunan.hlju@gmail.com; ghfu@hlju.edu.cn).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TASLP.2017.2788181

TABLE I
TRANSITIONS FOR JOINT POS TAGGING AND DEPENDENCY PARSING

Transition	Definition	Condition
SHIFT	$\langle S, x_n \vert B, T, D\rangle \Rightarrow \langle S \vert x_n, B, T, D\rangle$	$\vert B\vert > 0 \wedge \vert T\vert = N - \vert B\vert \wedge D_{-1}.l \neq \perp$
LEFT	$\langle S \vert x_m \vert x_h, B, T, D\rangle \Rightarrow \langle S \vert x_h, B, T, D \cup \{\langle h, m, \perp\rangle\}\rangle$	$\vert S\vert > 1 \wedge \vert T\vert = N - \vert B\vert \wedge D_{-1}.l \neq \perp$
RIGHT	$\langle S \vert x_h \vert x_m, B, T, D\rangle \Rightarrow \langle S \vert x_h, B, T, D \cup \{\langle h, m, \perp\rangle\}\rangle$	$\vert S\vert > 1 \wedge \vert T\vert = N - \vert B\vert \wedge D_{-1}.l \neq \perp$
TAG_t	$\langle S, B, T, D\rangle \Rightarrow \langle S, B, T \cup \{t\}, D\rangle$	$\vert T\vert = N - \vert B\vert - 1$
LABEL_l	$\langle S \vert x_h, B, T, D \cup \{\langle h, m, \perp\rangle \Rightarrow \langle S \vert x_h, B, T, D \cup \{\langle h, m, l\rangle\}\rangle$	$D_{-1}.l = \perp$

We use a quadruple $\langle S, B, T, D\rangle$ to denote a configuration, which consists of a stack S, a buffer B, a tag sequence T, and a dependency arc set D. We define five categories of actions SHIFT (moving a word from the buffer to the stack), LEFT (generating a right-headed dependency arc), RIGHT (generating a left-headed dependency arc), TAG_t (tagging the last word moved into stack as t), and LABEL_l (labeling the last generated arc as l) for the transitions between configurations. We use $\perp$ to denote an undefined syntactic label, and $D_{-1}.l$ to denote the syntactic label of the last generated dependency arc. N is the length of the input sentence.

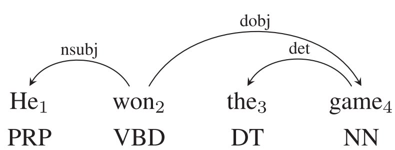


Fig. 1. Part-of-speech tagging and dependency parsing. Given an English sentence "He won the game", our goal is to predict its corresponding part-of-speech tag sequence "PRP VBD DT NN" and dependency tree $\{\langle 2, 1, \text{nsubj}\rangle, \langle 4, 3, \text{det}\rangle, \langle 2, 4, \text{dobj}\rangle\}$.

More formally, given a natural language sentence $\mathbf{x} = x_1, \ldots, x_N$, we denote its corresponding POS tag sequence as $\mathbf{t} = t_1, \ldots, t_N$, where $t \in \mathcal{T}$ is a POS tag and $\mathcal{T}$ is a set of all possible tags. A dependency tree is denoted by $\mathbf{d} = \{\langle h, m, l\rangle \vert 0 < h \leq N, 0 < m \leq N, l \in \mathcal{L}\}$. We use $\langle h, m, l\rangle$ to represent a dependency arc, where x_h is a head word, x_m is a modifier, and l is syntactic label. We use $\mathcal{L}$ to denote the set of all possible syntactic labels. The dependency tree in Fig. 1 consists of three arcs: $\langle 2, 1, \textit{nsubj}\rangle$, $\langle 2, 4, \textit{dobj}\rangle$, and $\langle 4, 3, \textit{det}\rangle$.

Therefore, the goal of our work is to generate a tag sequence $\mathbf{t}$ and a dependency tree $\mathbf{d}$ for a given sentence $\mathbf{x}$.

B. Transition System

In this work, we leverage a transition-based approach [16] to joint POS tagging and dependency parsing, which uses classifiers to predict individual actions of shift-reduce algorithms.

We define a *configuration* as a quadruple $c = \langle S, B, T, D\rangle$, where

1) S: a *stack* that is a disjoint sublist of words,
2) B: a *buffer* that is a sublist of words to be processed,
3) T: a *tag sequence* that stores the result of POS tagging,
4) D: a *dependency arc set* that stores the result of dependency parsing.

As shown in Table I, we define five categories of actions for the transition between configurations:[1]

1) *Shift:* move the leftmost word from the buffer B to the stack S;
2) *Left:* combine the top two items on the stack, x_m and x_h, replace them with x_h as the head, and add an unlabeled dependency arc $\langle h, m, \perp\rangle$ to D;
3) *Right:* combine the top two items on the stack, x_h and x_m, replace them with x_h as the head, and add an unlabeled dependency arc $\langle h, m, \perp\rangle$ to D;
4) Tag_t*:* assign a POS tag t to the last added word if the previous action is SHIFT (i.e., $\vert T\vert = N - \vert B\vert - 1$);
5) $Label_l$*:* assign a syntactic label l to the last generated dependency arc if the previous action is LEFT or RIGHT (i.e., $D_{-1}.l = \perp$).

where N is the length of the input sentence. We follow Bohnet and Nivre [13] to use $\perp$ to denote an undefined syntactic label. $D_{-1}.l$ represents the syntactic label of the last added dependency arc. Note that the first three actions are used to determine the skeletons of dependency trees, which can be applied on condition that all words removed from the buffer are tagged (i.e., $\vert T\vert = N - \vert B\vert$), and all generated dependency arcs are labeled (i.e., $D_{-1}.l \neq \perp$).

Compared with the transition system of Bohnet and Nivre [13], which integrates POS/dependency labels into the SHIFT/LEFT/RIGHT actions, our transition system supports various pre-training techniques, which are important to improve the performances of our model. For example, we can pre-train the POS tagging related parameters on a separate POS tagging neural model.

Table II demonstrates the process of joint tagging and dependency parsing for the example in Fig. 1. The initial configuration at step 0 is $c_0 = \langle \emptyset, \{x_1, x_2, x_3, x_4\}, \emptyset, \emptyset\rangle$. In step 1, the action SHIFT moves the leftmost word x_1 (i.e., "He") from the buffer B to the stack S. Then, the action TAG_{PRP} assigns a POS tag "PRP" to the last shifted word "He". In this way, the configuration keeps changing by applying various actions until the terminal configuration (i.e., the stack contains only one item, the buffer is empty, all words are tagged, and all arcs are labeled) is generated.

C. Modeling

Given a sentence $\mathbf{x}$ with N words, tag sequence $\mathbf{t}$ and dependency tree $\mathbf{d}$ corresponds to a unique sequence of action-configuration pairs $\{\langle c_i, a_i\rangle\}_{i=1}^{4N-2}$, as shown in Table II.[2] Note

[1] While it is possible to integrate two actions into one action (e.g., combining SHIFT and TAG_t into SHIFT_t) [13], we find that separating tag and label actions (i.e., TAG_t and LABEL_l) from structural actions (i.e., SHIFT, LEFT, and RIGHT) leads to significant improvements over using combined actions.

[2] We follow Chen and Manning [7] to map a parse to a unique sequence of action-configuration pairs by using the "shortest stack" strategy.

TABLE II
THE PROCESS OF JOINT POS TAGGING AND DEPENDENCY PARSING FOR THE EXAMPLE IN FIG. 1

Step	Transition	Stack (S)	Buffer (B)	Tags (T)	Dependencies (D)
0			He_1 won_2 the_3 game_4		
1	SHIFT	He_1	won_2 the_3 game_4		
2	TAG_{PRP}	He_1	won_2 the_3 game_4	PRP	
3	SHIFT	He_1 won_2	the_3 game_4	PRP	
4	TAG_{VBD}	He_1 won_2	the_3 game_4	PRP VBD	
5	LEFT	won_2	the_3 game_4	PRP VBD	$\langle 2, 1, \perp\rangle$
6	$\text{LABEL}_{\text{nsubj}}$	won_2	the_3 game_4	PRP VBD	$\langle 2, 1, \text{nsubj}\rangle$
7	SHIFT	won_2 the_3	game_4	PRP VBD	$\langle 2, 1, \text{nsubj}\rangle$
8	TAG_{DT}	won_2 the_3	game_4	PRP VBD DT	$\langle 2, 1, \text{nsubj}\rangle$
9	SHIFT	won_2 the_3 game_4		PRP VBD DT	$\langle 2, 1, \text{nsubj}\rangle$
10	TAG_{NN}	won_2 the_3 game_4		PRP VBD DT NN	$\langle 2, 1, \text{nsubj}\rangle$
11	LEFT	won_2 game_4		PRP VBD DT NN	$\langle 2, 1, \text{nsubj}\rangle$ $\langle 4, 3, \perp\rangle$
12	$\text{LABEL}_{\text{det}}$	won_2 game_4		PRP VBD DT NN	$\langle 2, 1, \text{nsubj}\rangle$ $\langle 4, 3, \text{det}\rangle$
13	RIGHT	won_2		PRP VBD DT NN	$\langle 2, 1, \text{nsubj}\rangle$ $\langle 4, 3, \text{det}\rangle$ $\langle 2, 4, \perp\rangle$
14	$\text{LABEL}_{\text{dobj}}$	won_2		PRP VBD DT NN	$\langle 2, 1, \text{nsubj}\rangle$ $\langle 4, 3, \text{det}\rangle$ $\langle 2, 4, \text{dobj}\rangle$

that the number of SHIFT actions is N, LEFT or RIGHT is $N-1$, TAG_t is N, and LABEL_l is $N-1$, where SHIFT and TAG_t have the same number as words, LEFT/RIGHT and LABEL have the same number as dependency arcs.

As a result, the probabilistic model for transition-based joint POS tagging and dependency parsing is defined as

$$P(\mathbf{t}, \mathbf{d}|\mathbf{x}; \boldsymbol{\theta}) = \prod_{i=1}^{4N-2} P(a_i|c_{i-1}; \boldsymbol{\theta}) \times P(c_i|c_{i-1}, a_i) \quad (1)$$

Note that $P(c_i|c_{i-1}, a_i) = 1$ if and only if a_i is a legal transition between c_{i-1} and c_i. Otherwise, $P(c_i|c_{i-1}, a_i) = 0$. Therefore, we only need to focus on the action probability conditioned on the previous configuration.

In our transition system, there are three types of conflicts:

1) *Tag conflict:* among all possible POS tags $\{\text{TAG}_t | t \in \mathcal{T}\}$,
2) *Shift/reduce conflict:* between SHIFT, LEFT, and RIGHT. For example, at step 5 in Table II, both SHIFT and LEFT can be applied,
3) *Label conflict:* among all possible syntactic labels $\{\text{LABEL}_l | l \in \mathcal{L}\}$.

To resolve these conflicts, we develop three corresponding neural network based classifiers. Note that the separation of structural actions from tagging and labeling actions results in three small classifiers with fewer classes (i.e., $|\mathcal{T}|$ classes for the tag classifier, 3 for the shift/reduce classifier, and $|\mathcal{L}|$ for the label classifier) rather than one big classifier with much more classes (i.e., $|\mathcal{T}| + 2|\mathcal{L}|$).

1) Basic Features: We use $\boldsymbol{x}_n$ to denote the vector representation of the n-th word x_n. In our experiments, we follow Kiperwasser and Goldberg [9] to learn $\boldsymbol{x}_n$ using bidirectional LSTM whose inputs are concatenations of randomly initialized word embeddings with additional pre-trained embeddings as well as character-based representations [3], [17]. We use $\boldsymbol{t}_n$ to denote the vector representation of the n-th POS tag t_n, which can be learned using a unidirectional LSTM based on randomly initialized tag embeddings. Note that the bidirectional LSTM feature representations for words are computed before joint POS tagging and dependency parsing while the unidirectional LSTM feature representations for tags are calculated during the search on the fly.

2) Tag Classification: Resolving the tag conflict is a $|\mathcal{T}|$-class classification problem. Instead of using conventional feature templates that are highly sparse and inevitably incomplete, we leverage a neural network based classifier. To determine the POS tag of the last word added to the stack, which is represented as x_{S_0}, the input layer consists of the following representations:

1) $\boldsymbol{x}_{S_1}$*:* the word representation of the second item in the stack,
2) $\boldsymbol{t}_{S_1}$*:* the tag representation of the second item in the stack,
3) $\boldsymbol{x}_{B_{-2}}$*:* the word representation of the second last item removed from the buffer,
4) $\boldsymbol{t}_{B_{-2}}$*:* the tag representation of the second last item removed from the buffer,
5) $\boldsymbol{x}_{S_0}$*:* the word representation of the first item in the stack,
6) $\boldsymbol{x}_{B_0}$*:* the word representation of the first item in the buffer.

where, $\boldsymbol{x}_{B_{-2}}$, $\boldsymbol{x}_{S_0}$, $\boldsymbol{x}_{B_0}$ are window-based features that have been widely adopted in previous work [4] and $\boldsymbol{t}_{B_{-2}}$ models the previous tag which has been widely used implicitly by markov assumption in CRF models. Note that $x_{B_{-2}}$, x_{S_0}, x_{B_0} are sequential words and x_{S_1} is not necessarily identical to $x_{B_{-2}}$ due to the RIGHT action.

We expect that these representations can provide useful contextual information for resolving the tagging ambiguity. Note that the tagging classifier is capable of exploiting syntactic information encoded in $\boldsymbol{x}_{S_1}$ and $\boldsymbol{t}_{S_1}$. The two features are not the direct output of dependency trees, and thus we use the parsing features for tagging in an implicit manner.

As shown in Fig. 2(a), the hidden layer is calculated as

$$\mathbf{h}_{\text{tag}}^{S_0} = \mathbf{W}_{\text{tag}}^{(1)} [\boldsymbol{x}_{S_1}; \boldsymbol{t}_{S_1}; \boldsymbol{x}_{B_{-2}}; \boldsymbol{t}_{B_{-2}}; \boldsymbol{x}_{S_0}; \boldsymbol{x}_{B_0}] \quad (2)$$

Then, the probability for tagging x_{S_0} as t is computed at the softmax layer:

$$P_{\text{tag}}(a|c; \boldsymbol{\theta}) = \text{softmax}\Big(\mathbf{W}_{\text{tag}}^{(2)} \mathbf{h}_{\text{tag}}^{S_0}\Big) \quad (3)$$

where $a \in \{\text{TAG}_t | t \in \mathcal{T}\}$.

3) Shift/Reduce Classification: Resolving the shift/reduce conflict is a 3-class classification problem. As shown in Fig. 2(b),

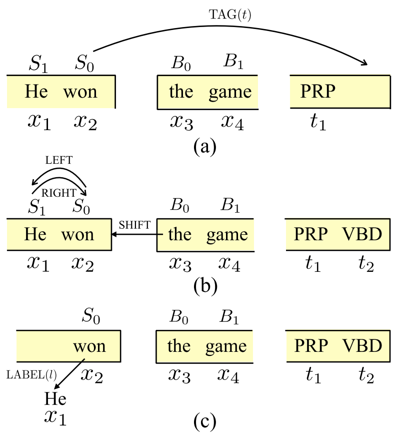


Fig. 2. Examples to illustrate the three classifiers. (a) Tag classification. (b) Shift/reduce classification. (c) Label classification.

we also use a neural classifier, in which the hidden layer is given by:[3]

$$\mathbf{h}_{\text{parse}} = \mathbf{W}^{(1)}_{\text{parse}}[\boldsymbol{x}_{S_2};\boldsymbol{t}_{S_2};\boldsymbol{x}_{S_1};\boldsymbol{t}_{S_1};\boldsymbol{x}_{S_0};\boldsymbol{t}_{S_0};\boldsymbol{x}_{B_0}] \tag{4}$$

where S_2 denotes the third item in the stack. Note that the shift/reduce classifier is capable of exploiting lexical information encoded in $\boldsymbol{t}_{S_2}$, $\boldsymbol{t}_{S_1}$, and $\boldsymbol{t}_{S_0}$.

Therefore, the shift/reduce classification probability is computed as

$$P_{\text{parse}}(a|c;\boldsymbol{\theta}) = \text{softmax}\Big(\mathbf{W}^{(2)}_{\text{parse}}\mathbf{h}_{\text{parse}}\Big) \tag{5}$$

where $a \in \{\text{SHIFT}, \text{LEFT}, \text{RIGHT}\}$.

4) Label Classification: Resolving the label conflict is a $|\mathcal{L}|$-class classification problem. As shown in Fig. 2(c), the corresponding neural classifier takes the word and tag representations of the first two items in the stack as input:

$$\mathbf{h}_{\text{label}} = \mathbf{W}^{(1)}_{\text{label}}[\boldsymbol{x}_{S_1};\boldsymbol{t}_{S_1};\boldsymbol{x}_{S_0};\boldsymbol{t}_{S_0}] \tag{6}$$

Clearly, labeling a dependency arc also depends on tag representations $\boldsymbol{t}_{S_1}$ and $\boldsymbol{t}_{S_0}$.

The label classification probability is computed as

$$P_{\text{label}}(a|c;\boldsymbol{\theta}) = \text{softmax}\Big(\mathbf{W}^{(2)}_{\text{label}}\mathbf{h}_{\text{label}}\Big) \tag{7}$$

where $a \in \{\text{LABEL}_l | l \in \mathcal{L}\}$.

D. Training and Parsing

Given a set of training examples $\{\langle\mathbf{x}^{(k)}, \mathbf{t}^{(k)}, \mathbf{d}^{(k)}\rangle\}_{k=1}^{K}$, the training objective is to minimize the cross-entropy loss plus a ℓ_2-regularization term:

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\text{argmin}}\left\{-\sum_{k=1}^{K}\log P(\mathbf{t}^{(k)}, \mathbf{d}^{(k)}|\mathbf{x}^{(k)};\boldsymbol{\theta}) + \frac{\lambda}{2}||\boldsymbol{\theta}||^2\right\} \tag{8}$$

In parsing, we follow Chen and Manning [7] to perform greedy decoding. The most probable tag sequence and dependency tree corresponds to a sequence of action-configuration pairs with the highest probability: $\{\langle\hat{c}_i, \hat{a}_i\rangle\}_{i=1}^{4N-2}$, where

$$\hat{a}_i = \underset{a}{\text{argmax}}\, P(a|\hat{c}_{i-1};\hat{\boldsymbol{\theta}}) \tag{9}$$

and $\hat{c}_i$ is obtained by applying $\hat{a}_i$ to $\hat{c}_{i-1}$.

III. Experiments

A. Setup

1) Datasets and Evaluation: We evaluate our approach on three datasets: the English Penn Treebank (PTB)[4] with annotated phrase-structure trees of English, the Chinese Penn Treebank (CTB) version 5.1[5] with annotated phrase-structure trees of Chinese, and the Universal Dependencies (UD) version 1.2[6] with annotated dependency trees across a number of natural languages.

We use the standard splitting method to divide the PTB dataset into training, development and test sections, and convert the phrase-structure trees into dependency trees by the Stanford dependency converter v3.3.0 [22]. For the CTB 5.1 dataset, we follow previous work [11], [13] to split the dataset into training, development and test sections, and use the Penn2Malt tool with the head-finding rule of [23] to convert the phrase-structure trees into dependencies. For the UD dataset, we follow Ammar *et al.* [24], using the same subset of seven languages including German (de), English (en), Spanish (es), French (fr), Italian (it), Portuguese (pt) and Swedish (sv) and using the same data splitting method.

For POS tagging, we use the standard tagging accuracy (POS) based on words as the major evaluation metric. For dependency parsing, we use two metrics, namely unlabeled attachment score (UAS) and labeled attachment score (LAS), where UAS denotes the ratio of the correctly-headed words with respect to the total number of words, which considers only the head of a word, and LAS takes into account the dependency label as well, and is emlpoyed as the major metric to evaluate dependency parsing.

2) Hyper-Parameters and Training Details: We tune all hyper-parameters in our models according the development results. Concretely, the dimension sizes of word, tag and character embeddings are 150, 50 and 50, respectively. We use the same pre-trained word embeddings for PTB and CTB 5.1 as Dyer *et al.* [8],[7] and do not use any pre-trained embeddings for UD, and the dimension size of the hidden states in neural classifiers is 300.

We exploit the Adam optimizer [25] to update model parameters during training, setting the hyper-parameters β_1 and β_2 both to 0.9. Gradient clipping [26] by a max norm 5.0 is used to avoid gradient exploding. To avoid overfitting, we use ℓ_2-regularization by a parameter 10^{-8} as well as the dropout

[3]Although it is possible to use hidden states in the tag classifier (e.g., $\mathbf{h}^{S_0}_{\text{tag}}$) to replace tag representations $\boldsymbol{t}_{S_0}$ as suggested by Zhang and Weiss [18], we find that it results in degenerate tagging and parsing results as compared with Eq. (4).

[4]https://catalog.ldc.upenn.edu/LDC99T42

[5]https://catalog.ldc.upenn.edu/LDC2005T01

[6]http://universaldependencies.org

[7]We thank the authors very much for sharing their data with us.

TABLE III
FINAL RESULTS ON THE DATASETS OF PTB AND CTB 5.1, WHERE THE TAGGING ACCURACY BEING 100% DENOTES GOLD-STANDARD POS TAGS ARE EMPLOYED

Method	PTB			CTB 5.1		
	POS	UAS	LAS	POS	UAS	LAS
Joint models						
Hatori *et al.* [11]	–	–	–	93.94	81.33	–
Bohnet and Nivre [13]	97.42	93.67	92.68	93.24	81.42	77.91
Zhang and Weiss [18]	–	93.43	91.41	–	–	–
this work (Joint)	**97.54**	**94.18**	**92.26**	**95.58**	**83.99**	**81.39**
Pipeline models						
Dyer *et al.* [8]	–	93.10	90.90	100	87.20	85.70
Kiperwasser and Goldberg [9]	–	93.90	91.90	100	87.60	86.10
Andor *et al.* [19]	–	94.61	92.79	–	–	–
Chen *et al.* [20]	–	94.10	91.49	100	88.10	85.70
Dozen and Manning [21]	–	95.74	94.08	–	89.30	88.23
this work (auto POS)	97.45	93.74	91.32	95.06	82.68	79.93
this work (gold POS)	100	94.73	93.53	100	88.75	87.53

We include the results of state-of-the-art previous transition-based parsers as well. In particular, Andor *et al.* (2016) use beam search in decoding and Bohnet and Nivre (2012) use a different method to produce dependency trees.

technique [27] with a drop rate of 0.25. Since the arc-standard algorithm can only handle the projective trees, we apply a projectivization step to the training sets of the UD dataset.

B. Main Results

Table III shows the final results of our models on PTB and CTB 5.1. We include the pipeline performances as well. Our joint model brings significant improvements on both POS tagging (POS) and dependency parsing (LAS) compared with the pipeline model (the p-value is below 10^{-5} using pairwise t-test). In addition, we compare our joint model with the baseline parsing model using gold-standard POS tags, which can be treated as the oracle performances of our joint model. Although the joint model gives improved performances over the pipeline model, it still has large spaces to reach the oracle performances, which demonstrates the effect of POS tags in dependency parsing.

We compare our model with previous work as well. On the one hand, we compare our joint model with previous joint models. As shown in Table III, our neural joint model shows the highest results for both PTB and CTB 5.1, obtaining much higher performances in dependency parsing, which demonstrates the effect of the neural features. On the other hand, we compare our baseline model with state-of-the-art transition-based dependency parsing models.

Typically, the PTB results are reported by using auto POS tags and the CTB 5.1 results are reported by using gold-standard POS tags, respectively. Our baseline model produces strong enough results for both PTB and CTB 5.1. In particular, Dozen and Manning [21] is a graph-based neural model for dependency parsing, which has achieved the top performance on the parsing task. We leave it as a future work to investigate the joint models under this framework.

TABLE IV
FINAL DEPENDENCY PARSING RESULTS (LAS) ON THE UD DATASET

Method	de	en	es	fr	it	pt	sv	AVG
Ballesteros *et al.* [17]	73.0	77.9	77.8	78.0	84.2	80.4	74.5	78.0
Zhang and Weiss [18]	74.2	80.7	80.7	80.0	85.8	80.4	77.5	79.9
this work (Pipeline)	74.6	80.6	80.6	78.9	84.9	81.6	77.6	79.8
this work (Joint)	**77.1**	**82.5**	**82.5**	**81.2**	**87.0**	**83.1**	**80.4**	**82.0**

Table IV shows the final results on the UD dataset. Joint models also achieves significantly better results in comparison with the pipeline models (p-value below 10^{-5}), which is similar to our finding on CTB 5.1. Besides, our joint model achieves the best-reported results among the transition-based models, even by using a greedy manner for decoding, which can be attributed to the effective exploration of the interaction between the tagging and parsing in our joint model, while no previous work has studied it under the neural setting to our knowledge. The work of Zhang and Weiss [18] resembles our work most, which improve a feed-forward dependency parser by using POS tags in a pipeline way by stack-propagation. While our joint model benefits from the use of LSTM, and in addition, we find that directly using the resulting tags rather than the penultimate hidden representations of a tag classifier leads to better results.

C. Discussion

To investigate the effect of POS tagging on dependency parsing, we conduct analysis on the CTB 5.1 dataset to illustrate the effect of the joint model. Here we examine in detail to see the benefits from the interaction between tagging and parsing in our joint model. First, we can remove the tag representations from parsing in Eq. (4) and Eq. (6):

$$\tilde{\mathbf{h}}_{\text{parse}} = \tilde{\mathbf{W}}^{(1)}_{\text{parse}}[\boldsymbol{x}_{S_2}; \boldsymbol{x}_{S_1}; \boldsymbol{x}_{S_0}; \boldsymbol{x}_{B_0}] \tag{10}$$

$$\tilde{\mathbf{h}}_{\text{label}} = \tilde{\mathbf{W}}^{(1)}_{\text{label}}[\boldsymbol{x}_{S_1}; \boldsymbol{x}_{S_0}] \tag{11}$$

Similarly, we can also remove the syntactic information from tagging in Eq. (2) to investigate the effect of dependency parsing on POS tagging:

$$\tilde{\mathbf{h}}^{S_0}_{\text{tag}} = \tilde{\mathbf{W}}^{(1)}_{\text{tag}}[\boldsymbol{x}_{B_{-2}}; \boldsymbol{t}_{B_{-2}}; \boldsymbol{x}_{S_0}; \boldsymbol{x}_{B_0}] \tag{12}$$

Table V gives the tagging and parsing results on the CTB 5.1 development set. We observe that disabling the interactions between tagging and parsing significantly deteriorates both tagging and parsing quality.

An interesting finding is that providing lexical information to parsing ("tag → parse") leads to more benefits than providing syntactic information to tagging ("tag ← parse"). This is because tagging ambiguity is mostly local while dependency parsing heavily depends on POS tags to predict syntactic structures.

Note that enabling "tag → parse" only also improves the tagging accuracy itself. One possible reason is that tagging and parsing is still connected via the sharing of word embeddings and bidirectional LSTM hidden states although the connection at hidden layer in classifiers is explicitly disabled.

TABLE V
INTERACTION BETWEEN POS TAGGING AND DEPENDENCY PARSING

Interaction		POS	UAS	LAS
tag → parse	tag ← parse			
×	×	95.19	83.38	80.66
×	√	95.25	83.56	80.82
√	×	95.50	84.10	81.59
√	√	**95.63**	**84.20**	**81.76**

"tag → parse" denotes that parsing leverages lexical information and "tag ← parse" denotes that tagging exploits syntactic information. The interactions can be disabled as shown in Eq. (10)–(12). The tagging and parsing results are evaluated on the Chinese Penn Treebank development set.

IV. Related Work

Our work is closely related to two lines of research: (1) joint POS tagging and syntactic parsing using feature templates, and (2) neural syntactic parsing.

A. Joint Modeling With Feature Templates

Most previous endeavors on joint POS tagging and dependency parsing have focused on developing linear models with feature templates [11]–[13], [28]. They introduce transition systems that can perform POS tagging and dependency parsing in a joint search space.

Our transition system differs from previous work in the separation of structural, tagging, and labeling actions. This results in three small classifiers with fewer classes (i.e., $|\mathcal{T}|$ classes for the tag classifier, 3 for the shift/reduce classifier, and $|\mathcal{L}|$ for the label classifier) rather than one big classifier with much more classes (i.e., $|\mathcal{T}| + 2|\mathcal{L}|$).

More importantly, we use continuous representations instead of discrete indicator features to build the classifiers. As indicated by Chen and Manning [7], lexicalized indicator features crucial for improving parsing accuracy are highly sparse and often incomplete. Alternatively, we resort to neural networks to learn representations from data to circumvent the sparsity and incompleteness problems. Another benefit of using neural networks is that there is no need to compose individual features to obtain more complex features like conventional discriminative dependency parsing [8].

It should also be mentioned that there are interactions between the decisions regarding POS tagging and syntactic parsing in research on constituent parsing [29], [30].

B. Neural POS Tagging and Syntactic Parsing

Our work is also inspired by recent advances in applying neural networks to POS tagging [4], dependency parsing [7]–[9], [17], [19]–[21], [24], [31], [32] and constituency parsing [33]–[36].

Among them, our work bears the most resemblance to [18], which propose stack-propagation to integrate a tagging model into a neural parser. They propose a stacked pipeline of models and utilize POS tags as a regularizer of learned representations. While Zhang and Weiss [18] use the hidden layer of the tagger network as the input for the parser, we are interested in enabling tagging and parsing to benefit each other in a joint search space. As a result, the tagger is able to resolve long-distance tagging ambiguity by exploiting syntactic information. Meanwhile, the error propagation problem the parser faces can be alleviated due to the cascaded error reduction by joint modeling.

V. Conclusion

We have presented an approach to joint part-of-speech tagging and dependency parsing using transition-based neural networks. Based on a five-action transition system, we develop three classifiers to resolve structural, tagging, and labeling conflicts. As our approach allows lexicality and syntax to interact with each other in the joint search process, it improves over previous work on joint POS tagging and dependency parsing on three treebanks across a variety of natural languages. Our code is released at https://github.com/tianlinyang/joint-parser.

References

[1] M. Collins, "Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms," in *Proc. Empirical Methods Natural Lang. Process.*, 2002.
[2] K. Toutanova, D. Klein, C. D. Manning, and Y. Singer, "Feature-rich part-of-speech tagging with a cyclic dependency network," in *Proc. North Amer. Chapter Assoc. Comput. Linguistics Human Lang. Technol.*, 2003, pp. 173–180.
[3] C. N. dos Santos and B. Zadrozny, "Learning character-level representations for part-of-speech tagging," in *Proc. Int. Conf. Mach. Learn.*, 2014, pp. 1818–1826.
[4] Z. Huang, W. Xu, and K. Yu, "Bidirectional LSTM-CRF models for sequence tagging," arXiv:1508.01991, 2015.
[5] R. McDonald, K. Crammer, and F. Pereira, "Online large-margin training of dependency parsers," in *Proc. 43rd Annu. Meeting Assoc. Comput. Linguistics*, 2005, pp. 91–98.
[6] J. Nivre, J. Hall, J. Nilsson, G. Eryigit, and S. Marinov, "Labeled pseudo-projective dependency parsing with support vector machines," in *Proc. 10th Conf. Comput. Natural Lang. Learn.*, 2006, pp. 221–225.
[7] D. Chen and C. D. Manning, "A fast and accurate dependency parser using neural networks," in *Proc. 2014 Conf. Empirical Methods Natural Lang. Process.*, 2014, pp. 740–750.
[8] C. Dyer, M. Ballesteros, W. Ling, A. Mattews, and A. Smith, Noah, "Transition-based depdnency parsing with stack long short-term memory," in *Proc. 53rd Annu. Meeting Assoc. Comput. Linguistics, 7th Int. Joint Conf. Natural Lang. Process.*, 2015, pp. 334–343.
[9] E. Kiperwasser and Y. Goldberg, "Simple and accurate dependency parsing using bidirectional LSTM feature representations," in *Proc. Trans. Assoc. Comput. Linguistics*, 2016, pp. 313–327.
[10] W. Sun, X. Peng, and X. Wan, "Capturing long-distance dependencies in sequence models: A case study of Chinese part-of-speech tagging," in *Proc. Int. Joint Conf. Natural Lang. Process.*, 2013.
[11] J. Hatori, T. Matsuzaki, Y. Miyao, and J. Tsujii, "Incremental joint POS tagging and dependency parsing in Chinese," in *Proc. Int. Joint Conf. Natural Lang. Process.*, 2011.
[12] Z. Li, M. Zhang, W. Che, T. Liu, W. Chen, and H. Li, "Joint models for Chinese POS tagging and dependency parsing," in *Proc. Empirical Methods Natural Lang. Process.*, 2011, pp. 1180–1191.
[13] B. Bohnet and J. Nivre, "A transition-based system for joint part-of-speech tagging and labeled non-projective parsing," in *Proc. Empirical Methods Natural Lang. Process.*, 2012, pp. 1455–1465.
[14] Z. Li, M. Zhang, W. Che, and T. Liu, "A separately passive-aggressive training algorithm for joint POS tagging and dependency parsing," in *Proc. Int. Conf. Comput. Linguistics*, 2012, pp. 1681–1698.
[15] M. Zhang, W. Che, T. Liu, and Z. Li, "Stacking heterogeneous joint models of Chinese POS tagging and dependency parsing," in *Proc. Int. Conf. Comput. Linguistics*, 2012, pp. 3071–3088.
[16] J. Nivre, "Algorithms for deterministic incremental depdendency parsing," *Comput. Linguist.*, vol. 34, pp. 513–553, 2008.

[17] M. Ballesteros, C. Dyer, and N. Smith, "Improved transition-based parsing by modeling characters instead of words with LSTMs," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2015, pp. 349–359.

[18] Y. Zhang and D. Weiss, "Stack-propagation: Improved representation learning for syntax," in *Proc. Assoc. Comput. Linguistics*, 2016.

[19] D. Andor *et al.*, "Globally normalized transition-based neural networks," in *Proc. Assoc. Comput. Linguistics*, 2016.

[20] H. Cheng, H. Fang, X. He, J. Gao, and L. Deng, "Bi-directional attention with agreement for dependency parsing," in *Proc. Empirical Methods Natural Lang. Process.*, 2016, pp. 2204–2214.

[21] T. Dozat and C. D. Manning, "Deep Biaffine attention for neural dependency parsing," in *Proc. Int. Conf. Learn. Representations*, 2017.

[22] M.-C. de Marneffe, B. MacCartney, and C. D. Manning, "Generating typed dependency parses from phrase structure parses," in *Proc. Int. Conf. Lang. Resour. Eval.*, 2006, pp. 449–454.

[23] Y. Zhang and S. Clark, "A tale of two parsers: Investigating and combining graph-based and transition-based dependency parsing," in *Proc. Empirical Methods Natural Lang. Process.*, 2008, pp. 562–571.

[24] W. Ammar, G. Mulcaire, M. Ballesteros, C. Dyer, and N. A. Smith, "Many languages, one parser," *Trans. Assoc. Comput. Linguistics*, vol. 6, 2016.

[25] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Representations*, 2015.

[26] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *Proc. 30th Int. Conf. Mach. Learn.*, 2013, pp. 1310–1318.

[27] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhtdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, pp. 1929–1958, 2014.

[28] B. Bohnet, J. Nivre, I. Boguslavsky, R. Farkas, F. Ginter, and J. Hajic, "Joint morphological and syntactic analysis for richly inflected languages," *Trans. Assoc. Comput. Linguistics*, vol. 1, pp. 415–428, 2013.

[29] Z. Wang, C. Zong, and N. Xue, "A lattice-based framework for joint chinese word segmentation, POS tagging and parsing," in *Proc. 51st Annu. Meeting Assoc. Comput. Linguistics*, 2013, pp. 623–627.

[30] Z. Wang and N. Xue, "Joint POS tagging and transition-based constituent parsing in chinese with non-local features," in *Proc. Annu. Meeting Assoc. Comput. Linguistics*, 2014, pp. 733–742.

[31] C. Alberti, D. Weiss, G. Coppola, and S. Petrov, "Improved transition-based parsing and tagging with neural networks," in *Proc. Empirical Methods Natural Lang. Process.*, 2015.

[32] W. Wang and B. Chang, "Graph-based dependency parsing with bidirectional LSTM," in *Proc. 54th Annu. Meeting Assoc. Comput. Linguistics*, 2016, pp. 2306–2315.

[33] C. Dyer, A. Kuncoro, M. Ballesteros, and N. A. Smith, "Recurrent neural network grammars," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics*, 2016.

[34] J. Cross and L. Huang, "Span-based constituency parsing with a structure-label system and provably optimal dynamic oracles," in *Proc. Empirical Methods Natural Lang. Process.*, 2016.

[35] A. Kuncoro, M. Ballesteros, L. Kong, C. Dyer, G. Neubig, and N. A. Smith, "What do recurrent neural network grammars learn about syntax?" in *Proc. Eur. Chapter Assoc. Comput. Linguistics*, 2017.

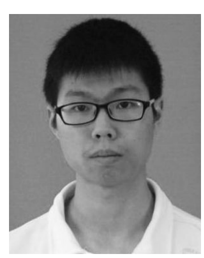
[36] J. Liu and Y. Zhang, "In-order transition-based constituent parsing," *Trans. Assoc. Comput. Linguistics*, vol. 5, 2017.

Liner Yang is currently working toward the Ph.D. degree at the Department of Computer Science and Technology, Tsinghua University, Beijing, China. His current research interests include representation learning and syntactic parsing.

Meishan Zhang received the Ph.D. degree in computer science from the Harbin Institute of Technology, Harbin, China, in 2015. He is currently an Associate Professor with Heilongjiang University, Harbin, China. His current research interests include natural language processing, machine learning, and text mining.

Yang Liu received the Ph.D. degree from the Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China, in 2007. He is currently an Associate Professor with the Department of Computer Science and Technology, Tsinghua University, Beijing, China. His research interests include natural language processing and machine translation.

Maosong Sun received the Ph.D. degree in computational linguistics from the City University of Hong Kong, Hong Kong, in 2004. He is currently a Professor with the Department of Computer Science and Technology, Tsinghua University, Beijing, China. His research interests include natural language processing, web intelligence, and machine learning.



Nan Yu is currently working toward the Master degree at the School of Computer Science and Technology in Heilongjiang University, Harbin, China. His current research interest include syntactic parsing.

Guohong Fu received the Ph.D. degree in computer science from the Harbin Institute of Technology, Harbin, China, in 2001. He is currently a Professor with Heilongjiang University, Harbin, China.

His current research interests include natural language processing, Chinese computing, and text mining.