Neural Joint Model for Transition-based Chinese Syntactic Analysis

Shuhei Kurita **Daisuke Kawahara** **Sadao Kurohashi**
Graduate School of Informatics, Kyoto University
{kurita, dk, kuro}@nlp.ist.i.kyoto-u.ac.jp

Abstract

We present neural network-based joint models for Chinese word segmentation, POS tagging and dependency parsing. Our models are the first neural approaches for fully joint Chinese analysis that is known to prevent the error propagation problem of pipeline models. Although word embeddings play a key role in dependency parsing, they cannot be applied directly to the joint task in the previous work. To address this problem, we propose embeddings of character strings, in addition to words. Experiments show that our models outperform existing systems in Chinese word segmentation and POS tagging, and perform preferable accuracies in dependency parsing. We also explore bi-LSTM models with fewer features.

1 Introduction

Dependency parsers have been enhanced by the use of neural networks and embedding vectors (Chen and Manning, 2014; Weiss et al., 2015; Zhou et al., 2015; Alberti et al., 2015; Andor et al., 2016; Dyer et al., 2015). When these dependency parsers process sentences in English and other languages that use symbols for word separations, they can be very accurate. However, for languages that do not contain word separation symbols, dependency parsers are used in pipeline processes with word segmentation and POS tagging models, and encounter serious problems because of error propagations. In particular, Chinese word segmentation is notoriously difficult because sentences are written without word dividers and Chinese words are not clearly defined. Hence, the pipeline of word segmentation, POS tagging and dependency parsing always suffers from word segmentation errors. Once words have been wrongly-segmented, word embeddings and traditional one-hot word features, used in dependency parsers, will mistake the precise meanings of the original sentences. As a result, pipeline models achieve dependency scores of around 80% for Chinese.

A traditional solution to this error propagation problem is to use joint models. Many Chinese words play multiple grammatical roles with only one grammatical form. Therefore, determining the word boundaries and the subsequent tagging and dependency parsing are closely correlated. Transition-based joint models for Chinese word segmentation, POS tagging and dependency parsing are proposed by Hatori et al. (2012) and Zhang et al. (2014). Hatori et al. (2012) state that dependency information improves the performances of word segmentation and POS tagging, and develop the first transition-based joint word segmentation, POS tagging and dependency parsing model. Zhang et al. (2014) expand this and find that both the inter-word dependencies and intra-word dependencies are helpful in word segmentation and POS tagging.

Although the models of Hatori et al. (2012) and Zhang et al. (2014) perform better than pipeline models, they rely on the one-hot representation of characters and words, and do not assume the similarities among characters and words. In addition, not only words and characters but also many incomplete tokens appear in the transition-based joint parsing process. Such incomplete or unknown words (UNK) could become important cues for parsing, but they are not listed in dictionaries or pre-trained word embeddings. Some recent studies show that character-based embeddings are effective in neural parsing (Ballesteros et al., 2015; Zheng et al., 2015), but their models could not be directly applied to joint models because they use given word segmentations. To solve

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 1204–1214
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1111

these problems, we propose neural network-based joint models for word segmentation, POS tagging and dependency parsing. We use both character and word embeddings for known tokens and apply character string embeddings for unknown tokens.

Another problem in the models of Hatori et al. (2012) and Zhang et al. (2014) is that they rely on detailed feature engineering. Recently, bidirectional LSTM (bi-LSTM) based neural network models with very few feature extraction are proposed (Kiperwasser and Goldberg, 2016; Cross and Huang, 2016). In their models, the bi-LSTM is used to represent the tokens including their context. Indeed, such neural networks can observe whole sentence through the bi-LSTM. This bi-LSTM is similar to that of neural machine translation models of Bahdanau et al. (2014). As a result, Kiperwasser and Goldberg (2016) achieve competitive scores with the previous state-of-the-art models. We also develop joint models with n-gram character string bi-LSTM.

In the experiments, we obtain state-of-the-art Chinese word segmentation and POS tagging scores, and the pipeline of the dependency model achieves the better dependency scores than the previous joint models. To the best of our knowledge, this is the first model to use embeddings and neural networks for Chinese full joint parsing.

Our contributions are summarized as follows: (1) we propose the first embedding-based fully joint parsing model, (2) we use character string embeddings for UNK and incomplete tokens. (3) we also explore bi-LSTM models to avoid the detailed feature engineering in previous approaches. (4) in experiments using Chinese corpus, we achieve state-of-the-art scores in word segmentation, POS tagging and dependency parsing.

2 Model

All full joint parsing models we present in this paper use the transition-based algorithm in Section 2.1 and the embeddings of character strings in Section 2.2. We present two neural networks: the feed-forward neural network models in Section 2.3 and the bi-LSTM models in Section 2.4.

2.1 Transition-based Algorithm for Joint Segmentation, POS Tagging, and Dependency Parsing

Based on Hatori et al. (2012), we use a modified arc-standard algorithm for character transitions (Figure 1). The model consists of one buffer and one stack. The buffer contains characters in the input sentence, and the stack contains words shifted from the buffer. The stack words may have their child nodes. The words in the stack are formed by the following transition operations.

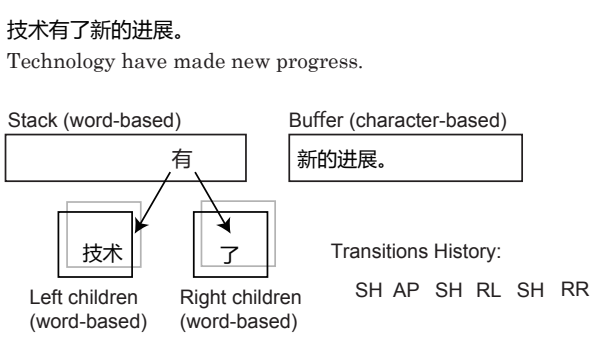

Figure 1: Transition-based Chinese joint model for word segmentation, POS tagging and dependency parsing.

- `SH(t)` (*shift*): Shift the first character of the buffer to the top of the stack as a new word.
- `AP` (*append*): Append the first character of the buffer to the end of the top word of the stack.
- `RR` (*reduce-right*): Reduce the right word of the top two words of the stack, and make the right child node of the left word.
- `RL` (*reduce-left*): Reduce the left word of the top two words of the stack, and make the left child node of the right word.

The `RR` and `RL` operations are the same as those of the arc-standard algorithm (Nivre, 2004a). `SH` makes a new word whereas `AP` makes the current word longer by adding one character. The POS tags are attached with the `SH(t)` transition.

In this paper, we explore both greedy models and beam decoding models. This parsing algorithm works in both types. We also develop a joint model of word segmentation and POS tagging, along with a dependency parsing model. The joint model of word segmentation and POS tagging does not have `RR` and `RL` transitions.

2.2 Embeddings of Character Strings

First, we explain the embeddings used in the neural networks. Later, we explain details of the neural networks in Section 2.3 and 2.4.

Both meaningful words and incomplete tokens appear during transition-based joint parsing. Although embeddings of incomplete tokens are not used in previous work, they could become useful features in several cases. For example, "南京东路" (Nanjing East Road, the famous shopping street of Shanghai) is treated as a single Chinese word in the Penn Chinese Treebank (CTB) corpus. There are other named entities of this form in CTB, e.g, "北京西路" (Beijing West Road) and "湘西路" (Hunan West Road). In these cases, "南京" (Nanjing) and "北京" (Beijing) are location words, while "东路" (East Road) and "西路" (West Road) are sub-words. "东路" and "西路" are similar in terms of their character composition and usage, which is not sufficiently considered in the previous work. Moreover, representations of incomplete tokens are helpful for compensating the segmentation ambiguity. Suppose that the parser makes over-segmentation errors and segments "南京东路" to "南京" and "东路". In this case, "东路" becomes UNK. However, the models could infer that "东路" is also a location, from its character composition and neighboring words. This could give models robustness of segmentation errors. In our models, we prepare the word and character embeddings in the pre-training. We also use the embeddings of character strings for sub-words and UNK which are not in the pre-trained embeddings.

The characters and words are embedded in the same vector space during pre-training. We prepare the same training corpus with the segmented word files and the segmented character files. Both files are concatenated and learned by word2vec (Mikolov et al., 2013). We use the embeddings of 1M frequent words and characters. Words and characters that are in the training set and do not have pre-trained embeddings are given randomly initialized embeddings. The development set and the test set have out-of-vocabulary (OOV) tokens for these embeddings.

The embeddings of the unknown character strings are generated in the neural computation graph when they are required. Consider a character string $c_1 c_2 \cdots c_n$ consisting of characters c_i. When this character string is not in the pre-trained embeddings, the model obtains the embeddings $\mathbf{v}(c_1 c_2 \cdots c_n)$ by the mean of each character embeddings $\sum_{i=1}^{n} \mathbf{v}(c_i)$. Embeddings of words, characters and character strings have the same dimension and are chosen in the neural computation graph. We avoid using the "UNK" vector as far as possible, because this degenerates the information about unknown tokens. However, models use the "UNK" vector if the parser encounters characters that are not in the pre-trained embeddings, though this is quite uncommon.

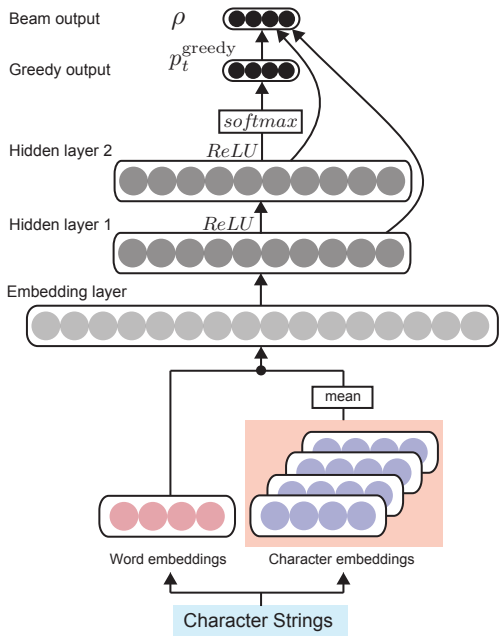

Figure 2: The feed-forward neural network model. The greedy output is obtained at the second top layer, while the beam decoding output is obtained at the top layer. The input character strings are translated into word embeddings if the embeddings of the character strings are available. Otherwise, the embeddings of the character strings are used.

2.3 Feed-forward Neural Network

2.3.1 Neural Network

We present a feed-forward neural network model in Figure 2. The neural network for greedy training is based on the neural networks of Chen and Manning (2014) and Weiss et al. (2015). We add the dynamic generation of the embeddings of character strings for unknown tokens, as described in Section 2.2. This neural network has two hidden layers with 8,000 dimensions. This is larger than Chen and Manning (2014) (200 dimensions) or Weiss et al. (2015) (1,024 or 2,048 dimensions). We use the ReLU for the activation function of the hidden layers (Nair and Hinton, 2010) and the softmax function for the output layer of the greedy

Type	Value
Size of $\mathbf{h_1}$,$\mathbf{h_2}$	8,000
Initial learning rate	0.01
Initial learning rate of beam decoding	0.001
Embedding vocabulary size	1M
Embedding vector size	200
Small embedding vector size	20
Minibatch size	200

Table 1: Parameters for neural network structure and training.

neural network. There are three randomly initialized weight matrices between the embedding layers and the softmax function. The loss function $L(\theta)$ for the greedy training is

$$L(\theta) = -\sum_{s,t} \log p_{s,t}^{\text{greedy}} + \frac{\lambda}{2}||\theta||^2,$$

$$p_{s,t}^{\text{greedy}}(\boldsymbol{\beta}) \propto \exp\left(\sum_j w_{tj}\beta_j + b_t\right),$$

where t denotes one transition among the transition set $\mathcal{T}$ ($t \in \mathcal{T}$). s denotes one element of the single mini-batch. $\boldsymbol{\beta}$ denotes the output of the previous layer. $\mathbf{w}$ and $\mathbf{b}$ denote the weight matrix and the bias term. θ contains all parameters. We use the $L2$ penalty term and the Dropout. The backprop is performed including the word and character embeddings. We use Adagrad (Duchi et al., 2010) to optimize learning rate. We also consider Adam (Kingma and Ba, 2015) and SGD, but find that Adagrad performs better in this model. The other learning parameters are summarized in Table 1.

In our model implementation, we divide all sentences into training batches. Sentences in the same training batches are simultaneously processed by the neural mini-batches. By doing so, the model can parse all sentences of the training batch in the number of transitions required to parse the longest sentence in the batch. This allows the model to parse more sentences at once, as long as the neural mini-batch can be allocated to the GPU memory. This can be applied to beam decoding.

2.3.2 Features

The features of this neural network are listed in Table 2. We use three kinds of features: (1) features obtained from Hatori et al. (2012) by removing combinations of features, (2) features obtained from Chen and Manning (2014), (3) original features related to character strings. In particular, the original features include sub-words, character strings across the buffer and the stack, and character strings in the buffer. Character strings across the buffer and stack could capture the currently-segmented word. To avoid using character strings that are too long, we restrict the length of character string to a maximum of four characters. Unlike Hatori et al. (2012), we use sequential characters of sentences for features, and avoid hand-engineered combinations among one-hot features, because such combinations could be automatically generated in the neural hidden layers as distributed representations (Hinton et al., 1986).

Type	Features
Stack word and tags	s0w, s1w, s2w
	s0p, s1p, s2p
Stack 1 children and tags	s0l0w, s0r0w, s0l1w, s0r1w
	s0l0p, s0r0p, s0l1p, s0r1p
Stack 2 children	s1l0w, s1r0w, s1l1w, s1r1w
Children of children	s0l0lw, s0r0rw, s1l0lw, s1r0rw
Buffer characters	b0c, b1c, b2c, b3c
Previously shifted words	q0w, q1w
Previously shifted tags	q0p, q1p
Character of q0	q0e
Parts of q0 word	q0f1, q0f2, q0f3
Strings across q0 and buf.	q0b1, q0b2, q0b3
Strings of buffer characters	b0-2, b0-3, b0-4
	b1-3, b1-4, b1-5
	b2-4, b2-5, b2-6
	b3-5, b3-6
	b4-6
Length of q0	lenq0

Table 2: Features for the joint model. "q0" denotes the last shifted word and "q1" denotes the word shifted before "q0". In "part of q0 word", "f1", "f2" and "f3" denote sub-words of "q0", which are 1, 2 and 3 sequential characters including the last character of "q0" respectively. In "strings across q0 and buf.", "q0bX" denotes "q0" and X sequential characters of the buffer. This feature could capture words that boundaries have not determined yet. In "strings of buffer characters", "bX-Y" denotes sequential characters from the X-th to Y-th character of the buffer. The suffix "e" denotes the end character of the word. The dimension of the embedding of "length of q0" is 20.

In the later section, we evaluate a joint model for word segmentation and POS tagging. This model does not use the children and children-of-children of stack words as features.

2.3.3 Beam Search

Structured learning plays an important role in previous joint parsing models for Chinese.[1] In this paper, we use the structured learning model proposed by Weiss et al. (2015) and Andor et al. (2016).

In Figure 2, the output layer for the beam decoding is at the top of the network. There are a perceptron layer which has inputs from the two hidden layers and the greedy output layer: $[\mathbf{h}_1, \mathbf{h}_2, \mathbf{p}^{\text{greedy}}(\mathbf{y})]$. This layer is learned by the following cost function (Andor et al., 2016):

$$L(d^*_{1:j};\theta) = -\sum_{i=1}^{j}\rho(d^*_{1:i-1}, d^*_i;\theta) + \ln \sum_{d'_{1:j}\in\mathcal{B}_{1:j}} \exp \sum_{i=1}^{j}\rho(d'_{1:i-1}, d'_i;\theta),$$

where $d_{1:j}$ denotes the transition path and $d^*_{1:j}$ denotes the gold transition path. $\mathcal{B}_{1:j}$ is the set of transition paths from 1 to j step in beam. ρ is the value of the top layer in Figure 2. This training can be applied throughout the network. However, we separately train the last beam layer and the previous greedy network in practice, as in Andor et al. (2016). First, we train the last perceptron layer using the beam cost function freezing the previous greedy-trained layers. After the last layer has been well trained, backprop is performed including the previous layers. We notice that training the embedding layer at this stage could make the results worse, and thus we exclude it. Note that this whole network backprop requires considerable GPU memory. Hence, we exclude particularly large batches from the training, because they cannot be on GPU memory. We use multiple beam sizes for training because models can be trained faster with small beam sizes. After the small beam size training, we use larger beam sizes. The test of this fully joint model takes place with a beam size of 16.

Hatori et al. (2012) use special alignment steps in beam decoding. The `AP` transition has size-2 steps, whereas the other transitions have a size-1 step. Using this alignment, the total number of steps for an N-character sentence is guaranteed to be $2N-1$ (excluding the root arc) for any transition path. This can be interpreted as the `AP` transition doing two things: appending characters and resolving intra-word dependencies. This alignment stepping assumes that the intra-word dependencies of characters to the right of the characters exist in each Chinese word.

[1] Hatori et al. (2012) report that structured learning with a beam size of 64 is optimal.

2.4 Bi-LSTM Model

In Section 2.3, we describe a neural network model with feature extraction. Unfortunately, although this model is fast and very accurate, it has two problems: (1) the neural network cannot see the whole sentence information. (2) it relies on feature engineering. To solve these problems, Kiperwasser and Goldberg (2016) propose a bi-LSTM neural network parsing model. Surprisingly, their model uses very few features, and bi-LSTM is applied to represent the context of the features. Their neural network consists of three parts: bi-LSTM, a feature extraction function and a multilayer perceptron (MLP). First, all tokens in the sentences are converted to embeddings. Second, the bi-LSTM reads all embeddings of the sentence. Third, the feature function extracts the feature representations of tokens from the bi-LSTM layer. Finally, an MLP with one hidden layer outputs the transition scores of the transition-based parser.

In this paper, we propose a Chinese joint parsing model with simple and global features using n-gram bi-LSTM and a simple feature extraction function. The model is described in Figure 3. We consider that Chinese sentences consist of tokens, including words, UNKs and incomplete tokens, which can have some meanings and are useful for parsing. Such tokens appear in many parts of the sentence and have arbitrary lengths. To capture them, we propose the n-gram bi-LSTM. The n-gram bi-LSTM read through characters $c_i \cdots c_{i+n-1}$ of the sentence (c_i is the i-th character). For example, the 1-gram bi-LSTM reads each character, and the 2-gram bi-LSTM reads two consecutive characters $c_i c_{i+1}$. After the n-gram forward LSTM reads character string $c_i \cdots c_{i+n-1}$, it next reads $c_{i+1} \cdots c_{i+n}$. The backward LSTM reads from $c_{i+1} \cdots c_{i+n}$ toward $c_i \cdots c_{i+n-1}$. This allows models to capture any n-gram character strings in the input sentence.[2] All n-gram inputs to bi-LSTM are given by the embeddings of words and characters or the dynamically generated embeddings of character strings, as described in

[2] At the end of the sentence of length N, character strings $c_i \cdots c_N (N < i+n-1)$, which are shorter than n characters, are used.

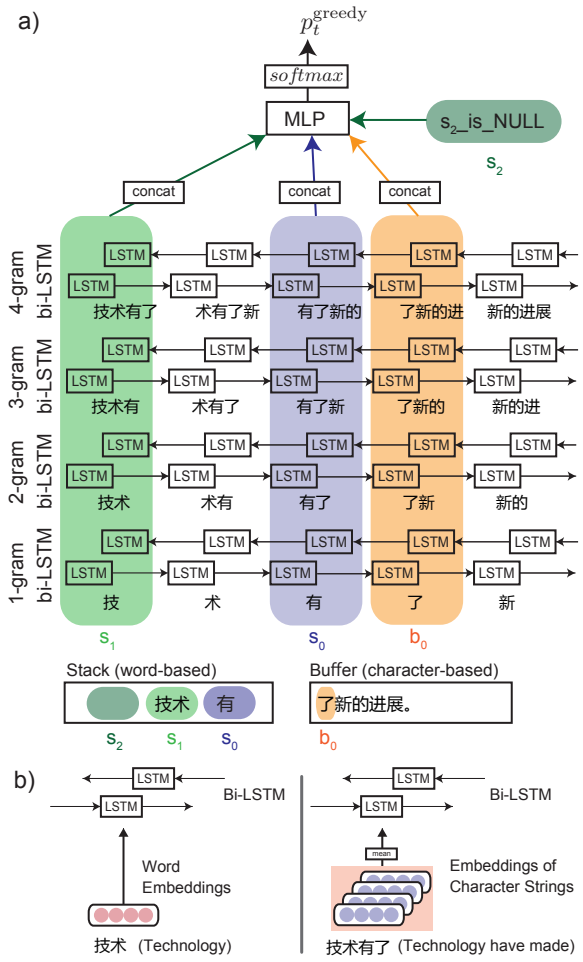

Figure 3: The bi-LSTM model. (a): The Chinese sentence "技术有了新的进展。" has been processed. (b): Similar to the feed-forward neural network model, the embeddings of words, characters and character strings are used. In this figure, a word "技术"(technology) has its embedding, while a token "技术有了"(technology have made) does not.

Section 2.2. Although these arbitrary n-gram tokens produce UNKs, character string embeddings can capture similarities among them. Following the bi-LSTM layer, the feature function extracts the corresponding outputs of the bi-LSTM layer. We summarize the features in Table 3. Finally, MLP and the softmax function outputs the transition probability. We use an MLP with three hidden layers as for the model in Section 2.3. We train this neural network with the loss function for the greedy training.

Model	Features
4 features	s0w, s1w, s2w, b0c
8 features	s0w, s1w, s2w, b0c s0r0w, s0l0w, s1r0w, s1l0w

Table 3: Features for the bi-LSTM models. All features are words and characters. We experiment both four and eight features models.

		#snt	#oov
CTB-5	Train	18k	-
	Dev.	350	553
	Test	348	278
CTB-7	Train	31k	-
	Dev.	10k	13k
	Test	10k	13k

Table 4: Summary of datasets.

3 Experiments

3.1 Experimental Settings

We use the Penn Chinese Treebank 5.1 (CTB-5) and 7 (CTB-7) datasets to evaluate our models, following the splitting of Jiang et al. (2008) for CTB-5 and Wang et al. (2011) for CTB-7. The statistics of datasets are presented in Table 4. We use the Chinese Gigaword Corpus for embedding pre-training. Our model is developed for unlabeled dependencies. The development set is used for parameter tuning. Following Hatori et al. (2012) and Zhang et al. (2014), we use the standard word-level evaluation with F1-measure. The POS tags and dependencies cannot be correct unless the corresponding words are correctly segmented.

We trained three models: SegTag, SegTagDep and Dep. SegTag is the joint word segmentation and POS tagging model. SegTagDep is the full joint segmentation, tagging and dependency parsing model. Dep is the dependency parsing model which is similar to Weiss et al. (2015) and Andor et al. (2016), but uses the embeddings of character strings. Dep compensates for UNKs and segmentation errors caused by previous word segmentation using embeddings of character strings. We will examine this effect later.

Most experiments are conducted on GPUs, but some of beam decoding processes are performed on CPUs because of the large mini-batch size. The neural network is implemented with Theano.

Model	Seg	POS
Hatori+12 SegTag	97.66	93.61
Hatori+12 SegTag(d)	98.18	94.08
Hatori+12 SegTagDep	97.73	94.46
Hatori+12 SegTagDep(d)	98.26	94.64
M. Zhang+14 EAG	97.76	94.36
Y. Zhang+15	98.04	94.47
SegTag(g)	98.41	**94.84**
SegTag	**98.60**	94.76

Table 5: Joint segmentation and POS tagging scores. Both scores are in F-measure. In Hatori et al. (2012), (d) denotes the use of dictionaries. (g) denotes greedy trained models. All scores for previous models are taken from Hatori et al. (2012), Zhang et al. (2014) and Zhang et al. (2015).

3.2 Results

3.2.1 Joint Segmentation and POS Tagging

First, we evaluate the joint segmentation and POS tagging model (SegTag). Table 5 compares the performance of segmentation and POS tagging using the CTB-5 dataset. We train two modles: a greedy-trained model and a model trained with beams of size 4. We compare our model to three previous approaches: Hatori et al. (2012), Zhang et al. (2014) and Zhang et al. (2015). Our SegTag joint model is superior to these previous models, including Hatori et al. (2012)'s model with rich dictionary information, in terms of both segmentation and POS tagging accuracy.

3.2.2 Joint Segmentation, POS Tagging and Dependency Parsing

Table 6 presents the results of our full joint model. We employ the greedy trained full joint model SegTagDep(g) and the beam decoding model SegTagDep. All scores for the existing models in this table are taken from Zhang et al. (2014). Though our model surpasses the previous best end-to-end joint models in terms of segmentation and POS tagging, the dependency score is slightly lower than the previous models. The greedy model SegTagDep(g) achieves slightly lower scores than beam models, although this model works considerably fast because it does not use beam decoding.

Model	Seg	POS	Dep
Hatori+12	97.75	94.33	81.56
M. Zhang+14 EAG	97.76	94.36	**81.70**
SegTagDep(g)	98.24	94.49	80.15
SegTagDep	**98.37**	**94.83**	81.42

Table 6: Joint Segmentation, POS Tagging and Dependency Parsing. Hatori et al. (2012)'s CTB-5 scores are reported in Zhang et al. (2014). EAG in Zhang et al. (2014) denotes the arc-eager model. (g) denotes greedy trained models.

Model	Seg	POS	Dep
Hatori+12	97.75	94.33	81.56
M. Zhang+14 STD	97.67	94.28	81.63
M. Zhang+14 EAG	97.76	94.36	81.70
Y. Zhang+15	98.04	94.47	82.01
SegTagDep(g)	98.24	94.49	80.15
SegTagDep	98.37	**94.83**‡	81.42‡
SegTag+Dep	**98.60**‡	94.76‡	**82.60**‡

Table 7: The SegTag+Dep model. Note that the model of Zhang et al. (2015) requires other base parsers. ‡ denotes that the improvement is statistically siginificant at $p < 0.01$ compared with SegTagDep(g) using paired t-test.

3.2.3 Pipeline of Our Joint SegTag and Dep Model

We use our joint SegTag model for the pipeline input of the Dep model (SegTag+Dep). Both SegTag and Dep models are trained and tested by the beam cost function with beams of size 4. Table 7 presents the results. Our SegTag+Dep model performs best in terms of the dependency and word segmentation. The SegTag+Dep model is better than the full joint model. This is because most segmentation errors of these models occur around named entities. Hatori et al. (2012)'s alignment step assumes the intra-word dependencies in words, while named entities do not always have them. For example, SegTag+Dep model treats named entity "海赛克", a company name, as one word, while the SegTagDep model divides this to "海" (sea) and "赛克", where "赛克" could be used for foreigner's name. For such words, SegTagDep prefers `SH` because `AP` has size-2 step of the character appending and intra-word dependency resolution, which does not exist for named entities. This problem could be solved by adding a special transition `AP_named_entity` which is similar to `AP` but with size-1 step and used

Model	Dep
Dep(g)-cs	80.51
Dep(g)	80.98

Table 8: SegTag+Dep(g) model with and without character strings (cs) representations. Note that we compare these models with greedy training for simplicity's sake.

Model	Seg	POS	Dep
SegTagDep(g) -q0bX	97.81	93.79	79.16
SegTagDep(g)	98.24	94.49	80.15

Table 9: SegTagDep model with and without (-q0bX) features across the buffer and stack. We compare these models with greedy training (g).

Model	Seg	POS	Dep
Hatori+12	95.42	90.62	73.58
M. Zhang+14 STD	95.53	90.75	**75.63**
SegTagDep(g)	96.06	90.28	73.98
SegTagDep	95.86	90.91‡	74.04
SegTag+Dep	**96.23**‡	**91.25**‡	75.28‡

Table 10: Results from SegTag+Dep and SegTagDep applied to the CTB-7 corpus. (g) denotes greedy trained models. ‡ denotes that the improvement is statistically siginificant at $p < 0.01$ compared with SegTagDep(g) using paired t-test.

only for named entities. Additionally, Zhang et al. (2014)'s STD (arc-standard) model works slightly better than Hatori et al. (2012)'s fully joint model in terms of the dependency score. Zhang et al. (2014)'s STD model is similar to our SegTag+Dep because they combine a word segmentator and a dependency parser using "deque" of words.

3.2.4 Effect of Character String Embeddings

Finally, we compare the two pipeline models of SegTag+Dep to show the effectiveness of using character string representations instead of "UNK" embeddings. We use two dependency models with greedy training: Dep(g) for dependency model and Dep(g)-cs for dependency model without the character string embeddings . In the Dep(g)-cs model, we use the "UNK" embedding when the embeddings of the input features are unavailable, whereas we use the character string embeddings in model Dep(g). The results are presented in Table 8. When the models encounter unknown tokens, using the embeddings of character strings is better than using the "UNK" embedding.

3.2.5 Effect of Features across the Buffer and Stack

We test the effect of special features: q0bX in Table 2. The q0bX features capture the tokens across the buffer and stack. Joint transition-based parsing models by Hatori et al. (2012) and Chen and Manning (2014) decide POS tags of words before corresponding word segmentations are determined. In our model, the q0bX features capture words even if their segmentations are not determined. We examine the effectiveness of these features by training greedy full joint models with and without them. The results are shown in Table 9. The q0bX features boost not only POS tagging scores but also word segmentation scores.

3.2.6 CTB-7 Experiments

We also test the SegTagDep and SegTag+Dep models on CTB-7. In these experiments, we notice that the MLP with four hidden layers performs better than the MLP with three hidden layers, but we could not find definite differences in the experiments in CTB-5. We speculate that this is caused by the difference in the training set size. We present the final results with four hidden layers in Table 10.

3.2.7 Bi-LSTM Model

We experiment the n-gram bi-LSTMs models with four and eight features listed in Table 3. We summarize the result in Table 11. The greedy bi-LSTM models perform slightly worse than the previous models, but they do not rely on feature engineering.

4 Related Work

Zhang and Clark (2008) propose an incremental joint word segmentation and POS tagging model driven by a single perceptron. Zhang and Clark (2010) improve this model by using both character and word-based decoding. Hatori et al. (2011) propose a transition-based joint POS tagging and dependency parsing model. Zhang et al. (2013) propose a joint model using character structures of words for constituency parsing. Wang et al. (2013) also propose a lattice-based joint model for constituency parsing. Zhang et al. (2015) propose joint segmentation, POS tagging and dependency re-ranking system. This system requires

Model	Seg	POS	Dep
Hatori+12	97.75	94.33	81.56
M. Zhang+14 EAG	97.76	94.36	81.70
SegTagDep (g)	98.24	94.49	80.15
Bi-LSTM 4feat.(g)	97.72	93.12	79.03
Bi-LSTM 8feat.(g)	97.70	93.37	79.38

Table 11: Bi-LSTM feature extraction model. "4feat." and "8feat." denote the use of four and eight features.

base parsers. In neural joint models, Zheng et al. (2013) propose a neural network-based Chinese word segmentation model based on tag inferences. They extend their models for joint segmentation and POS tagging. Zhu et al. (2015) propose the re-ranking system of parsing results with recursive convolutional neural network.

5 Conclusion

We propose the joint parsing models by the feed-forward and bi-LSTM neural networks. Both of them use the character string embeddings. The character string embeddings help to capture the similarities of incomplete tokens. We also explore the neural network with few features using n-gram bi-LSTMs. Our SegTagDep joint model achieves better scores of Chinese word segmentation and POS tagging than previous joint models, and our SegTag and Dep pipeline model achieves state-of-the-art score of dependency parsing. The bi-LSTM models reduce the cost of feature engineering.

References

Chris Alberti, David Weiss, Greg Coppola, and Slav Petrov. 2015. Improved transition-based parsing and tagging with neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 1354–1359. http://aclweb.org/anthology/D15-1159.

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally normalized transition-based neural networks. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 2442–2452. http://www.aclweb.org/anthology/P16-1231.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR* abs/1409.0473. http://arxiv.org/abs/1409.0473.

Miguel Ballesteros, Chris Dyer, and Noah A. Smith. 2015. 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*. Association for Computational Linguistics, pages 349–359. http://aclweb.org/anthology/D15-1041.

Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, pages 740–750. http://www.aclweb.org/anthology/D14-1082.

James Cross and Liang Huang. 2016. Incremental parsing with minimal features using bi-directional lstm. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 32–37. http://anthology.aclweb.org/P16-2006.

John Duchi, Elad Hazan, and Yoram Singer. 2010. Adaptive subgradient methods for online learning and stochastic optimization. UCB/EECS-2010-24.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 334–343. http://www.aclweb.org/anthology/P15-1033.

Jun Hatori, Takuya Matsuzaki, Yusuke Miyao, and Jun'ichi Tsujii. 2011. Incremental joint pos tagging and dependency parsing in chinese. In *Proceedings of 5th International Joint Conference on Natural Language Processing*. Asian Federation of Natural Language Processing, pages 1216–1224. http://www.aclweb.org/anthology/I11-1136.

Jun Hatori, Takuya Matsuzaki, Yusuke Miyao, and Jun'ichi Tsujii. 2012. Incremental joint approach to word segmentation, pos tagging, and dependency parsing in chinese. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 1045–1053. http://www.aclweb.org/anthology/P12-1110.

Geoffrey E. Hinton, J. L. McClelland, and D. E. Rumelhart. 1986. Learning distributed representations of concepts. In *Proceedings of the eighth annual conference of the cognitive science society*. pages Vol.1, p.12.

Wenbin Jiang, Liang Huang, Qun Liu, and Yajuan Lü. 2008. A cascaded linear model for joint chinese word segmentation and part-of-speech tagging. In *Proceedings of ACL-08: HLT*. Association for Computational Linguistics, pages 897–904.

D. P. Kingma and J. Ba. 2015. Adam: A method for stochastic optimization. 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)*.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional lstm feature representations. *Transactions of the Association for Computational Linguistics* 4:313–327. https://transacl.org/ojs/index.php/tacl/article/view/885.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. volume abs/1301.3781. http://arxiv.org/abs/1301.3781.

Vinod Nair and Geoffrey E. Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10), June 21-24, 2010, Haifa, Israel*. pages 807–814.

Joakim Nivre. 2004a. Incrementality in deterministic dependency parsing. In Frank Keller, Stephen Clark, Matthew Crocker, and Mark Steedman, editors, *Proceedings of the ACL Workshop Incremental Parsing: Bringing Engineering and Cognition Together*. Association for Computational Linguistics, pages 50–57.

Yiou Wang, Jun'ichi Kazama, Yoshimasa Tsuruoka, Wenliang Chen, Yujie Zhang, and Kentaro Torisawa. 2011. Improving chinese word segmentation and pos tagging with semi-supervised methods using large auto-analyzed data. In *Proceedings of 5th International Joint Conference on Natural Language Processing*. Asian Federation of Natural Language Processing, Chiang Mai, Thailand, pages 309–317. http://www.aclweb.org/anthology/I11-1035.

Zhiguo Wang, Chengqing Zong, and Nianwen Xue. 2013. A lattice-based framework for joint chinese word segmentation, pos tagging and parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, Sofia, Bulgaria, pages 623–627. http://www.aclweb.org/anthology/P13-2110.

David Weiss, Chris Alberti, Michael Collins, and Slav Petrov. 2015. Structured training for neural network transition-based parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 323–333. http://www.aclweb.org/anthology/P15-1032.

Meishan Zhang, Yue Zhang, Wanxiang Che, and Ting Liu. 2013. Chinese parsing exploiting characters. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Sofia, Bulgaria, pages 125–134. http://www.aclweb.org/anthology/P13-1013.

Meishan Zhang, Yue Zhang, Wanxiang Che, and Ting Liu. 2014. Character-level chinese dependency parsing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 1326–1336. http://www.aclweb.org/anthology/P14-1125.

Yuan Zhang, Chengtao Li, Regina Barzilay, and Kareem Darwish. 2015. Randomized greedy inference for joint segmentation, pos tagging and dependency parsing. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligenc*. Association for Computational Linguistics, pages 42–52. http://www.aclweb.org/anthology/N15-1005.

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

Yue Zhang and Stephen Clark. 2010. A fast decoder for joint word segmentation and POS-tagging using a single discriminative model. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 843–852. http://www.aclweb.org/anthology/D10-1082.

Xiaoqing Zheng, Hanyang Chen, and Tianyu Xu. 2013. Deep learning for Chinese word segmentation and POS tagging. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 647–657. http://www.aclweb.org/anthology/D13-1061.

Xiaoqing Zheng, Haoyuan Peng, Yi Chen, Pengjing Zhang, and Zhang Wenqiang. 2015. Character-based parsing with convolutional neural network. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. page 153.

Hao Zhou, Yue Zhang, Shujian Huang, and Jiajun Chen. 2015. 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). Association for Computational Linguistics, pages 1213–1222. http://www.aclweb.org/anthology/P15-1117.

Chenxi Zhu, Xipeng Qiu, Xinchi Chen, and Xuanjing Huang. 2015. A re-ranking model for dependency parser with recursive convolutional neural network. 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)*. Association for Computational Linguistics, pages 1159–1168. http://www.aclweb.org/anthology/P15-1112.