# RNN-Based Sequence-Preserved Attention for Dependency Parsing

**Yi Zhou, Junying Zhou, Lu Liu, Jiangtao Feng, Haoyuan Peng, Xiaoqing Zheng**
School of Computer Science, Fudan University, Shanghai, China
Shanghai Key Laboratory of Intelligent Information Processing
{zhouyi13, junyingzhou14, l_liu15, fengjt16, hypeng15, zhengxq}@fudan.edu.cn

## Abstract

Recurrent neural networks (RNN) combined with attention mechanism has proved to be useful for various NLP tasks including machine translation, sequence labeling and syntactic parsing. The attention mechanism is usually applied by estimating the weights (or importance) of inputs and taking the weighted sum of inputs as derived features. Although such features have demonstrated their effectiveness, they may fail to capture the sequence information due to the simple weighted sum being used to produce them. The order of the words does matter to the meaning or the structure of the sentences, especially for syntactic parsing, which aims to recover the structure from a sequence of words. In this study, we propose an RNN-based attention to capture the relevant and sequence-preserved features from a sentence, and use the derived features to perform the dependency parsing. We evaluated the graph-based and transition-based parsing models enhanced with the RNN-based sequence-preserved attention on the both English PTB and Chinese CTB datasets. The experimental results show that the enhanced systems were improved with significant increase in parsing accuracy.

## Introduction

Typical methods to dependency parsing can be partitioned into graph-based and transition-based parsers. Graph-based parsers (McDonald 2006; Zheng 2017) utilize a certain algorithm to find the best tree based on a scoring function over candidate dependency trees, which is formulated as the sum of arc scores. Transition-based parsers (Nivre 2004; 2008) view parsing as a sequence of transitions to incrementally build a dependency tree, with each transition selected by a classifier at each step during the parsing process. Specifically, graph-based parsers are globally trained, taking into account the first-order or high-order features, which are defined on the graph or subgraph for later decoding (Mcdonald, Crammer, and Pereira 2005). But those features are defined over a limited history of parsing decisions. On the contrary, transition-based ones are trained locally and make use of greedy inference algorithms, while it can define features over a rich history of parsing decisions. In a word, the features for two paradigms are quite different.



Deep neural networks are introduced in feature selection these years to generate relevant representations automatically (Chen and Manning 2014). (Dyer et al. 2015) firstly employed Recurrent Neural Networks for dependency parsing, since the recurrent structure is good at bridging long time lags between relative inputs, which will be of great superiority in looking for long-distance dependencies (Kiperwasser and Goldberg 2016b; Cheng et al. 2016; Dozat and Manning 2016; Zhang, Cheng, and Lapata 2017). However, Recurrent Neural Networks may suffer from the trouble that inputs in early time steps tend to have less influence after a long distance on the final outputs. For the purpose of solving this problem, the attention mechanism has been proposed.

The attention mechanism has gained popularity for its ability to learn alignments between two different modularities (Luong, Pham, and Manning 2015). It was firstly introduced in the area of machine translation (Bahdanau, Cho, and Bengio 2014), learning to align words between the source language and the target one. It has later been applied in dialog generation (Shang, Lu, and Li 2015) by automatically align information in the response with those in the post, and applied in reading comprehension (Seo et al. 2016) by aligning keywords in the query with those in the context to find the answer.

However, unlike encoder-decoder models such as machine translation and dialog generation, dependency parsing is quite another task. A key difference is that in encoder-decoder tasks, there exists a strong **one-to-one** relationship between the source modularity and the target one, measured by semantic relatedness (often similarity) between two words. In dependency parsing, the mapping from a certain word towards the important information to pay attention to may be **one-to-many**. In other words, during the search for the head of a word, many different parts of the sentence should be taken into consideration. Sequential order among these parts is important.

In this work, our mechanism collects helpful information squentially, unlike the parallel style in vanilla attention. During the sequence-preserved procedure, the word to attend is first provided as initial information to generate a prior representation; then a sequence of words to be paid attention to is processed in order. The attention score and attentional representation at each time step is dependent on those at the

previous step, thus preserving the sequence information.

The sequence-preserved attention mechanism could help improve the performance of both graph-based and transition-based parsers. Generally speaking, features for graph-based parsers are not expressive enough, and features for transition-based parsers tend to suffer from the lack of global information. Word representations with rich non-local features (Zhang and Nivre 2011) may help gain an increase in accuracy of both of them, which is where the sequence-preserved attention can help.

To be specific, this mechanism tries to encode each word in a sentence with recurrent neural networks and sequence-preserved attention. The parser firstly encodes each word into an context-specific representation with bi-directional recurrent neural networks; then it tries to catch information closely related to a certain word from the context with another bi-directional recurrent neural network. This approach is inspired by the procedure that someone first scans a sequence to grasp the overall meaning, and then in order to pick the head of a word, he will scan the sequence forth and back around it again to collect context information. We use the feature representations to train a graph-based parser and a transition-based parser respectively. Results show that our feature representation helps the transition-based parser achieve state-of-the-art scores, while the graph-based parser can also benefit a lot to achieve close to state-of-the-art results in graph-based approaches. We call this model Sequence-Preserved Attention Dependency Parser, which we will refer to as SPA-DP later[1]. Our contributions can be listed as follows: firstly, we attempt to find out better word representations for a parsing framework; secondly, we improve the attention mechanism for dependency parsing, which keeps simplicity and efficiency at the same time.

## Related Work

**Dependency Parsing** The model is closely related to (Kiperwasser and Goldberg 2016b), proposing a general mechanism to learn representations for dependency parsing in two paradigms: graph-based and transition-based parsing. One significant difference is that we treat graph-based parsing as a procedure of greedy head selection, like (Zhang, Cheng, and Lapata 2017) and (Dozat and Manning 2016), rather than max-spanning-tree generation from a score matrix (McDonald 2006; Zheng 2017), because our experiments indicate that the former method can outperform the latter.

For transition-based dependency parsing, this work differs a lot from recent work in that our work learns how to integrate rich global features into local word representations (Zhang and Nivre 2011), still leaving transition decisions in a greedy way. This method keeps simplicity and efficiency at the same time, while most recent work focuses more on how to correct the error made by locally-optimal decisions. For example, recent researchers have taken great efforts to overcome the shortcomings of greedy search by employing global learning mechanism, such as designing dynamic oracles or more robust transition systems (Goldberg and Nivre 2012; Qi and Manning 2017), integrating with beam search (Zhang and Nivre 2012; Zhou et al. 2015), utilizing structured learning (Weiss et al. 2015; Vaswani and Sagae 2016), augmenting data (Alberti, Weiss, and Petrov 2015), keeping deeper parsing history (Dyer et al. 2015), minimizing conditional-random-field objective (Andor et al. 2016), etc. Experiments manifest that our model can surpass all above models by learning global representations for each word in a sentence, which can still be combined with globally optimized decision learning methods.

Apart from above two mainstream parsing methodologies, some other variants have been introduced. An easy-first approach tries to build the parse tree in a bottom-to-up way (Goldberg and Elhadad 2010; Kiperwasser and Goldberg 2016a). Several ensemble methods are proposed to overcome the shortcomings of both graph and transition based approaches, say adding parsing results of a parser as guide information to another (Nivre and McDonald 2008; Zhang and Clark 2008; Bohnet and Kuhn 2012), selecting the best parsing tree by re-ranking mechanism (Zhu et al. 2015), etc.

**Attention Mechanism** The attention mechanism was firstly introduced in the area of machine translation (Bahdanau, Cho, and Bengio 2014) and has been successfully applied in dialog generation (Shang, Lu, and Li 2015), reading comprehension (Seo et al. 2016), etc.

Our model is mostly close to the work of (Cheng et al. 2016), which first applied the attention mechanism to dependency parsing. However, our approach can be distinguished from it in three aspects: firstly, our work focuses on learning global representation of words, while theirs employs attention to model parsing history; secondly, their work forces word representations to select the candidate heads from two directions and then vote the best (they call this "bi-directional agreement"), while our work selects the head coherently by the bi-directional word representation, more fast and accurate accordingly; last but not least, our work can be applied to transition based parsing as well, while theirs cannot.

# Bi-RNN Based Dependency Parsing

In dependency parsing, recurrent neural networks have been proven useful to bridge long time lags between relative inputs, which will be of great superiority in finding out long-distance dependencies. The feature of each word is modeled by bi-directional recurrent neural networks in this section. The feature representations are then applied to a transition-based parser and a graph-based parser respectively.

## Word Feature Representations

Considering a fixed-sized word dictionary $\mathcal{D}$, the vector representations are stored in a word embedding matrix $E^{word} \in \mathbb{R}^{d^{word} \times |\mathcal{D}|}$, where $d^{word}$ is the dimensionality of the vector space and $|\mathcal{D}|$ is the size of the dictionary. Analogously, the part-of-speech(POS) tags are also mapped to a $d^{pos}$ dimensional vector space represented by $E^{pos} \in \mathbb{R}^{d^{pos} \times |\mathcal{P}|}$, where

[1]The source code of our parser is available at https://github.com/dugu9sword/spa-dp

Table 1: Special Configurations of An Arc-Standard System

| Components | Initial State | Terminal State |
|---|---|---|
| $\sigma$ | $[\omega_0]$ | $[\omega_0]$ |
| $\beta$ | $[\omega_1, \omega_2, ..., \omega_N]$ | $\emptyset$ |
| $A$ | $\emptyset$ | $A_c$ |

$d^{pos}$ is the dimensionality of the vector space and $|\mathcal{P}|$ is the size of the dictionary.

Let $S = \omega_{0:N}$ denote a sentence of length $N$, with $\omega_0$ denoting the artificial $ROOT$ node. We use $T = t_{0:N}$ as the token embedding, with $t_i$ represents $\omega_i$. The token embedding is computed as:

$$t_i = E^{word} e_i^{word} \oplus E^{pos} e_i^{pos} \tag{1}$$

where $e_i^{word}$ and $e_i^{pos}$ are the one-hot representation for the $i$-th word and its part-of-speech tag, $E^{word}$ and $E^{pos}$ are both parameters to be learned which store the contiguous embedding for corresponding feature, $\oplus$ is the concatenation operation.

In our case, the recurrent neural networks (RNN) is abstracted as a parameterized function $\text{RNN}_\theta(x_{1:n})$ or $\text{RNN}_\theta(x_1, x_2, ...x_n)$ mapping a sequence of $n$ input vectors $x_{1:n}$ to a sequence of $n$ output vectors $o_{1:n}$. Each output vector $o_i$ can be thought of as a summary of $x_{1:i}$ since it is conditioned on them.

We first use a forward RNN ($\text{RNN}^F$) to process the sentence from left to right and then use a backward RNN ($\text{RNN}^B$) to process from right to left:

$$\begin{aligned} h_i^F &= \text{RNN}^F(\overrightarrow{t_{0:i}}) \\ h_i^B &= \text{RNN}^B(\overleftarrow{t_{i:N}}) \end{aligned} \tag{2}$$

Then each word $\omega_i$ in the sentence can be represented by concatenating the representation from the bi-directional RNNs:

$$h_i = h_i^F \oplus h_i^B \tag{3}$$

## Training Objective

**Transition-Based Dependency Parsing** In a transition-based parser, the framework assumes a transition system which processes sentences and produces parse trees(Nivre 2008). Two components are defined in a transition system: a set of configurations and a set of actions which are applied to the configurations. When a sentence is being parsed, the system is initialized to a initial configuration and then a sequence of actions are taken to change the configurations repeatedly. The transition system will achieve a final configuration after a finite number of steps, and the parse tree is generated. In this paper, we examine only greedy parsing, which uses a classifier to predict the correct transition based on features extracted from the configuration.

We employ the arc-standard system (Nivre 2004) which is composed of a configuration $c = (\sigma, \beta, A)$ consisting of a stack $\sigma$, a buffer $\beta$ and a set of dependency arcs $T$. For the sequence $S = \omega_{0:N}$, the initial and final configuration is defined in table 1, and $A_c$ is returned as the parse tree.

Besides, three types of transitions are defined:

$$\begin{aligned} \text{SHIFT}[(\sigma, b_1|\beta, A)] &= (\sigma|b_1, \beta, A) \\ \text{ARC}_\mathcal{L}[(\sigma|s_2|s_1, \beta, A)] &= (\sigma|s_2, \beta, A \cup \{(s_2 \rightarrow s_1)\}) \\ \text{ARC}_\mathcal{R}[(\sigma|s_2|s_1, \beta, A)] &= (\sigma|s_1, \beta, A \cup \{(s_1 \rightarrow s_2)\}) \end{aligned}$$

where $s_1$ and $s_2$ denote the first and second words on the stack respectively and $b_1$ denotes the first word on the buffer.

Given a sentence $S = \omega_{0:N}$, the vector representations are $R = r_{0:N}$. Given a configuration $c = (\sigma, \beta, A)$, we concatenate the representation of first three words on the stack and first one word on the buffer as a representation of the configuration:

$$\phi(c) = r_{s_3} \oplus r_{s_2} \oplus r_{s_1} \oplus r_{b_1} \tag{4}$$

We build a standard neural network with one hidden layer as the classifier to choose the proper action for the configuration $c$.

$$y = W_{t2} \cdot max(0, W_{t1} \cdot \phi(c) + b_{t1}) + b_{t2} \tag{5}$$

$$P = softmax(y) \tag{6}$$

where $W_{t1} \in \mathbb{R}^{d_h \times 4 \cdot d_r}$, $b_{t1} \in \mathbb{R}^{d_h}$, $W_{t2} \in \mathbb{R}^{3 \times d_h}$ and $b_{t2} \in \mathbb{R}^3$. The final output $P$ is the probability of three different actions to choose in the current configuration.

The training objective is to minimize the cross-entropy loss as:

$$L(\theta) = -\frac{1}{|\mathcal{Q}|} \Sigma_{S \in \mathcal{Q}} \Sigma_{i=1}^{|\mathcal{C}^S|} ln P_{a_i} \tag{7}$$

where $\mathcal{Q}$ is the dataset and $\mathcal{C}^S$ is the configuration of the transition system generated from the sentence $S$, $a_i$ is the gold action for a specific configuration $\mathcal{C}_i^S$ in the sentence.

**Graph Based Dependency Parsing** We formulate our dependency parser as head selection as (Zhang, Cheng, and Lapata 2017; Dozat and Manning 2016; Hashimoto et al. 2016).

Given a sentence $S = \omega_{0:N}$, the vector representations are $R = r_{0:N}$. We define a function $s(\cdot, \cdot)$ to map a candidate word pair to a score which measures the possibility that $\omega_j$ is the head of $\omega_i$, similar to the attention mechanism defined in (Luong, Pham, and Manning 2015).

$$s(r_j, r_i) = \begin{cases} r_j^\top W r_i & \textit{bi-linear} \\ v^\top max(0, W(r_j \oplus r_i) + b) & \textit{concat} \end{cases} \tag{8}$$

where $v$, $W$, $b$ are parameters to be learned.

The probability that a word $w_j$ is the head of another word $w_i$ can be computed by normalizing the score function with respect to all possible heads:

$$P(w_j|w_i) = \frac{e^{s(r_j, r_i)}}{\sum_{k=0}^{N} e^{s(r_k, r_i)}} \tag{9}$$

The model is trained by minimizing the cross entropy of the gold head-modifier pairs in the training set:

$$L(\theta) = -\frac{1}{|\mathcal{Q}|} \sum_{S \in \mathcal{Q}} \sum_{i=1}^{|S|} ln P(r_{\mathcal{H}(S,i)}, r_i) \tag{10}$$

where $\mathcal{Q}$ is the dataset and $\mathcal{H}(S, i)$ is the index of the gold head of $i$th word in the sentence $S$. Afterwards, we use the Chu-Liu-Edmonds algorithm to make sure that the generated graph is a tree (Chu 1965).

## Sequence-Preserved Attention

We propose a sequence-preserved attention mechanism inspired by the procedure of understanding a word in a sentence composed of two steps: **SCAN** and **RESCAN**. In the **SCAN** step, a sentence is scanned to grasp the overall meaning, while in the **RESCAN** step, the sentence is rescanned from the word to its boundary, or reversely. In this section, we first give a brief introduction to vanilla attention (Bahdanau, Cho, and Bengio 2014; Luong, Pham, and Manning 2015). Then we propose sequence-preserved attention mechanism to compute feature representations of words in a sequence, where the attentional representation of each word is computed sequentially.

**The Scan Step** As in *section Bi-RNN Based Dependency Parsing*, given a sentence $S = \omega_{0:N}$, $\omega_i$ is firstly transformed to a vector representation $t_i$ . The **SCAN** step can be modeled by a bi-directional RNN to get the feature representations of a word:

$$\textsc{Scan}(t_{[0:N]})_i = \textsc{Rnn}^F(\overrightarrow{t_{0:i}}) \oplus \textsc{Rnn}^B(\overleftarrow{t_{i:N}}) \tag{11}$$

We denote the results of the **SCAN** step as $h_{0:N}$ where $h_i = \textsc{Scan}(t_{[0:N]})_i$.

**Vanilla Attention** Given a sentence $\omega_{0:N}$ and a word $\omega_i$, the attention mechanism is usually employed to select $\omega_j$, $j \in \{0, ..., N\}$ relevant to $\omega_i$. $\omega_i$ is transformed to a vector representation $h_i$ during the **SCAN** step. The relevance between any word $\omega_j$ and $\omega_i$ is measured by a *relevance score* $s_{ij}$, which is normalized to derive a probability distribution, namely *attention* $a_{ij}$, over the sentence, the procedure is illustrated in Figure 1.

$$s_{ij} = \textsc{Mlp}(h_i \oplus h_j) \tag{12}$$

$$a_{ij} = \frac{e^{s_{ij}}}{\sum_{k=0}^{N} e^{s_{ik}}} \tag{13}$$

where MLP is a multi-layer perceptron. Then the derived attention can be used to compute a summary of the words in the sentence relevant to the given word $\omega_i$ by weighted average, namely *attentional representation* $m_i$.

$$m_i = \sum_{j=0}^{N} a_{ij} \cdot h_j \tag{14}$$

The attentional representation $m_i$ is able to extract features from the memory units relevant to the query which can be applied to downstream tasks. In dependency parsing, the attentional representation can be used as the feature of a word when selecting its head.

**Sequence-Preserved Attention** Given a sequence $b_{1:N}$ and a word $k$, the sequence-preserved attention mechanism is usually employed to select $b_i$, $i \in \{1, ..., N\}$ relevant to $k$. $b_i$ is transformed to a vector representation $v_i$ during the **SCAN** step, and $k$ is transformed to a vector representation $u$. The sequence-preserved attention mechanism first processes the word $k$ to attain the initial attentional representation $m_0$, based on which the $v_{[1:N]}$ are processed sequentially to gain corresponding attentional representations.

For the $i$th word in the sequence, a local attentional representation $\tilde{m}_i$ is computed as a summary of both the current input $v_i$ and the previous attentional representation $m_{i-1}$ , the relevance score $s_i$ is computed to measure the relevance between $v_i$ and $m_{i-1}$. The relevance score $s_i$ is then normalized as the attention $a_i$, controlling the weight of $\tilde{m}_i$ and $v_i$ to be averaged. The procedure is illustrated in Figure 2.

$$s_i = \begin{cases} \mathcal{G}_\lambda(m_{i-1}, v_i) & i > 0 \\ \mathcal{G}_\lambda(\mathbf{0}, u) & i = 0 \end{cases} \tag{15}$$

$$a_i = sigmoid(s_i) \tag{16}$$

$$m_i = \begin{cases} (1 - a_i) \otimes m_{i-1} + a_i \otimes \tilde{m}_i & i > 0 \\ a_0 \otimes \tilde{m}_0 & i = 0 \end{cases} \tag{17}$$

$$\tilde{m}_i = \begin{cases} \mathcal{G}_\mu(m_{i-1}, v_i) & i > 0 \\ \mathcal{G}_\mu(\mathbf{0}, u) & i = 0 \end{cases} \tag{18}$$

where $\mathcal{G}_\lambda$ is a function to decide how much relevance score should be allocated to the specific word $b_i$, taking the last attentional representation $m_{i-1}$ into consideration; $\mathcal{G}_\mu$ (i.e. $\tilde{m}_i$) has the functionality of generating a local attentional representation at current time step, combining the current input $v_i$ with last attentional representation $m_{i-1}$. The sequence-preserved attentional representation $m_i$ is calculated as a weighted average of the last attentional representation $m_{i-1}$ and the local attentional representation $\tilde{m}_i$, where $\mathcal{G}_\lambda$ serves as the weight.

Since the attention scores and attentional representations are defined recursively, the final attentional representation of $b$ with respect to $\omega_{[0:N]}$ is $m_N$:

$$\textsc{Spa}(u, v_1, v_2, ..., v_N) = m_N \tag{19}$$

The sequence-preserved attention tries to locate the relevant words in a sentence sequentially. Coincidentally, this design fits the gated recurrent unit well(Bahdanau, Cho, and Bengio 2014). To make the definition of GRU and sequence-preserved attention match with each other explicitly, we can define $\mathcal{G}_\lambda$, $\mathcal{G}_\mu$ as follows:

$$\mathcal{G}_\tau(v_{i-1}, m_i) = sigmoid(\textsc{Mlp}_\tau(m_i \oplus v_{i-1})) \tag{20}$$

$$\mathcal{G}_\lambda(v_{i-1}, m_i) = sigmoid(\textsc{Mlp}_\lambda(m_i \oplus v_{i-1})) \tag{21}$$

$$\mathcal{G}_\mu(v_{i-1}, m_i) = \textsc{Mlp}_\mu(m_i \oplus (\mathcal{G}_\tau(v_{i-1}, m_i) \otimes v_{i-1})) \tag{22}$$

where the attentional representation $a_i$ serves as the *hidden state*, the $\mathcal{G}_\tau$ serves as a *reset gate* which controls how much information should be reset, and the attention score $s_i$ serves as a *update gate* which controls how much information from the last state shall flow into current state.

Since the GRU is similar with the sequence-preserved attention to some extent, we would like to employ GRU as an implementation of sequence-preserved attention, the attentional representation of $b$ with respect to $\omega_{[0:N]}$ can be computed by

$$\begin{aligned} m_b &= \textsc{Spa}(u, v_1, v_2, ..., v_N) \\ &= \textsc{Gru}(u, v_1, v_2, ..., v_N) \end{aligned} \tag{23}$$

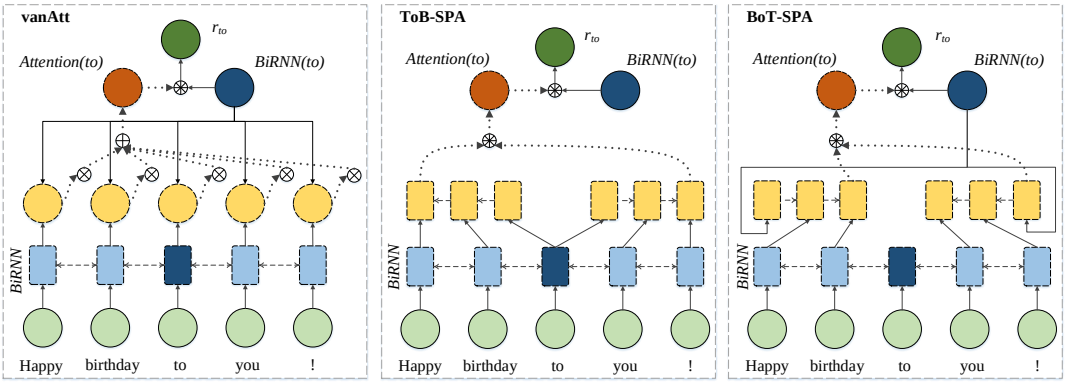


Figure 1: Architecture of three models: vanilla attention model (**vanAtt**), SPA model from target to boundary(**ToB-SPA**), and SPA model from boundary to target (**BoT-SPA**). In the figure, the $\oplus$ symbol denotes an element-wise *plus* operation, the $\otimes$ symbol denotes an element-wise *multiplication* operation, the $\circledast$ symbol denotes a *concatenation* operation. Each rectangle denotes an LSTM cell, each circle denotes a vector.

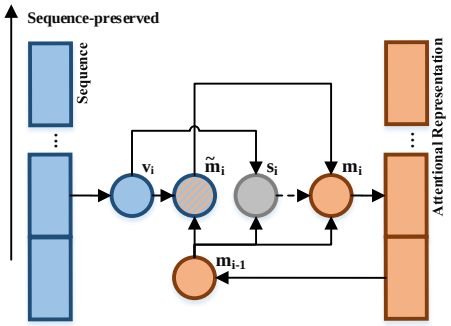


Figure 2: The procedure of calculation of sequence-preserved attention.

**The Rescan Step With Sequence-Preserved Attention** In the **RESCAN** step, the sentence is rescanned from the word $\omega$ to its boundary, or reversely. $\omega_i$ is transformed to a vector representation $h_i$ as in the former *section Bi-RNN Based Dependency Parsing*.

$$\textsc{Rescan}(h_{[0:N]})_i = \textsc{Rescan}_{\mathcal{L}}(h_i, h_{\leq i}) \oplus \textsc{Rescan}_{\mathcal{R}}(h_i, h_{\geq i}) \tag{24}$$

where $\textsc{Rescan}_{\mathcal{L}}$ denotes the procedure of scanning the left part of the word in the sentence and $\textsc{Rescan}_{\mathcal{R}}$ denotes the procedure of scanning the right part of the word in the sentence.

Imitating the two different behaviors of rescanning mode, we can derive two variants of the sequence-preserved attention, as in Figure 1:

- **Target to Boundary (ToB)** To gather valuable information around some word, **scan from the word to the boundary of the sentence**, drop useless information and fetch usable information incrementally. We will refer to this as **ToB-SPA** later. This procedure can be formulated as:

$$\begin{aligned} \textsc{Rescan}_{\mathcal{L}}(h_i, h_{\leq i}) &= \textsc{Gru}(h_i, h_{i-1}, ..., h_0) \\ \textsc{Rescan}_{\mathcal{R}}(h_i, h_{\geq i}) &= \textsc{Gru}(h_i, h_{i+1}, ..., h_N) \end{aligned} \tag{25}$$

- **Boundary to Target (BoT)** RNN has the feature that it tends to put more weight on input closer to the current time step. In dependency parsing, words closer to a certain word have more effect on head selection. So we reverse the procedure in **ToB**, **scan the sentence bi-directionally with the word as the first input**. We will refer to this procedure as **BoT-SPA** later. It can be formulated as:

$$\begin{aligned} \textsc{Rescan}_{\mathcal{L}}(h_i, h_{\leq i}) &= \textsc{Gru}(h_i, h_0, h_1, ..., h_{i-1}) \\ \textsc{Rescan}_{\mathcal{R}}(h_i, h_{\geq i}) &= \textsc{Gru}(h_i, h_N, h_{N-1}..., h_{i+1}) \end{aligned} \tag{26}$$

By concatenating the scanning results $\textsc{Scan}(t_{[0:N]})_i$ (the word feature representation) and the rescanning results $\textsc{Rescan}(h_{[0:N]})_i$ (the attentional representation), we can derive the final representation of a word:

$$r_i = \textsc{Scan}(t_{[0:N]})_i \oplus \textsc{Rescan}(h_{[0:N]})_i \tag{27}$$

which can be used as word representations in the dependency parsing as in *section Bi-RNN Based Dependency Parsing*.

## Experiments

We show the results of the models on the English Penn Tree Bank (PTB) and the Chinese Penn Tree Bank (CTB). We follow the standard split of PTB, with the section 2-21 for training, section 22 for development and section 23 for testing. For English PTB, the POS tags are generated from the Stanford POS tagger (Toutanova et al. 2003) while for CTB, gold word segmentation and POS tags are used.

### Implementations And Hyper-Parameters Choice

The model is implemented with the PyTorch[2] deep learning framework. All experiments were run on a computer

[2]https://pytorch.org

Table 2: Parsing With Different Attention Mechanism

| | Model | UAS |
|---|---|---|
| Transition | BiRNN | 94.02 |
| | BiRNN + vanilla attention | 94.33(+0.31) |
| | BiRNN + ToB-SPA | 94.25(+0.23) |
| | BiRNN + BoT-SPA | **94.72** (+0.70) |
| Graph | BiRNN | 93.80 |
| | BiRNN + vanilla attention | 94.30(+0.50) |
| | BiRNN + ToB-SPA | 94.50(+0.70) |
| | BiRNN + BoT-SPA | **94.79**(+0.99) |

equipped with an Intel Core i7 processor, an 8GB RAM and a NVIDIA GTX-1070 GPU.

Hyper parameters are fine tuned on the PTB 3.3.0 development set. In terms of the RNN unit for word feature representation, we find that the LSTM cell can outperform the GRU cell in its stronger modeling ability. For graph-based parser, the hidden size of the bi-RNN is set to 400, and the hidden size of RNN for sequence-preserved attention is set to 100; for transition-based parser, the hidden size of both the bi-RNN and the RNN for sequence-preserved attention is set to 700.

Considering different kinds of pre-trained word embeddings, with the Glove (Pennington, Socher, and Manning 2014) word-embedding the neural networks can converge better than CBoW and Skip-gram (Mikolov et al. 2013). The dimension of word vectors is 300 and the dimension of tag vectors is 50. We train the model with word embedding and tag embedding drop out set to 30% respectively.

In terms of the score function in graph based parser, experiments show that the *bi-linear* mode can outperform the *concat* mode.

The model is trained with a batch size of 32 by an Adam optimizer(Kingma and Ba 2014).

## Results

We report the UAS results of simple bi-RNN, vanilla attention model, ToB-SPA and BoT-SPA on English PTB in Table 2. Results show that the vanilla attention model can beat the simple bi-RNN model, and our BoT-SPA model can outperform almost all other models.

We also report the results on the English PTB and Chinese CTB in Table 3 and 4 respectively.

Our parser can reach state-of-the-art results in transition-based parsers. It is worth noting that although the state-of-the-art parser in English PTB (Kuncoro et al. 2016) can reach an accuracy of neatly 95.80%, their work shall be considered as a framework that is not equal with other transition-based ones since it has access to original phrase structure annotation where conjunctions can be parsed more correctly. Thus we think their work should not be regarded as one competitor of ours, and our work exceeds all the other parsers in English PTB.

In graph-based dependency parsing, our work can also beat most former work when keeping simplicity and efficiency.

Table 3: Results On the English PTB Dataset

| | Model | UAS | LAS |
|---|---|---|---|
| Transition | (Chen and Manning 2014) | 92.00 | 91.80 |
| | (Zhu et al. 2015) | 94.16 | - |
| | (Kiperwasser and Goldberg 2016b) | 93.90 | 91.90 |
| | (Andor et al. 2016) | 94.61 | 92.79 |
| | SPA-DP | **94.72** | 92.57 |
| Graph | (Kiperwasser and Goldberg 2016b) | 93.00 | 90.90 |
| | (Hashimoto et al. 2016) | 94.67 | 92.90 |
| | (Zhang, Cheng, and Lapata 2017) | 94.10 | 91.90 |
| | (Dozat and Manning 2016) | **95.74** | 94.08 |
| | (Zheng 2017) | 95.53 | 93.94 |
| | SPA-DP | 94.79 | 92.61 |

Table 4: Results On the Chinese CTB Dataset

| | Model | UAS | LAS |
|---|---|---|---|
| Transition | (Chen and Manning 2014) | 83.90 | 82.40 |
| | (Andor et al. 2016) | 84.72 | 80.85 |
| | (Ballesteros et al. 2016) | 87.65 | 86.21 |
| | (Kiperwasser and Goldberg 2016b) | 87.60 | 86.10 |
| | SPA-DP | **88.15** | 85.51 |
| Graph | (Kiperwasser and Goldberg 2016b) | 87.10 | 85.50 |
| | (Cheng et al. 2016) | 88.10 | 85.70 |
| | (Dozat and Manning 2016) | 89.30 | 88.23 |
| | (Zheng 2017) | **89.42** | 87.94 |
| | SPA-DP | 88.04 | 85.40 |

## Error Analysis

To characterize the errors made by parsers and the enhancement in accuracy by importing sequence-preserved attention, we present some analysis on the accuracy with respect to the sentence length and linguistic factors such as part-of-speech tags. All analysis are conducted on the unlabeled attachment results from the English PTB testing set with a graph-based parser.

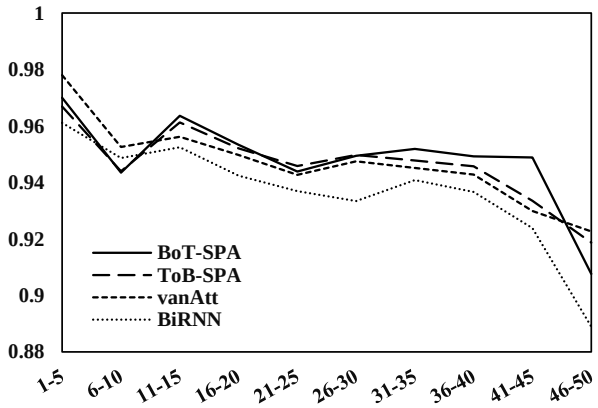


Figure 3: Accuracy with respect to length

**Length Factors** It is well known that parsers tend to have lower accuracies for long-distance dependency. Figure 3 shows the accuracies of different models with respect to sen-

tence length. It is obvious that a dependency parser adopting attention mechanism outperforms the baseline model for a margin increase in accuracy.

In general, the SPA model is able to gain better results for most sentences than those of the vanilla attention model, while the BoT-SPA model is better than ToB-SPA, since in a sentence, the dependency between two words is affected more by the closest words than those far away. The ToB-SPA model uses an RNN to "read" words into the distance, most of which are irrelevant and may serve more as noise.

Conspicuously, the accuracy of BoT-SPA model stays the high level and even increase a bit as the sequence extends from 25 to 50, when others' drastically drop. This phenomenon shows that our sequence-preserved attention mechanism is good at grasping long-distance relationship.

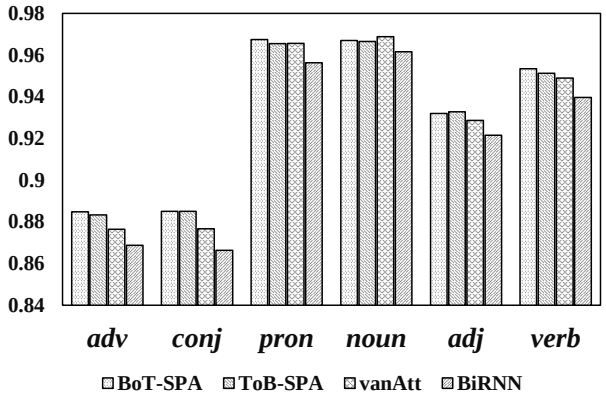


Figure 4: Accuracy with respect to POS tags

**Linguistic Factors** We distinguish *nouns*, *verbs*, *pronouns*, *adjectives*, *adverbs*, *conjunctions* like (McDonald and Nivre 2007). We count the accuracy with respect to different POS-tags on the testing set and compare that of four parsers as in Figure 4.

It can be concluded that the BoT-SPA model can outperform almost all other models on most categories.

- **Nouns & Pronouns** Typically in a sentence, the nouns and pronouns are attached to verbs and lower in the parse tree, thus tend to be easy to find out the correct head. All four parsers can perform well but the model with vanilla attention or sequence-preserved attention can still perform surpass the baseline model.
- **Verbs** In a parse tree, the verb tends to be closer to the root thus has longer-distance dependency than nouns and pronouns, which makes it a little bit more difficult to parse.
- **Adjectives** Adjectives are the furthest from the root node and have few siblings, thus theoretically should be the easiest to parse, but the results are rather opposite. One possible explanation is that graph-based parser is generally globally normalized, and more attention is paid to long distance dependency, causing a decrease in accuracy for short distance dependency.
- **Conjunctions & Adverbs** In a sentence, adverbs and conjunctions often have long dependency lengths, therefore all four parsers have rather pool performance on them. Although the head detection is difficult, explicit increase in the accuracy can be shown from the result. The SPA model gains an improvement in accuracy than the BiRNN baseline for about 1.37%, and exceeds the vanilla attention for about 0.44%. This can be viewed as a strong evidence that sequence-preserved attention has the ability to model long distance dependency than the vanilla one. Without extra information, the vanilla attention may be trapped in the dilemma to choose head under the circumstance of the long distance.

## Conclusions

We have described an RNN-based sequence-preserved attention method to capture the global and relevant information for each word of an input sentence being parsed. The proposed method tries to overcome the shortcoming of vanilla attention mechanism that neglects the relative order in which the features occur. For each parse unit, the sequence-preserved attention is able to capture the rich and global features, being sensitive to the order of the words, from an input sentence. The results of experiments show that the features extracted by the sequence-preserved attention benefit both the graph-based and transition-based parsing systems with significant improvements across two different languages.

## Acknowledgments

The authors would like to thank the anonymous reviewers for their valuable comments. This work was partly supported by a grant from Shanghai Municipal Natural Science Foundation (No. 15511104303).

## References

Alberti, C.; Weiss, D.; and Petrov, S. 2015. Improved transition-based parsing and tagging with neural networks. *Conference on Empirical Methods on Natural Language Processing*.

Andor, D.; Alberti, C.; Weiss, D.; Severyn, A.; Presta, A.; Ganchev, K.; Petrov, S.; and Collins, M. 2016. Globally normalized transition-based neural networks. *arXiv preprint arXiv:1603.06042*.

Bahdanau, D.; Cho, K.; and Bengio, Y. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Ballesteros, M.; Goldberg, Y.; Dyer, C.; and Smith, N. A. 2016. Training with exploration improves a greedy stack-lstm parser. *arXiv preprint arXiv:1603.03793*.

Bohnet, B., and Kuhn, J. 2012. The best of both worlds: A graph-based completion model for transition-based parsers. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, EACL '12.

Chen, D., and Manning, C. 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.

Cheng, H.; Fang, H.; He, X.; Gao, J.; and Deng, L. 2016. Bidirectional attention with agreement for dependency parsing. *arXiv preprint arXiv:1608.02076*.

Chu, Y.-J. 1965. On the shortest arborescence of a directed graph. *Science Sinica*.

Dozat, T., and Manning, C. D. 2016. Deep biaffine attention for neural dependency parsing. *arXiv preprint arXiv:1611.01734*.

Dyer, C.; Ballesteros, M.; Ling, W.; Matthews, A.; and Smith, N. A. 2015. Transition-based dependency parsing with stack long short-term memory. *arXiv preprint arXiv:1505.08075*.

Goldberg, Y., and Elhadad, M. 2010. An efficient algorithm for easy-first non-directional dependency parsing. In *NAACL*. Association for Computational Linguistics.

Goldberg, Y., and Nivre, J. 2012. A dynamic oracle for arc-eager dependency parsing. In *COLING*.

Hashimoto, K.; Xiong, C.; Tsuruoka, Y.; and Socher, R. 2016. A joint many-task model: Growing a neural network for multiple nlp tasks. *arXiv preprint arXiv:1611.01587*.

Kingma, D., and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kiperwasser, E., and Goldberg, Y. 2016a. Easy-first dependency parsing with hierarchical tree lstms. *arXiv preprint arXiv:1603.00375*.

Kiperwasser, E., and Goldberg, Y. 2016b. Simple and accurate dependency parsing using bidirectional lstm feature representations. *arXiv preprint arXiv:1603.04351*.

Kuncoro, A.; Ballesteros, M.; Kong, L.; Dyer, C.; Neubig, G.; and Smith, N. A. 2016. What do recurrent neural network grammars learn about syntax? *arXiv preprint arXiv:1611.05774*.

Luong, M.-T.; Pham, H.; and Manning, C. D. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.

McDonald, R., and Nivre, J. 2007. Characterizing the errors of data-driven dependency parsing models. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*.

Mcdonald, R. T.; Crammer, K.; and Pereira, F. 2005. Online large-margin training of dependency parsers. *Proceedings of the 43rd Annual Meeting of the ACL*.

McDonald, R. 2006. Discriminative training and spanning tree algorithms for dependency parsing. *Ph. D. Thesis, Computer and Information Science, University of Pennsylvania*.

Mikolov, T.; Chen, K.; Corrado, G.; and Dean, J. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Nivre, J., and McDonald, R. T. 2008. Integrating graph-based and transition-based dependency parsers. In *ACL*.

Nivre, J. 2004. Incrementality in deterministic dependency parsing. In *Proceedings of the Workshop on Incremental Parsing: Bringing Engineering and Cognition Together*. Association for Computational Linguistics.

Nivre, J. 2008. Algorithms for deterministic incremental dependency parsing. *Computational Linguistics*.

Pennington, J.; Socher, R.; and Manning, C. 2014. Glove: Global vectors for word representation. In *EMNLP*.

Qi, P., and Manning, C. D. 2017. Arc-swift: A novel transition system for dependency parsing. *arXiv preprint arXiv:1705.04434*.

Seo, M.; Kembhavi, A.; Farhadi, A.; and Hajishirzi, H. 2016. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603*.

Shang, L.; Lu, Z.; and Li, H. 2015. Neural responding machine for short-text conversation. *arXiv preprint arXiv:1503.02364*.

Toutanova, K.; Klein, D.; Manning, C. D.; and Singer, Y. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*. Association for Computational Linguistics.

Vaswani, A., and Sagae, K. 2016. Efficient structured inference for transition-based parsing with neural networks and error states. *Transactions of the Association for Computational Linguistics*.

Weiss, D.; Alberti, C.; Collins, M.; and Petrov, S. 2015. Structured training for neural network transition-based parsing. *arXiv preprint arXiv:1506.06158*.

Zhang, Y., and Clark, S. 2008. A tale of two parsers: investigating and combining graph-based and transition-based dependency parsing using beam-search. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Zhang, Y., and Nivre, J. 2011. Transition-based dependency parsing with rich non-local features. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Short Papers - Volume 2*, HLT '11. Association for Computational Linguistics.

Zhang, Y., and Nivre, J. 2012. Analyzing the effect of global learning and beam-search on transition-based dependency parsing. In *COLING (Posters)*.

Zhang, X.; Cheng, J.; and Lapata, M. 2017. Dependency parsing as head selection. *conference of the european chapter of the association for computational linguistics*.

Zheng, X. 2017. Incremental graph-based neural dependency parsing. *Conference on Empirical Methods on Natural Language Processing*.

Zhou, H.; Zhang, Y.; Huang, S.; and Chen, J. 2015. A neural probabilistic structured-prediction model for transition-based dependency parsing. In *ACL*.

Zhu, C.; Qiu, X.; Chen, X.; and Huang, X. 2015. A reranking model for dependency parser with recursive convolutional neural network. *arXiv preprint arXiv:1505.05667*.