

# Bi-directional Attention with Agreement for Dependency Parsing

Hao Cheng Hao Fang  
University of Washington  
{chenghao, hfang}@uw.edu

Xiaodong He Jianfeng Gao Li Deng  
Microsoft Research  
{xiaohe, jfgao, deng}@microsoft.com

## Abstract

We develop a novel bi-directional attention model for dependency parsing, which learns to agree on headword predictions from the forward and backward parsing directions. The parsing procedure for each direction is formulated as sequentially querying the memory component that stores continuous headword embeddings. The proposed parser makes use of *soft* headword embeddings, allowing the model to implicitly capture high-order parsing history without dramatically increasing the computational complexity. We conduct experiments on English, Chinese, and 12 other languages from the CoNLL 2006 shared task, showing that the proposed model achieves state-of-the-art unlabeled attachment scores on 6 languages.<sup>1</sup>

## 1 Introduction

Recently, several neural network models have been developed for efficiently accessing long-term memory and discovering dependencies in sequential data. The memory network framework has been studied in the context of question answering and language modeling (Weston et al., 2015; Sukhbaatar et al., 2015), whereas the neural attention model under the encoder-decoder framework has been applied to machine translation (Bahdanau et al., 2015) and constituency parsing (Vinyals et al., 2015b). Both frameworks learn the latent alignment between the source and target sequences, and the mechanism of

attention over the *encoder* can be viewed as a soft operation on the *memory*. Although already used in the encoder for capturing global context information (Bahdanau et al., 2015), the bi-directional recurrent neural network (RNN) has yet to be employed in the decoder. Bi-directional decoding is expected to be advantageous over the previously developed uni-directional counterpart, because the former exploits richer contextual information. Intuitively, we can use two separate uni-directional RNNs where each one constructs its respective attended encoder context vectors for computing RNN hidden states. However, the drawback of this approach is that the decoder would often produce different alignments resulting in discrepancies for the forward and backward directions. In this paper, we design a training objective function to enforce attention agreement between both directions, inspired by the alignment-by-agreement idea from Liang et al. (2006). Specifically, we develop a dependency parser (BiAtt-DP) using a bi-directional attention model based on the memory network. Given that the golden alignment is observed for dependency parsing in the training stage, we further derive a simple and interpretable approximation for the agreement objective, which makes a natural connection between the latent and observed alignment cases.

The proposed BiAtt-DP parses a sentence in a linear order via sequentially querying the memory component that stores continuous embeddings for all headwords. In other words, we consider all possible arcs during the parsing. This formulation is adopted by graph-based parsers such as the MST-Parser (McDonald et al., 2005). The consideration

<sup>1</sup>Our software and models are available at <https://github.com/hao-cheng/biattdp>.

of all possible arcs makes the proposed BiAtt-DP different from many recently developed neural dependency parsers (Chen and Manning, 2014; Weiss et al., 2015; Alberti et al., 2015; Dyer et al., 2015; Ballesteros et al., 2015), which use a transition-based algorithm by modeling the parsing procedure as a sequence of actions on buffers. Moreover, unlike most graph-based parsers which may suffer from high computational complexity when utilizing high-order parsing history (McDonald and Pereira, 2006), the proposed BiAtt-DP can implicitly inject such information into the model while keeping the computational complexity in the order of  $\mathcal{O}(n^2)$  for a sentence with  $n$  words. This is achieved by feeding the RNN in the query component with a *soft* headword embedding, which is computed as the probability-weighted sum of all headword embeddings in the memory component.

To the best of our knowledge, this is the first attempt to apply memory network models to graph-based dependency parsing. Moreover, it is the first extension of neural attention models from unidirection to multi-direction by enforcing agreement on alignments. Experiments on English, Chinese, and 12 languages from the CoNLL 2006 shared task show the BiAtt-DP can achieve competitive parsing accuracy with several state-of-the-art parsers. Furthermore, our model achieves the highest unlabeled attachment score (UAS) on Chinese, Czech, Dutch, German, Spanish and Turkish.

## 2 A MemNet-based Dependency Parser

The proposed parser first encodes each word in a sentence by continuous embeddings using a bi-directional RNN, and then performs two types of operations, *i.e.* 1) headword predictions based on bi-directional parsing history and 2) the relation prediction conditioned on the current modifier and its *predicted* headword both in the embedding space. In the following, we first present how the token embeddings are constructed. Then, the key components of the proposed parser, *i.e.* the memory component and the query component, are discussed in detail. Lastly, we describe the parsing algorithm using a bi-directional attention model with agreement.

### 2.1 Token Embeddings

In the proposed BiAtt-DP, the memory and query components share the same token embeddings. We use the notion of additive token embedding as in (Botha and Blunsom, 2014) to utilize the available information about the token, *e.g.*, its word form, lemma, part-of-speech (POS) tag, and morphological features. Specifically, the token embedding is computed as

$$\mathbf{E}^{\text{form}} \mathbf{e}_i^{\text{form}} + \mathbf{E}^{\text{pos}} \mathbf{e}_i^{\text{pos}} + \mathbf{E}^{\text{lemma}} \mathbf{e}_i^{\text{lemma}} + \dots,$$

where  $\mathbf{e}_i$ 's are one-hot encoding vectors for the  $i$ -th word, and  $\mathbf{E}$ 's are parameters to be learned that store the continuous embeddings for corresponding feature. Note those one-hot encoding vectors have different dimensions, depending on individual vocabulary sizes, and all  $\mathbf{E}$ 's have the same first dimension but different second dimension. The additive token embeddings allow us to easily integrate a variety of information. Moreover, we only need to make a single decision on the dimensionality of the token embedding, rather than a combination of decisions on word embeddings and POS tag embeddings as in concatenated token embeddings used by Chen and Manning (2014), Dyer et al. (2015) and Weiss et al. (2015). It reduces the number of model parameters to be tuned, especially when lots of different features are used. In our experiments, the word form and fine-grained POS tag are always used, whereas other features are used depending on their availability in the dataset. All singleton words, lemmas, and POS tags are replaced by special tokens.

The additive token embeddings are transformed into another space before they are used by the memory and query components, *i.e.*

$$\mathbf{x}_i = \text{LReLU} \left[ \mathbf{P} \left( \mathbf{E}^{\text{form}} \mathbf{e}_i^{\text{form}} + \dots \right) \right],$$

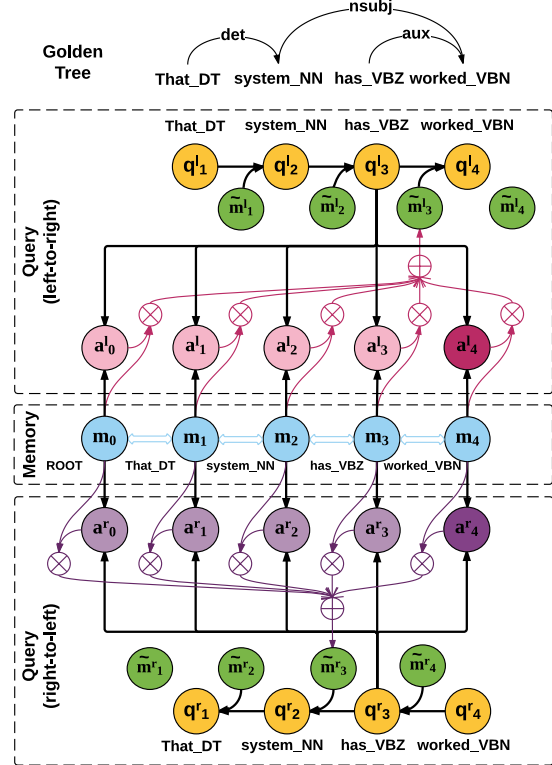
where  $\mathbf{P}$  is the projection matrix and is shared by the memory and query components as well. The activation function of this projection layer is the leaky rectified linear (LReLU) function (Mass et al., 2013) with 0.1 as the slope of the negative part. In the remaining part of the paper, we refer to  $\mathbf{x}_i \in \mathbb{R}^p$  as the token embedding for word at position  $i$ . Note the subscript  $i$  is substituted by  $j$  and  $t$  for the memory and query components, respectively.

## 2.2 Components

As shown in Figure 1, the proposed BiAtt-DP has three components, *i.e.* a memory component, a left-to-right query component, and a right-to-left query component. Given a sentence of length  $n$ , the parser first uses a bi-directional RNN to construct  $n + 1$  headword embeddings,  $\mathbf{m}_0, \mathbf{m}_1, \dots, \mathbf{m}_n \in \mathbb{R}^e$ , with  $\mathbf{m}_0$  reserved for the ROOT symbol. Each query component is an uni-directional attention model. In a query component, a sequence of  $n$  modifier embeddings  $\mathbf{q}_1, \dots, \mathbf{q}_n \in \mathbb{R}^d$  are constructed recursively by conditioning on all headword embeddings. To address the vanishing gradient issue in RNNs, we use the gated recurrent unit (GRU) proposed by Cho et al. (2014), where an update gate and a reset gate are employed to control the information flow. We replace the hyperbolic tangent function in GRU with the LReL function, which is faster to compute and achieves better parsing accuracy in our preliminary studies. In the following, we refer to headword and modifier embeddings as memory and query vectors, respectively.

**Memory Component:** The proposed BiAtt-DP uses a bi-directional RNN to obtain the memory vectors. At time step  $j$ , the current hidden state vector  $\mathbf{h}_j^l \in \mathbb{R}^{e/2}$  (or  $\mathbf{h}_j^r \in \mathbb{R}^{e/2}$ ) is computed as a non-linear transformation based on the current input vector  $\mathbf{x}_j$  and the previous hidden state vector  $\mathbf{h}_{j-1}^l$  (or  $\mathbf{h}_{j+1}^r$ ), *i.e.*  $\mathbf{h}_j^l = \text{GRU}(\mathbf{h}_{j-1}^l, \mathbf{x}_j)$  (or  $\mathbf{h}_j^r = \text{GRU}(\mathbf{h}_{j+1}^r, \mathbf{x}_j)$ ). Ideally, the recursive nature of the RNN allows it to capture all context information from one-side, and a bi-directional RNN can thus capture context information from both sides. We concatenate the hidden layers of the left-to-right RNN and the right-to-left RNN for the word at position  $j$  as the memory vector  $\mathbf{m}_j = [\mathbf{h}_j^l; \mathbf{h}_j^r]$ . These memory vectors are expected to encode the words and their context information in the headword space.

**Query Component:** For each query component, we use a single-directional RNN with GRU to obtain the query vectors  $\mathbf{q}_j$ 's, which are the hidden state vectors of the RNN. Each  $\mathbf{q}_t$  is used to query the memory component, returning association scores  $s_{t,j}$ 's between the word at position  $t$  and the head-



**Figure 1:** The structure of the BiAtt-DP. The figure only illustrates the parsing process at the time step for *has*. Blue and yellow circles are memory and query vectors, respectively. Red and purple circles represent headword probabilities predicted from corresponding query components. Green circles represent soft headword embeddings. Black arrowed lines are connections carrying weight matrices.  $\otimes$  and  $\oplus$  indicate element-wise multiplication and addition, respectively. For simplicity, we ignore the token embedding  $\mathbf{x}_t$  connected to the RNN hidden layers  $\mathbf{m}_j$ ,  $\mathbf{q}_t^l$  and  $\mathbf{q}_t^r$ .

word at position  $j$  for  $j \in \{0, \dots, n\}$ , *i.e.*

$$s_{t,j} = \mathbf{v}^T \phi(\mathbf{C}\mathbf{m}_j + \mathbf{D}\mathbf{q}_t), \quad (1)$$

where  $\phi(\cdot)$  is the element-wise hyperbolic tangent function, and  $\mathbf{C} \in \mathbb{R}^{h \times e}$ ,  $\mathbf{D} \in \mathbb{R}^{h \times d}$  and  $\mathbf{v} \in \mathbb{R}^h$  are model parameters. Then, we can obtain probabilities (aka attention weights),  $a_{t,0}, \dots, a_{t,n}$ , over all headwords in the sentence by normalizing  $s_{t,j}$ 's, using a softmax function

$$\mathbf{a}_t = \text{softmax}(\mathbf{s}_t). \quad (2)$$

The *soft* headword embedding is then defined as  $\tilde{\mathbf{m}}_t = \sum_{j=1}^n a_{t,j} \mathbf{m}_j$ . At each time step  $t$ , the

RNN takes the *soft* headword embedding  $\tilde{\mathbf{m}}_{t-1}^l$  or  $\tilde{\mathbf{m}}_{t+1}^r$  as the input, in addition to the token embedding  $\mathbf{x}_t$ . Formally, for the forward case, the  $\mathbf{q}_t$  can be computed as  $\mathbf{q}_t = \text{GRU}(\mathbf{q}_{t-1}, [\tilde{\mathbf{m}}_t; \mathbf{x}_t])$ . Although the RNN is able to capture long-span context information to some extent, the local context may very easily dominate the hidden state. Therefore, this additional soft headword embedding allows the model to access long-span context information in a different channel. On the other hand, by recursively feeding both the query vector and the soft headword embedding into the RNN, the model *implicitly* captures high-order parsing history information, which can potentially improve the parsing accuracy (Yamada and Matsumoto, 2003; McDonald and Pereira, 2006). However, for a graph-based dependency parser, utilizing parsing history features is computationally expensive. For example, an  $k$ -th order MSTParser (McDonald and Pereira, 2006) has  $\mathcal{O}(n^{k+1})$  complexity for a sentence of  $n$  words. In contrast, the BiAtt-DP *implicitly* captures high-order parsing history while keeping the complexity in the order of  $\mathcal{O}(n^2)$ , *i.e.* for each direction. we compute  $n(n+1)$  pair-wise probabilities  $a_{t,j}$  for  $t = 1, \dots, n$  and  $j = 0, \dots, n$ .

In this paper, we choose to use soft headword embeddings rather than making hard decisions on headwords. In the latter case, beam search may potentially improve the parsing accuracy at the cost of higher computational complexity, *i.e.*  $\mathcal{O}(Bn^2)$  with a beam width of  $B$ . When using soft headword embeddings, there is no need to perform beam search. Moreover, it is straightforward to incorporate parsing history from both directions by using two query components at the cost of  $\mathcal{O}(2n^2)$ , which cannot be easily achieved when using beam search. The parsing decision can be made directly based on attention weights from the two query components or further rescored by the maximum spanning tree (MST) search algorithm.

### 2.3 Parsing by Attention with Agreement

For the bi-directional attention model, the underlying probability distributions  $\mathbf{a}_t^l$  and  $\mathbf{a}_t^r$  may not agree with each other. In order to encourage the agreement, we use the mathematically convenient metric, *i.e.* the squared Hellinger distance  $H^2(\mathbf{a}_t^l || \mathbf{a}_t^r)$ , for quantifying the distance between these two distri-

butions. For dependency parsing, when the golden alignment is known during training, we can derive an upper bound on the latent agreement objective as

$$H^2(\mathbf{a}_t^l, \mathbf{a}_t^r) \leq 2\sqrt{D(\mathbf{g}_t || \mathbf{a}_t^l) + D(\mathbf{g}_t || \mathbf{a}_t^r)},$$

where  $D(\cdot || \cdot)$  is the KL-divergence. The complete derivation is provided in the Appendix A. During optimization, we can safely drop the constant scaler and the square root operation in the upper bound, leading to the following loss function

$$D(\mathbf{g}_t || \mathbf{a}_t^l) + D(\mathbf{g}_t || \mathbf{a}_t^r) = 2D(\mathbf{g}_t || \mathbf{a}_t^l \odot \mathbf{a}_t^r), \quad (3)$$

where  $\odot$  indicates element-wise multiplication. The resulting loss function is equivalent to the cross-entropy loss, which is widely adopted for training neural networks.

As we can see, the loss function (3) tries to minimize the distance between the golden alignment and the intersection of the two directional attention alignments at every time step. Therefore, during inference, the headword prediction for the word at time step  $t$  can be obtained as

$$\operatorname{argmax}_j \log a_{t,j}^l + \log a_{t,j}^r,$$

seeking for agreement between both query components. This parsing procedure is also similar to the exhaustive left-to-right modifier-first search algorithm described in (Covington, 2001), but it is enhanced by an additional right-to-left search with the agreement enforcement. Alternatively, we can treat  $(\log a_{t,j}^l + \log a_{t,j}^r)$  as a score of the corresponding arc and then search for the MST to form a dependency parse tree, as proposed in (McDonald et al., 2005). The MST search is achieved via the Chu-Liu-Edmonds algorithm (Chu and Liu, 1965; Edmonds, 1967), which can be implemented in  $\mathcal{O}(n^2)$  for dense graphs according to Tarjan (1977). In practice, the MST search slows down the parsing speed by 6–10%. However, it forces the parser to produce a valid tree, and we observe a slight improvement on parsing accuracy in most cases.

After obtaining each modifier and its *soft* header embeddings, we use a single-layer perceptron to predict the head-modifier relation, *i.e.*

$$\mathbf{y}_t = \operatorname{softmax}(\mathbf{U} [\tilde{\mathbf{m}}_t^l; \tilde{\mathbf{m}}_t^r] + \mathbf{W} [\mathbf{q}_t^l; \mathbf{q}_t^r]), \quad (4)$$

where  $y_{t,1}, \dots, y_{t,m}$  are the probabilities of  $m$  possible relations, and  $\mathbf{U} \in \mathbb{R}^{m \times 2e}$  and  $\mathbf{W} \in \mathbb{R}^{m \times 2d}$  are model parameters.

### 3 Model Learning

For the  $t$ -th word (modifier)  $w_t$  in a sentence of length  $n$ , let  $H_t^l$  and  $H_t^r$  denote random variables representing the predicted headword from forward (left-to-right) and backward (right-to-left) parsing directions, respectively. Also let  $R_t$  denote the random variable representing the dependency relation for  $w_t$ . The joint probability of headword and relation predictions can be written as

$$\begin{aligned} & P(R_{1:n}, H_{1:n}^l, H_{1:n}^r | w_{1:n}) \\ &= \prod_{t=1}^n P(R_t | w_{1:n}) P(H_t^l | w_{1:n}) P(H_t^r | w_{1:n}) \\ &= \prod_{t=1}^n y_{t,R_t}^l \cdot a_{t,H_t^l}^l \cdot a_{t,H_t^r}^r \end{aligned} \quad (5)$$

where at each time step we assume head-modifier relations and headwords from both directions are independent with each other when conditioned on the global knowledge of the whole sentence. Note that the long-span context and high-order parsing history information are injected when we model  $P(H_t^l | w_{1:n})$ ,  $P(H_t^r | w_{1:n})$  and  $P(R_t | w_{1:n})$ , as discussed in Section 2.2.

As discussed in Section 2.3, the model can be trained by encouraging attention agreement between two query components. From (5), we observe that it is equivalent to maximizing the log-likelihood of the golden dependency tree (or minimizing the cross-entropy) for each training sentence, *i.e.*

$$\sum_{t=1}^n \left( \log y_{t,\text{relation}_t} + \log a_{t,\text{head}_t}^l + \log a_{t,\text{head}_t}^r \right),$$

where  $a_{t,j}$  and  $y_{t,r}$  are defined in (2) and (4), respectively, and  $\text{relation}_t$  and  $\text{head}_t$  are golden relation and headword labels, respectively. The gradients are computed via the back-propagation algorithm (Rumelhart et al., 1986). Errors of  $\mathbf{y}_t$  come from the arc labels, whereas there are two source of errors for  $\mathbf{a}_t$ , one from the headword labels and the other back-propagated from errors of  $\mathbf{y}_t$ . We use stochastic gradient descent with the Adam algorithm proposed in (Kingma and Ba, 2015). The

learning rate is halved at each iteration once the log-likelihood of the dev set decreases. The whole training procedure terminates when the log-likelihood decreases for the second time. All learning parameters except bias terms are initialized randomly according to the Gaussian distribution  $\mathcal{N}(0, 10^{-2})$ . In our experiments, we tune the initial learning rate with a step size of 0.0002, and choose the best one based on the log-likelihood on the dev set at the first epoch. Empirically, the selected initial learning rates fall in the range of [0.0004, 0.0010] for hidden layer size [128, 320], and tend to be larger when using a smaller hidden layer size, *i.e.* [0.0016, 0.0034] for hidden layer size around 80. The training data are randomly shuffled at every epoch.

## 4 Experiments

In this section, we present the parsing accuracy of the proposed BiAtt-DP on 14 languages. We report both UAS and labeled attachment score (LAS), obtained by the CoNLL-X eval.pl script<sup>2</sup> which ignores punctuation symbols. The headword predictions are made through the MST search, which slightly improves both UAS and LAS (less than 0.3% absolutely). Overall, the proposed BiAtt-DP achieves competitive parsing accuracy on all languages as state-of-the-art parsers, and obtains better UAS in 6 languages. We also show the impact of using POS tags and pre-trained word embeddings. Moreover, different variants of the full model are compared in this section.

### 4.1 Data

We work on the English Treebank-3 (PTB) dataset (Marcus et al., 1999), the Chinese Treebank-5.1 (CTB) dataset (Palmer et al., 2005), and 12 other languages from the CoNLL 2006 shared task (Buchholz and Marsi, 2006). For PTB and CTB datasets, we use exactly the same setup as in (Chen and Manning, 2014; Dyer et al., 2015). Specifically, we convert the English and Chinese data using the Stanford parser v3.3.0 (de Marneffe et al., 2006) and the Penn2Malt tool (Zhang and Clark, 2008), respectively.

For English, POS tags are obtained using the Stanford POS tagger v3.3.0 (Toutanova et al., 2003),

<sup>2</sup><http://ilk.uvt.nl/conll/software.html>

whereas for Chinese, we use gold segmentation and POS tags. When constructing the token embeddings for English and Chinese, both the word form and the POS tag are used. We also initialize  $\mathbf{E}^{\text{form}}$  by pre-trained word embeddings<sup>3</sup>.

For the 12 other languages, we randomly hold out 5% of the training data as the dev set. In addition to the word form and fine-grained POS tags, we use extra features such as lemmas, coarse-grained POS tags, and morphemes when they are available in the dataset. No pre-trained word embeddings are used for these 12 languages.

## 4.2 Model Configurations

The hidden layer size is kept the same across all RNNs in the proposed BiAtt-DP. We also require the dimension of the token embeddings to be the same as the hidden layer size. Note that we concatenate the hidden layers of two RNNs for constructing  $\mathbf{m}_j$ , and thus we have  $e = 2d$ . The weight matrices  $\mathbf{C}$  and  $\mathbf{D}$  respectively project vectors  $\mathbf{m}_j$  and  $\mathbf{q}_t$  to the same dimension  $h$ , which is equivalent to  $d$ . For English and Chinese, since the dimension of pre-trained word embeddings are 300, we use  $300 \times h$  as the dimension of embedding parameters  $\mathbf{E}$ 's. For the 12 other languages, we use square matrices for the embedding parameters  $\mathbf{E}$ 's. For all languages, We tune the hidden layer size and choose one according to UAS on the dev set. The selected hidden layer sizes for these languages are: 368 (English), 114 (Chinese), 128 (Arabic), 160 (Bulgarian), 224 (Czech), 176 (Danish), 220 (Dutch), 200 (German), 128 (Japanese), 168 (Portuguese), 128 (Slovene), 144 (Spanish), 176 (Swedish), and 128 (Turkish).

## 4.3 Results

We first compare our parser with state-of-the-art neural transition-based dependency parsers on PTB and CTB. For English, we also compare with state-of-the-art graph-based dependency parsers. The results are shown in Table 1 and Table 2, respectively. It can be seen that the BiAtt-DP outperforms all other graph-based parsers on PTB. Compared with

<sup>3</sup>For English, we use the dependency-based word embeddings at <https://goo.gl/tWke3I> (Levy and Goldberg, 2014). For Chinese, we pre-train 192-dimension skip-gram embeddings (Mikolov et al., 2013) on Chinese Gigawords (Graff et al., 2005).

Type	Method	UAS	LAS
Trans.	C&M (2014)	91.8 <sub>_</sub>	89.6 <sub>_</sub>
	Dyer et al. (2015)	93.2 <sub>_</sub>	90.9 <sub>_</sub>
	B&N (2012) <sup>†</sup>	93.33	91.22
	Alberti et al. (2015) <sup>†</sup>	94.23	92.41
	Weiss et al. (2015) <sup>†</sup>	94.26	92.41
	Andor et al. (2016) <sup>*</sup>	<b>94.41</b>	<b>92.55</b>
Graph	Bohnet (2010) <sup>†</sup>	92.88	90.71
	Martins et al. (2013) <sup>†</sup>	92.89	90.55
	Z&M (2014) <sup>†</sup>	93.22	91.02
	BiAtt-DP	<b>94.10</b>	<b>91.49</b>

**Table 1:** Parsing accuracy on PTB test set. Our parser uses the same POS tagger as C&M (2014) and Dyer et al. (2015), whereas other parsers use a different POS tagger. Results with <sup>†</sup> and <sup>\*</sup> are provided in (Alberti et al., 2015) and (Andor et al., 2016), respectively.

	Dev		Test	
	UAS	LAS	UAS	LAS
C&M (2014)	84.0	82.4	83.9	82.4
Dyer et al. (2015)	87.2	<b>85.9</b>	87.2	<b>85.7</b>
BiAtt-DP	<b>87.7</b>	85.3	<b>88.1</b>	<b>85.7</b>

**Table 2:** Parsing accuracy on CTB dev and test sets.

the transition-based parsers, it achieves better accuracy than Chen and Manning (2014), which uses a feed-forward neural network, and Dyer et al. (2015), which uses three stack LSTM networks. Compared with the integrated parsing and tagging models, the BiAtt-DP outperforms Bohnet and Nivre (2012) but has a small gap to Alberti et al. (2015). On CTB, it achieves best UAS and similar LAS. This may be caused by that the relation vocabulary size is relatively smaller than the average sentence length, which biases the joint objective to be more sensitive to UAS. The parsing speed is around 50–60 sents/sec measured on a desktop with Intel Core i7 CPU @ 3.33GHz using single thread.

Next, in Table 3 we show the parsing accuracy of the proposed BiAtt-DP on 12 languages in the CoNLL 2006 shared task, including comparison with state-of-the-art parsers. Specifically, we show UAS of the 3rd-order RBGParser as reported in (Lei et al., 2014) since it also uses low-dimensional continuous embeddings. However, there are several major differences between the RBGParser and the BiAtt-DP. First, in (Lei et al., 2014), the low-dimensional continuous embeddings are derived

Language	BiAtt-DP	RBGParser	Best Published	Crossed	Uncrossed	%Crossed
Arabic	80.34 [68.58]	79.95	<b>81.12</b> (Ma11)	17.24	80.71	0.58
Bulgarian	93.96 [89.55]	93.50	<b>94.02</b> (Zh14)	79.59	94.10	0.98
Czech	<b>91.16</b> [85.14]	90.50	90.32 (Ma13)	81.62	91.63	4.68
Danish	91.56 [85.53]	91.39	<b>92.00</b> (Zh13)	73.33	91.89	1.80
Dutch	<b>87.15</b> [82.41]	86.41	86.19 (Ma13)	82.82	87.66	10.48
German	<b>92.71</b> [89.80]	91.97	92.41 (Ma13)	85.93	92.90	2.70
Japanese	93.44 [90.67]	93.71	<b>93.72</b> (Ma11)	48.67	94.48	2.26
Portuguese	92.77 [88.44]	91.92	<b>93.03</b> (Ko10)	73.02	93.28	2.52
Slovene	86.01 [75.90]	86.24	<b>86.95</b> (Ma11)	60.11	86.99	3.66
Spanish	<b>88.74</b> [84.03]	88.00	87.98 (Zh14)	50.00	88.77	0.08
Swedish	90.50 [84.05]	91.00	<b>91.85</b> (Zh14)	45.16	90.78	0.62
Turkish	<b>78.43</b> [66.16]	76.84	77.55 (Ko10)	38.85	79.71	3.13

**Table 3:** UAS on 12 languages in the CoNLL 2006 shared task (Buchholz and Marsi, 2006). We also report corresponding LAS in squared brackets. The results of the 3rd-order RBGParser are reported in (Lei et al., 2014). Best published results on the same dataset in terms of UAS among (Pitler and McDonald, 2015), (Zhang and McDonald, 2014), (Zhang et al., 2013), (Zhang and McDonald, 2012), (Rush and Petrov, 2012), (Martins et al., 2013), (Martins et al., 2010), and (Koo et al., 2010). To study the effectiveness of the parser in dealing with non-projectivity, we follow (Pitler and McDonald, 2015), to compute the recall of crossed and uncrossed arcs in the gold tree, as well as the percentage of crossed arcs.

from low-rank tensors. Second, the RBGParser uses combined scoring of arcs by including traditional features from the MSTParser (McDonald and Pereira, 2006) / TurboParser (Martins et al., 2013). Third, the RBGParser employs a third-order parsing algorithm based on (Zhang et al., 2014), although it also implements a first-order parsing algorithm, which achieves lower UAS in general. In Table 3, we show that the proposed BiAtt-DP outperforms the RBGParser in most languages except Japanese, Slovene, and Swedish.

It can be observed from Table 3 that the BiAtt-DP has highly competitive parsing accuracy as state-of-the-art parsers. Moreover, it achieves best UAS for 5 out of 12 languages. For the remaining seven languages, the UAS gaps between the BiAtt-DP and state-of-the-art parsers are within 1.0%, except Swedish. An arguably fair comparison for the BiAtt-DP is the MSTParser (McDonald and Pereira, 2006), since the BiAtt-DP replaces the scoring function for arcs but uses exactly the same search algorithm. Due to the space limit, we refer readers to (Lei et al., 2014) for results of the MSTParsers (also shown in Appendix B). The BiAtt-DP consistently outperforms both parser by up to 5% absolute UAS score.

Finally, following (Pitler and McDonald, 2015), we also analyze the performance of the BiAtt-DP on both crossed and uncrossed arcs. Since the BiAtt-

DP uses a graph-based non-projective parsing algorithm, it is interesting to evaluate the performance on crossed arcs, which result in the non-projectivity of the dependency tree. The last three columns of Table 3 show the recall of crossed arcs, that of uncrossed arcs, and the percentage of crossed arcs in the test set. Pitler and McDonald (2015) reported numbers on the same data for Dutch, German, Portuguese, and Slovene as in this paper. For these four languages, the BiAtt-DP achieves better UAS than that reported in (Pitler and McDonald, 2015). More importantly, we observe that the improvement on recall of crossed arcs (around 10–18% absolutely) is much more significant than that of uncrossed arcs (around 1–3% absolutely), which indicates the effectiveness of the BiAtt-DP in parsing languages with non-projective trees.

#### 4.4 Ablative Study

Here we try to study the impact of using pre-trained word embeddings, POS tags, as well as the bi-directional query components on our model. First of all, we start from our best model (Model 1 in Table 4) on English, which uses 300 as the token embedding dimension and 368 as the hidden layer size. We keep those model parameter dimensions unchanged and analyze different factors by comparing the parsing accuracy on PTB dev set.

No.	INIT	POS	L2R	R2L	UAS	LAS
1	✓	✓	✓	✓	93.99	91.32
2		✓	✓	✓	93.36	90.42
3			✓	✓	91.87	87.85
4		✓		✓	92.64	89.66
5		✓	✓		92.47	89.47
6		✓	✓†	✓†	93.03	90.06

**Table 4:** Parsing accuracy on PTB dev set for different variants of the full model. INIT refers to using pre-trained word embeddings to initialize  $E^{\text{form}}$ . POS refers to using POS tags in token embeddings. L2R and R2L respectively indicate whether to use the left-to-right and right-to-left query components. † means the query component drops soft headword embeddings when constructing RNN hidden states.

The results are summarized in Table 4. Comparing Models 1–3, it can be observed that without using pre-trained word embeddings, both UAS and LAS drop by 0.6%, and without using POS tags in token embeddings, the numbers further drop by 1.6% in UAS and around 2.6% in LAS. In terms of query components, using single query component (Models 4–5) degrades UAS by 0.7–0.9% and LAS by around 1.0%, compared with Model 2. For Model 6, the soft headword embedding is only used for arc label predictions but not fed into the next hidden state, which is around 0.3% worse than Model 2. This supports the hypothesis about the usefulness of the parsing history information. We also implement a variant of Model 6 which produces one  $\mathbf{a}_t$  instead two by using both  $\mathbf{q}_t^l$  and  $\mathbf{q}_t^r$  in (1). It gets 92.44% UAS and 89.26% LAS, indicating that naively applying a bi-directional RNN may not be enough.

## 5 Related Work

**Neural Dependency Parsing:** Recently developed neural dependency parsers are mostly transition-based models, which read words sequentially from a buffer into a stack and incrementally build a parse tree by predicting a sequence of transitions (Yamada and Matsumoto, 2003; Nivre, 2003; Nivre, 2004). A feed-forward neural network is used in (Chen and Manning, 2014), where they represent the current state with 18 selected elements such as the top words on the stack and buffer. Each element is encoded by concatenated embeddings of words, POS tags, and arc labels. Their dependency parser achieves improvement

on both accuracy and parsing speed. Weiss et al. (2015) improve the parser using semi-supervised structured learning and unlabeled data. The model is extended to integrate parsing and tagging in (Alberti et al., 2015). On the other hand, Dyer et al. (2015) develop the stack LSTM architecture, which uses three LSTMs to respectively model the sequences of buffer states, stack states, and actions. Unlike the transition-based formulation, the proposed BiAtt-DP directly predicts the headword and the dependency relation at each time step. Specifically, there is no explicit representation of actions or headwords in our model. The model learns to retrieve the most relevant information from the input memory to make decisions on headwords and head-modifier relations.

**Graph-based Dependency Parsing:** In addition to the transition-based parsers, another line of research in dependency parsing uses graph-based models. Graph-based parser usually build a dependency tree from a directed graph and learns to scoring the possible arcs. Due to this nature, non-projective parsing can be done straightforwardly by most graph-based dependency parsers. The MST-Parser (McDonald et al., 2005) and the TurboParser (Martins et al., 2010) are two examples of graph-based parsers. The MSTParser formulates the parsing as searching for the MST, whereas the TurboParser performs approximate variational inference over a factor graph. The RBGParser proposed in (Lei et al., 2014) can also be viewed as a graph-based parser, which scores arcs using low-dimensional continuous features derived from low-rank tensors as well as features used by MST-Parser/TurboParser. It also employs a sampler-based algorithm for parsing (Zhang et al., 2014).

**Neural Attention Model:** The proposed BiAtt-DP is closely related to the memory network (Sukhbaatar et al., 2015) for question answering, as well as the neural attention models for machine translation (Bahdanau et al., 2015) and constituency parsing (Vinyals et al., 2015b). The way we query the memory component and obtain the soft headword embeddings is essentially the attention mechanism. However, different from the above studies where the alignment information is latent, in dependency parsing, the arc between the modifier and



headword is known during training. Thus, we can utilize these labels for attention weights. The similar idea is employed by the pointer network in (Vinyals et al., 2015a), which is used to solve three different combinatorial optimization problems.

## 6 Conclusion

In this paper, we develop a bi-directional attention model by encouraging agreement between the latent attention alignments. Through a simple and interpretable approximation, we make the connection between latent and observed alignments for training the model. We apply the bi-directional attention model incorporating the agreement objective during training to the proposed memory-network-based dependency parser. The resulting parser is able to implicitly capture the high-order parsing history without suffering from issue of high computational complexity for graph-based dependency parsing.

We have carried out empirical studies over 14 languages. The parsing accuracy of the proposed model is highly competitive with state-of-the-art dependency parsers. For English, the proposed BiAtt-DP outperforms all graph-based parsers. It also achieves state-of-the-art performance in 6 languages in terms of UAS, demonstrating the effectiveness of the proposed mechanism of bi-directional attention with agreement and its use in dependency parsing.

## A Upper Bound on $H^2(\mathbf{p}, \mathbf{q})$

Here, we use the following definition of squared Hellinger distance for countable space

$$H^2(\mathbf{p}, \mathbf{q}) = \frac{1}{2} \sum_i (\sqrt{p_i} - \sqrt{q_i})^2$$

where  $\mathbf{p}, \mathbf{q} \in \Delta^k$  are two  $k$ -simplexes. Introducing  $\mathbf{g} \in \Delta^k$ , the squared Hellinger distance can be upper bounded as

$$H^2(\mathbf{p}, \mathbf{q}) \leq \sqrt{2}H(\mathbf{p}, \mathbf{q}) \quad (6)$$

$$\leq \sqrt{2} [H(\mathbf{p}, \mathbf{g}) + H(\mathbf{q}, \mathbf{g})] \quad (7)$$

$$\leq 2\sqrt{H^2(\mathbf{p}, \mathbf{g}) + H^2(\mathbf{q}, \mathbf{g})} \quad (8)$$

where (6), (7) and (8) follow the inequalities between the  $\ell_1$ -norm and the  $\ell_2$ -norm, the triangle

inequality defined for a metric, and the Cauchy-Schwarz’s inequality, respectively. Using the relationship between the KL-divergence and the squared Hellinger distance, (8) can be further bounded by

$$2\sqrt{D(\mathbf{g}||\mathbf{p}) + D(\mathbf{g}||\mathbf{q})}.$$

## B UAS Scores of MSTParsers

Language	1st-order		2nd-order	
Arabic	78.30	(2.02)	78.75	(1.57)
Bulgarian	90.98	(3.00)	91.56	(2.42)
Czech	86.18	(4.88)	87.30	(3.76)
Danish	89.84	(1.80)	90.50	(1.14)
Dutch	82.89	(4.54)	84.11	(3.32)
German	89.54	(3.17)	90.14	(2.57)
Japanese	93.38	(0.14)	92.92	(0.60)
Portuguese	89.92	(3.17)	91.08	(2.01)
Slovene	82.09	(4.54)	83.25	(3.38)
Spanish	83.79	(4.59)	84.33	(4.05)
Swedish	88.27	(1.95)	89.05	(1.17)
Turkish	74.81	(3.74)	74.39	(4.16)
<b>Average</b>	85.83	(2.85)	86.45	(2.23)

**Table 5:** UAS scores of 1st-order and 2-nd order MSTParsers on 12 languages in the CoNLL 2006 shared task (Buchholz and Marsi, 2006). We use the numbers reported in (Lei et al., 2014). Numbers in brackets indicate the absolute improvement of the proposed BiAtt-DP over the MSTParsers.

## References

- Chris Alberti, David Weiss, Slav Petrov, and Slav Petrov. 2015. Improved transition-based parsing and tagging with neural networks. In *Proc. Conf. Empirical Methods Natural Language Process. (EMNLP)*, pages 1354–1359.
- 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 *Proc. Annu. Meeting Assoc. for Computational Linguistics (ACL)*.
- Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proc. Int. Conf. Learning Representations (ICLR)*.
- Miguel Ballesteros, Chris Dyer, and Noah A. Smith. 2015. Improved transition-based parsing by modeling characters instead of words with lstms. In *Proc. Conf. Empirical Methods Natural Language Process. (EMNLP)*, pages 349–359.

- Bernd Bohnet and Joakim Nivre. 2012. A transition-based system for joint part-of-speech tagging and labeled non-projective dependency parsing. In *Proc. Conf. Empirical Methods Natural Language Process. (EMNLP)*, pages 1455–1465.
- Bernd Bohnet. 2010. Very high accuracy and fast dependency parsing is not a contradiction. In *Proc. Int. Conf. Computational Linguistics (COLING)*, pages 89–97.
- Jan A. Botha and Phil Blunsom. 2014. Compositional morphology for word representations and language modelling. In *Proc. Int. Conf. Machine Learning (ICML)*.
- Sabine Buchholz and Erwin Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In *Proc. Conf. Computational Natural Language Learning (CoNLL)*, pages 149–164.
- Danqi Chen and Christopher D Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proc. Conf. Empirical Methods Natural Language Process. (EMNLP)*, pages 740–750.
- Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahadanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proc. Conf. Empirical Methods Natural Language Process. (EMNLP)*, pages 1724–1734.
- Yoeng-Jin Chu and Tseng-Hong Liu. 1965. On the shortest arborescence of a directed graph. *Science Sinica*, 14:1396–1400.
- Michael A. Covington. 2001. A fundamental algorithm for dependency parsing. In *Proc. Annu. ACM Southeast Conf.*, pages 95–102.
- Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proc. Int. Conf. Language Resources and Evaluation (LREC)*.
- Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proc. Annu. Meeting Assoc. for Computational Linguistics (ACL)*, pages 334–343.
- Jack Edmonds. 1967. Optimum branchings. *Journal of Research of the National Bureau of Standards*, 718(4):233–240.
- David Graff, Ke Chen, Junbo Kong, and Kazuaki Maeda. 2005. Chinese Gigaword Second Edition LDC2005T14. Web Download.
- Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proc. Int. Conf. Learning Representations (ICLR)*.
- Terry Koo, Alexander M. Rush, Michael Collins, Tommi Jaakkola, and David Sontag. 2010. Dual decomposition for parsing with non-projective head automata. In *Proc. Conf. Empirical Methods Natural Language Process. (EMNLP)*, pages 1288–1298.
- Tao Lei, Yu Xin, Yuan Zhang, Regina Barzilay, and Tommi Jaakkola. 2014. Low-rank tensors for scoring dependency structures. In *Proc. Annu. Meeting Assoc. for Computational Linguistics (ACL)*, pages 1381–1391.
- Omer Levy and Yoav Goldberg. 2014. Dependency-based word embeddings. In *Proc. Annu. Meeting Assoc. for Computational Linguistics (ACL)*, pages 302–308.
- Percy Liang, Ben Tasker, and Dan Klein. 2006. Alignment by agreement. In *Proc. Human Language Technology Conf. and Conf. North American Chapter Assoc. for Computational Linguistics (HLT-NAACL)*, pages 104–111.
- Mitchell Marcus, Beatrice Santorini, Mary Ann Marcinkiewicz, and Ann Taylor. 1999. Treebank-3 LDC99T42. Web Download.
- Andr e F. T. Martins, Noah A. Smith, and Eric P. Xing. 2010. Turbo parsers: Dependency parsing by approximate variational inference. In *Proc. Conf. Empirical Methods Natural Language Process. (EMNLP)*, pages 34–44.
- Andr e F. T. Martins, Miguel B. Almeida, and Noah A. Smith. 2013. Turing on the turbo: Fast third-order non-projective turbo parsers. In *Proc. Annu. Meeting Assoc. for Computational Linguistics (ACL)*, pages 617–622.
- Andrew L. Mass, Awni Y. Hannun, and Andrew Y. Ng. 2013. Rectifier nonlinearities improve neural network acoustic models. In *Proc. Int. Conf. Machine Learning (ICML)*.
- Ryan McDonald and Fernando Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *Proc. European Chapter Assoc. for Computational Linguistics (EACL)*, pages 81–88.
- Ryan McDonald, Fernando Pererira, Kiril Ribarov, and Jan Haji c. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proc. Human Language Technology Conf. and Conf. Empirical Methods Natural Language Process. (HLT/EMNLP)*, pages 523–530.
- Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *Proc. Workshop at Int. Conf. Learning Representations*.
- Joakim Nivre. 2003. An efficient algorithm for projective dependency parsing. In *Proc. Int. Conf. Parsing Technologies (IWPT)*, pages 149–160.

- Joakim Nivre. 2004. Incrementality in deterministic dependency parsing: Bringing engineering and cognition together. In *Proc. Workshop at ACL*.
- Martha Palmer, Fu-Dong Chiou, Nianwen Xue, and Tsan-Kuang Lee. 2005. Chinese Treebank 5.0 LDC2005T01. Web Download.
- Emily Pitler and Ryan McDonald. 2015. A linear-time translation system for crossing interval trees. In *Proc. Conf. North American Chapter Assoc. for Computational Linguistics (NAACL)*, pages 662–671.
- David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. 1986. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, October.
- Alexander M. Rush and Slav Petrov. 2012. Vine pruning for efficient multi-pass dependency parsing. In *Proc. Conf. North American Chapter Assoc. for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 498–507.
- Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. 2015. End-to-end memory networks. In *Proc. Annu. Conf. Neural Inform. Process. Syst. (NIPS)*, pages 2431–2439.
- Robert E. Tarjan. 1977. Finding optimum branchings. *Networks*, 7(1):25–35.
- Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proc. Human Language Technology Conf. and Conf. North American Chapter Assoc. for Computational Linguistics (HLT-NAACL)*, pages 173–180.
- Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015a. Pointer networks. In *Proc. Annu. Conf. Neural Inform. Process. Syst. (NIPS)*, pages 2692–2700.
- Oriol Vinyals, Lukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015b. Grammar as a foreign language. In *Proc. Annu. Conf. Neural Inform. Process. Syst. (NIPS)*, pages 2755–2763.
- David Weiss, Chris Alberti, Michael Collins, and Slav Petrov. 2015. Structured training for neural network transition-based parsing. In *Proc. Annu. Meeting Assoc. for Computational Linguistics (ACL)*, pages 323–333.
- Jason Weston, Sumit Chopra, and Antoine Bordes. 2015. Memory networks. In *Proc. Int. Conf. Learning Representations (ICLR)*.
- Hiroyasu Yamada and Yuji Matsumoto. 2003. Statistical dependency analysis with support vector machine. In *Proc. Int. Conf. Parsing Technologies (IWPT)*, pages 195–206.
- Yue Zhang and Stephen Clark. 2008. A tale of two parsers: investigating and combining graph-based and transition-based dependency parsing using beam-search. In *Proc. Conf. Empirical Methods Natural Language Process. (EMNLP)*, pages 562–571.
- Hao Zhang and Ryan McDonald. 2012. Generalized higher-order dependency parsing with cube pruning. In *Proc. Conf. Empirical Methods Natural Language Process. and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 320–331.
- Hao Zhang and Ryan McDonald. 2014. Enforcing structural diversity in cube-pruned dependency parsing. In *Proc. Annu. Meeting Assoc. for Computational Linguistics (ACL)*, pages 656–661.
- Hao Zhang, Liang Huang, Kai Zhao, and Ryan McDonald. 2013. Online learning for inexact hypergraph search. In *Proc. Conf. Empirical Methods Natural Language Process. (EMNLP)*, pages 908–913.
- Yuan Zhang, Tao Lei, Regina Barzilay, Tommi Jaakkola, and Amir Golberson. 2014. Steps to excellence: Simple inference with the refined scoring of dependency trees. In *Proc. Annu. Meeting Assoc. for Computational Linguistics (ACL)*, pages 197–207.