Attention-based Belief or Disbelief Feature Extraction for Dependency Parsing

Haoyuan Peng, Lu Liu, Yi Zhou, Junying Zhou, Xiaoqing Zheng

School of Computer Science, Fudan University, Shanghai, China
Shanghai Key Laboratory of Intelligent Information Processing
{hypeng15, l_liu15, zhouyi13, junyingzhou14, zhengxq}@fudan.edu.cn

Abstract

Existing neural dependency parsers usually encode each word in a sentence with bi-directional LSTMs, and estimate the score of an arc from the LSTM representations of the head and the modifier, possibly missing relevant context information for the arc being considered. In this study, we propose a neural feature extraction model that learns to extract arc-specific features. We apply a neural network-based attention method to collect *evidences* for and against each possible head-modifier pair, with which our model computes certainty scores of belief and disbelief, and determines the final arc score by subtracting the disbelief score from the belief score. Experiments on various datasets show that our arc-specific feature extraction mechanism significantly improves the performance of bi-directional LSTM-based models by explicitly modeling long-distance dependencies. On both English and Chinese, the proposed model achieve a better accuracy than most neural attention-based models.

Introduction

Dependency parsing has been proven useful for several natural language understanding tasks, such as relation extraction(Fundel, Küffner, and Zimmer 2006) and machine translation(Carreras and Collins 2009). A dependency parser represents the syntactic structure of a sentence with a dependency tree, in which each word and its modifier are connected by a labeled edge. There are typically two paradigms of dependency parsing models: transition-based and graph-based models. A transition-based model(Yamada and Matsumoto 2003) learns to predict a proper action from current parsing state and parsing history, and the parsing process is done by performing the predicted sequence of actions. A graph-based model defines the score of parse trees by summing the scores of its subgraphs. Various maximum spanning tree algorithms(Chu and Liu 1965; Eisner 1996) are adopted to find the parse tree with the highest score (McDonald et al. 2005). In graph-based models, parameters are learned to assign a correct score to each possible subgraph.

(McDonald and Satta 2007) show that when the tree score is factorized into high-order subgraphs instead of single arcs, exactly searching the parse tree with the highest score becomes an NP-hard problem. For that reason, graph-based parsers typically factorize the tree score in a first-order way(i.e. the arc scores are estimated independently), and score an arc with a set of local features by a linear model. Limited by the factorization over single arcs, the primary shortcoming of graph-based parsers is that features are defined over a limited subgraph but the model requires global learning and inference.

Much effort have been devoted to solving this problem. (Nivre and McDonald 2008) and (McDonald and Nivre 2011) try to integrate transition-based parsers and graph-based parsers by using the result of one to define features for the other. For graph-based parsers, the deficiency on features is made up by transition-based parsers which take rich features like parsing history into account. Another way for improvement is to introduce high-order features such as the modifier's siblings and children for each head-modifier pair, and search the optimal parse tree with an approximation algorithm(McDonald and Pereira 2006).

Regardless of the algorithms used, these traditional graph-based parsers described above heavily rely on handcrafted feature selection. Their feature templates are often manually designed to capture the syntactic structure, and the design of feature templates becomes a challengeable task. In order to avoid such feature engineering, many recent approaches introduce deep neural networks to learn dense features rather than traditional sparse indicator features, and the shallow classifiers for predicting the correct action in transition-based models, or the score functions in graph-based models are substituted with different kinds of neural networks including two-layer neural networks(Chen and Manning 2014), recurrent neural networks such as LSTM(Kiperwasser and Goldberg 2016b) and convolutional neural networks(Zheng 2017).

LSTM (Hochreiter and Schmidhuber 1997) is widely applied to generate feature representations with long distance global information, which are thought to be helpful for dependency parsing(Kiperwasser and Goldberg 2016b; Cheng et al. 2016; Zhang, Cheng, and Lapata 2016; Dozat and Manning 2017). LSTM-based models generate features by associating each word with a bi-directional LSTM vector representation and train the LSTM jointly with the parser. A weak point of such approaches is that when a sentence is fed into an LSTM, earlier inputs cause less impact on the final cell states or outputs, which are considered as the fea-

ture representations of words in the sentence, preventing the model from computing an accurate probability of an arc with the help of rich global information. To solve this problem, attention mechanism, which is first proposed in sequence-to-sequence models(Bahdanau, Cho, and Bengio 2014), is adopted in dependency parsing in two ways: feature generation and head selection. (Cheng et al. 2016) construct an attention mechanism allowing the model to capture high-order parsing history beyond the head and the modifier. (Kiperwasser and Goldberg 2016b) and (Zhang, Cheng, and Lapata 2016) predict the arc scores and select the headword for each word by neural attention mechanism, and (Dozat and Manning 2017) replace the MLP attention with a biaffine one.

Previous neural graph-based dependency parsers typically factorize the score of a parse tree in a first-order strategy, and they estimate the score of each head-modifier pair only from their LSTM representations, possibly missing relevant context information from the rest of the sentence to the pair. Inspired by certainty factor in expert systems, in this study we propose an attention-based mechanism to extract arc-specific high-order features for the estimation of each arc's score without dramatically increasing the computational complexity, by collecting evidences for or against the arc. The proposed parser first encodes each word in a sentence by a bi-directional LSTM. When scoring a head-modifier pair, model constructs two union representations for the pair by concatenating their LSTM representations and applying different MLPs on it. Then a neural attention mechanism is built between the two union representations and the rest of the sentence, with which we generate *evidence features* to compute the *belief score* and *disbelief score* for the pair. The final score is obtained by subtract the disbelief score from the belief score. Finally we run a maximum directed spanning tree algorithm on final scores to construct the parse tree.

To the best of our knowledge, the proposed model is the first one to introduce specific features defined for each particular arc with a neural attention mechanism. Experimental results demonstrate that such approach captures features with global information that are necessary for scoring arcs in a sentence.

Model

In this section, we first formalize the inference in our approach as to score each possible arc with specific feature representations generated by a bi-directional LSTM and a neural attention mechanism. Then, we present how we construct a continuous representation for each word in the input sentence and describe the key component of our model, i.e. the arc-specific feature extraction mechanism. Lastly we describe the entire parsing process.

Problem Formalization

Given an input sentence $x = (w_0, w_1, ..., w_n)$, dependency parsers find the highest scoring parse tree with an artificial added token w_0 as its *ROOT*, i.e.

$$y^* = \arg\max_{y \in Y} \; s(x, y) \tag{1}$$

where Y is the set of all possible parse trees. y^* is the target parse tree and $s(\cdot)$ is a function that calculates the overall score of a tree. In practice, $s(\cdot)$ is usually factorized into the summation of each arc's score to avoid the NP-hard exhaustive search, i.e.

$$s(x, y) = \sum_{(w_h, w_m) \in y} score(w_h, w_m) \tag{2}$$

where $score(\cdot)$ is a function to compute the score for a single arc in a parse tree between a headword w_h and its modifier w_m. In most traditional models, the arc score is computed by the dot product of a high-dimension sparse feature vector and a weighting parameter vector. The model performance relies on the choice of feature templates, and the design of features becomes the major challenge in model design. Feature engineering for a traditional parser is a labor-intensive task, for example, (Zhang and Nivre 2011) propose a feature set including 20 core components and 72 feature templates. Instead of using handcrafted traditional feature templates, in our proposed model features are generated by a bi-directional LSTM and a neural attention mechanism, and the score function is defined with neural networks which will be described in this section.

Word representations

Following previous research(Chen and Manning 2014; Dyer et al. 2015; Weiss et al. 2015; Cheng et al. 2016), we construct two fixed-size lookup tables, $E^{word} \in \mathbb{R}^{p \times |V|}$ and $E^{pos} \in \mathbb{R}^{q \times |P|}$to store word and POS tag embeddings, where V and P stand for the word set and POS tag set, and p and q are the dimensionality of word and POS tag embeddings. For each word w_i, we obtain its embedding e_i by concatenating its word and POS tag embedding.

$$e_i = E^{word} e_i^{word} \circ E^{pos} e_i^{pos} \tag{3}$$

where e_i^{word} and e_i^{pos} are one-hot binary index vectors and $\circ$ means vector concatenation.

The two lookup tables extract information for each single word, but it has been proven that long-distance information, such as words' positional information within the sentence is important for dependency parsing(McDonald, Crammer, and Pereira 2005). In order to leverage such information, we apply a bi-directional LSTM with two layers to generate the representation of each word in a sentence.

$$x_i = BiLSTM(e_{[0:n]}, i) \tag{4}$$

where n is the sentence length. In this case a bi-directional LSTM encodes each of the words in the sentence implicitly taking the whole sentence into account.

Feature Extraction by Collecting Evidences

In previous head selection models based on LSTM and neural attention(Kiperwasser and Goldberg 2016b; Zhang, Cheng, and Lapata 2016), once the LSTM representation of each word in the sentence is calculated, the model immediately concatenates the representations of each head and modifier, and feeds the concatenation result to an MLP to

obtain the score of the pair. Although LSTM representations have access to all the words and POS tags in the sentence, they are produced without guidance from any particular words or head-modifier pairs, and suffer from the fact that earlier inputs cause less impact on the final representation. In order to involve some features that are specific for a particular pair, we extend their works by applying a neural attention mechanism to extract high-order features by collecting evidences from the sentence that are for or against adding an arc between such a pair.

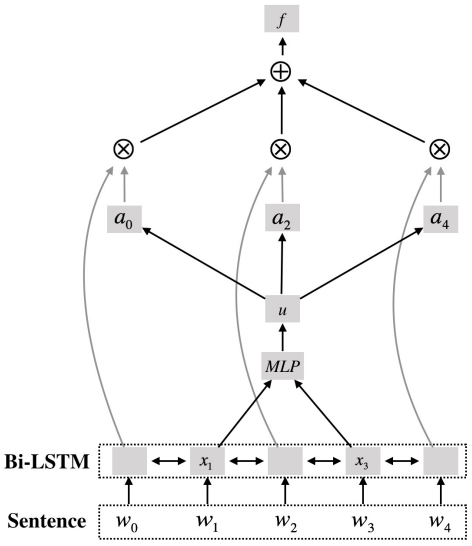

Figure 1: The structure of our feature extraction component. This figure illustrates how the evidence feature(either belief or disbelief) is produced for evaluating head-modifier pair (w_1, w_3). An MLP generates union representation of the pair, by which a neural attention mechanism computes weights for each other word in the sentence. Evidence feature of the pair is extracted by computing the weighted sum of the representations of other words. The softmax layer is omitted in the figure.

Suppose we intend to evaluate the score for headword w_h and modifier w_m, with x_h and x_m as their bi-directional LSTM representations with dimensionality r. Vanilla bi-directional LSTM-based models(Kiperwasser and Goldberg 2016b; Zhang, Cheng, and Lapata 2016) predict the score for arc (w_h, w_m) independently within the scope of x_h and x_m, missing enough context information from the rest of the sentence, though LSTM encodes the rest of the sentence implicitly. To overcome this shortcoming, when evaluating a head-modifier pair, with the knowledge about that pair our proposed model scans the input sentence again to extract specific features including evidences for belief and disbelief. Figure 1 shows how the evidence features are produced by our model. When Collecting evidences for belief, the proposed model first produces a union representation of x_h and x_m with a three-layer MLP denoted by MLP_b^r:

$$rep_b(h, m) = MLP_b^r(x_h \circ x_m) \tag{5}$$

rep_b plays a role similar to a feature map in convolutional neural networks, which project x_h and x_m to another vector space in order to collect evidences that are supportive for this pair, by computing the degree of support for each word w_i in the sentence except w_h and w_m. Similar as (Bahdanau, Cho, and Bengio 2014), we applied a neural attention on each x_i:

$$a_i^b(h, m) = x_i \cdot rep_b(h, m) \tag{6}$$

The attention score a_i^b stands for the importance of w_i in collecting evidence for believing the given head-modifier pair (w_h, w_m). We normalize the attention scores by applying a softmax layer on $\mathbf{a} = (a_1, a_2, ..., a_n)$,

$$a_i^b(h, m) \leftarrow \frac{exp(a_i^b(h, m))}{\sum\limits_{j \notin \{h,m\}} exp(a_j^b(h, m))} \tag{7}$$

and summarize all the supportive evidences as a feature vector for this pair by computing a weighted sum over all the evidences.

$$f_b(h, m) = \sum_{i \notin \{h,m\}} x_i \cdot a_i^b(h, m) \tag{8}$$

$f_b(h, m)$ is the summarization of all the supportive evidences all over the input sentence, and it is considered as one of the features for predicting the certainty score of belief. Taking $f_b(h, m)$ into account, we compute the certainty score of believing w_h and w_m with another three-layer MLP denoted by MLP_b^s:

$$belief(h, m) = MLP_b^s(x_h \circ x_m \circ f_b(h, m)) \tag{9}$$

In order to compute the score of disbelief, the proposed model replaces MLP_b^r and MLP_b^s with MLP_d^r and MLP_d^s, which have a same dimensionality. Another union representation $rep_d(h, m)$ is computed in a similar process, with which model extracts the evidence feature $f_d(h, m)$ against the given pair with a same attention mechanism, and computes the certainty score $disbelief(h, m)$ for disbelieving such pair. Model determines the final score of adding an arc between a pair by subtracting the disbelieving score from believing score.

$$score(h, m) = belief(h, m) - disbelief(h, m) \tag{10}$$

Parsing Algorithm

The parsing process in the proposed model consists of two steps: inference and refinement. The inference step predicts the most probable head for each word in the sentence basing on the score function, and the refinement step augments the result to produce valid trees. According to the fact that in a dependency parse tree, each word(except the *ROOT*) can only have exactly one head, we formalize the inference step as to select the most probable head w_j for each word w_i except *ROOT*. Giving a sentence $x = (w_0, w_1, ..., w_n)$, we calculate the score of arc (w_j, w_i) for each w_j and estimate the probability of w_j being the head of w_i as:

$$P(w_j|w_i) = \frac{exp(score(j, i))}{\sum\limits_{k \neq i} exp(score(k, i))}, i \neq 0 \tag{11}$$

to select the head for word w_i,

We train our model by minimizing the negative log-likelihood of all the correct arcs in the training set, so the loss function is:

$$J(\theta) = -\frac{1}{|T|}\sum_{x\in T}\sum_{i=1}^{len(x)} logP(h(w_i)|w_i) + \lambda||\theta||^2 \quad (12)$$

where T represents all the sentences in the training set, and $h(w_i)$ represents the correct head of word w_i in sentence x in the training set. $len(x)$ is the number of words in sentence x and the coefficient λ governs the importance of the regularization term.

The inference step described above fails to guarantee that the final parsing result forms a tree because there may be cycles, and some refinement on the result is needed to keep the final result being trees. Similar to most graph-based parsers, we adjust our inference result with a maximum directed spanning tree algorithm to produce well-formed trees. We view a sentence as a graph G with all its words including *ROOT* as its vertices and all possible arcs as its edges, then assign $P(w_i|w_j)$ to the weight of edge (i, j) in the graph. Now the problem of finding the highest scored parse tree becomes equivalent to finding a maximum spanning tree in G with *ROOT* as its root.

For applying our parser on projective languages(e.g. English and Chinese), we use Eisner algorithm(Eisner 1996) to find the maximum spanning tree. This dynamic programming algorithm has a time complexity of $O(n^3)$. Although we have only conducted experiments on projective languages, our model can be easily extended to non-projective languages(e.g. Czech) which are difficult for transition-based parsers to predict, by solving the maximum spanning tree problem with Chu-Liu-Edmonds algorithm(Chu and Liu 1965; Edmonds 1967).

So far we have showed how our proposed model do unlabeled dependency parsing, but it's necessary to extend it to produce labeled dependency arcs. On this purpose we trained a three-layer MLP with a softmax multi-class classifier denoted by MLP^l, with which we replace the the scoring layer in our model, to determine the label for each predicted arc. The classifier is trained to minimize the cross-entropy between the classifier output and the gold label. The classifier shares the first few layers including the bi-directional LSTM with the inference model and they are trained jointly.

Experiments

We conducted two sets of experiments on both English and Chinese to evaluate our model. In the following of this section, we first describe the datasets we used and the training details of our model. Then, we analyze the comparison result between the proposed model and a baseline model on both sentence length factor and linguistic factor to illustrate the effectiveness of the proposed arc-specific feature extraction in modeling long-distance dependencies. Finally, we compare the performance of our model with several recent parsers.

Dimensionality of	Value
Word Embeddings	50
POS Tag Embeddings	30
LSTM Hidden Units	200
Hidden Layer in MLP^r	300
Hidden Layer in MLP^s	400
Hidden Layer in MLP^l	400

Table 1: Model hyper-parameter configuration.

Experiment Setup

We evaluated the performance of our parser on English and Chinese with the Penn English Treebank-3(PTB) and Chinese Treebank as the datasets. For English, we adopted Stanford basic dependencies. Following the standard split of PTB, we used sections 2-21 for training, section 22 for development and 23 for testing. The POS tags were assigned by the Stanford tagger(Toutanova et al. 2003) with a tagging accuracy of 97.3%. For Chinese, we used the same setup as (Zhang and Clark 2008). In more detail, we used sections 001-815 and 1001-1136 for training, sections 886-931 and 1148-1151 for development and sections 816-885 and 1137-1147 for testing. We converted the original constituency syntactic trees to dependency trees with the Penn2Malt tool[1]. Following (Chen and Manning 2014) and (Dyer et al. 2015), we used the gold word segmentation and POS tags in Chinese.

Training Details

We trained our model on an Nvidia GPU card. Hyper-parameters were tuned on the development set of PTB, and their final configuration is listed in table 1.

Previous research demonstrates that initializing the model parameters with word embeddings trained from large unlabeled data can lead to better performance on many NLP tasks on both English(Collobert et al. 2011; Socher et al. 2011) and Chinese(Zheng, Chen, and Xu 2013). Leveraging this strategy, we used pre-trained GloVe (Pennington, Socher, and Manning 2014) word embeddings to initialize our word embedding matrix both on English and Chinese. For experiments on PTB, we pre-trained the embeddings on Wikipedia corpus. For experiments on CTB, we pre-trained the embeddings on Chinese Wikipedia corpus segmented with the model proposed by (Zheng, Chen, and Xu 2013). We used Adam(Kingma and Ba 2014) to optimize our model parameters with hyper-parameters recommended by the authors(i.e. learning rate = 0.001, $\beta_1 = 0.9$ and $\beta_2 = 0.999$). Dropout(Srivastava et al. 2014) is applied to our model by dropping nodes in LSTM layers with probability 0.5.

Sentence Length and Linguistic Factors

In order to know how well the arc-specific features introduced by our model benefit the performance, we re-implemented the model of (Zhang, Cheng, and Lapata 2016)

[1] https://stp.lingfil.uu.se/~nivre/research/Penn2Malt.html

as the baseline, by removing the attention-based feature extraction component from our proposed model, and compared their performances on sentences with different lengths. In the baseline model, arc scores are computed from the LSTM representations of the head and the modifier without any arc-specific feature. The baseline model achieves an arc-prediction accuracy of 94.3% on the development set of PTB and 87.3% on the development set of CTB. Following (Zhang, Cheng, and Lapata 2016), we sorted all the sentences in the development set of PTB and CTB by their length in a non-decreasing order, and divided them equally into 10 buckets. Then we evaluated the UAS of both models on sentences in each bucket and the results are showed in figure 2 and 3.

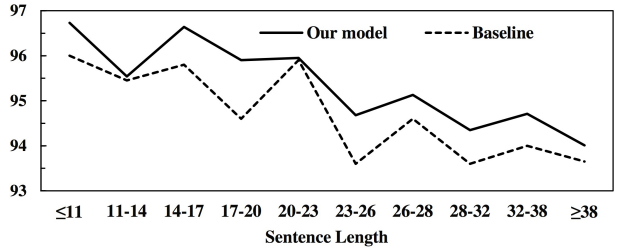

Figure 2: UAS against the sentence length on the development set of PTB. The horizontal axis is the range of the length of the sentences in each bucket.

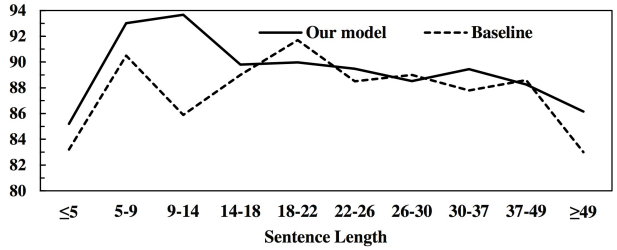

Figure 3: UAS against the sentence length on the development set of CTB. The horizontal axis is the range of the length of the sentences in each bucket.

Figure 2 and 3 illustrate that our proposed model with arc-specific feature extraction component obtains a better performance in almost all buckets of sentences than the baseline model. Another result of this experiment is that the flat accuracy curve suggests that our proposed model obtains a high performance on long sentences. Our proposed model achieves an accuracy over 94% even on the longest ten percents of sentences in PTB and an accuracy over 86% on the same bucket in CTB, in spite that dependency parsers often suffers from long-distance dependencies and have lower accuracies while processing long sentences. This behavior suggests that our feature extraction mechanism captures such long-distance relations by scanning the sentence again with the guidance from a particular pair.

According to the fact that previous research(McDonald and Nivre 2007) indicate that verbs and conjunctions are often involved in dependencies closer to the root that span longer distances, while nouns and pronouns are usually attached to verbs with shorter distance, we computed the UAS of words with various part-of-speech on the PTB development sets to verify the idea that the proposed model captures long-distance dependencies in a sentence.

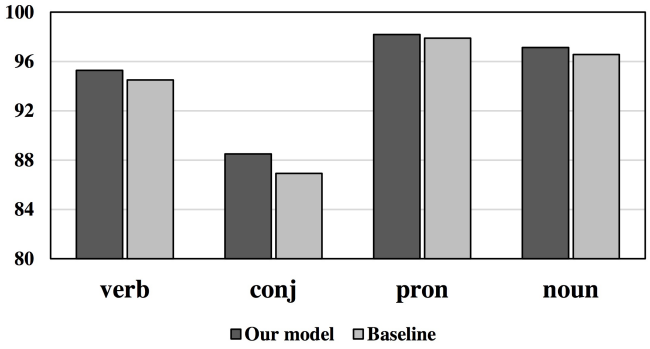

Figure 4: The performance of the proposed model and the baseline model on words with different part-of-speech. The performances are evaluated on the development set of PTB.

Figure 4 illustrates the accuracy of the proposed model and the baseline model for different part of speech(verbs, nouns, pronouns and conjunctives). Result shows that the proposed model obtains greater progress on verbs and conjunctions(0.79% and 1.58%), which are considered to involve dependencies that span long distances, than on pronouns and nouns(0.30% and 0.57%). This fact suggests that the proposed feature extraction component indeed captures the long-distance dependencies and makes better score evaluation for selecting the headwords for verbs and conjunctives.

Results

Table 2 and 3 compares the performance of our model with several recent parsers on two languages respectively. For English and Chinese, we report both unlabeled attachment score(UAS) and labeled attachment scores(LAS). Following previous works, punctuations are excluded from the evaluation.

Experimental results on PTB are shown in table 2. For comparison, we show the results of several recent studies with the same evaluation protocol. The first block in table 2 shows the performance of previous transition-based parsers, and the second block shows the performance of graph-based parsers.

We focus on the comparison between our proposed model and approaches from (Kiperwasser and Goldberg 2016b), (Zhang, Cheng, and Lapata 2016), (Cheng et al. 2016) and (Dozat and Manning 2017), which extract features for parser by a bi-directional LSTM and apply different neural attention mechanisms on it. (Kiperwasser and Goldberg 2016b) and (Zhang, Cheng, and Lapata 2016) adopt neural attention to select the headwords, and (Dozat and Manning 2017) develop a larger neural network with a biaffine attention. (Cheng et al. 2016) learn to capture the high-order parsing

Parser	UAS	LAS
(Chen and Manning 2014)	92.00	90.70
(Zhou et al. 2015)	93.28	92.35
(Ballesteros et al. 2016)	93.56	91.42
(Kiperwasser and Goldberg 2016b)	93.20	91.20
(Andor et al. 2016)	94.61	92.79
(Kuncoro et al. 2016)	**95.80**	**94.60**
(Kiperwasser and Goldberg 2016b)	93.00	90.90
(Zhang, Cheng, and Lapata 2016)	94.10	91.90
(Cheng et al. 2016)	94.10	91.49
(Hashimoto et al. 2016)	94.67	92.90
(Dozat and Manning 2017)	95.74	94.08
Our model	94.96	92.51

Table 2: Results on English dataset(PTB)

Parser	UAS	LAS
(Ballesteros et al. 2016)	87.65	86.21
(Kiperwasser and Goldberg 2016a)	87.10	85.50
(Kiperwasser and Goldberg 2016b)	87.60	86.10
(Zhang, Cheng, and Lapata 2016)	87.84	86.15
(Cheng et al. 2016)	88.10	85.70
(Dozat and Manning 2017)	**89.30**	**88.23**
Our model	88.40	85.74

Table 3: Results on Chinese dataset(CTB)

history with soft headword embeddings computed from a bi-directional attention model. Different from all the models above, our proposed model apply neural attention to extract arc-specific high-order features for the evaluation of each arc.

Table 2 illustrates that we achieved a significant improvement(at least 0.86% on UAS) on (Kiperwasser and Goldberg 2016b), (Zhang, Cheng, and Lapata 2016) and (Cheng et al. 2016). This experimental result suggests that our feature extraction mechanism extracts richer features and assigns a more accurate score for each arc, comparing to previous approaches which use attention to implicitly model the parsing history as high-order features or simply to select heads. This behavior can be explained as that prior models implicitly capture the high-order features with an attention mechanism but miss the important arc-specific features for a particular score estimation operation. In the proposed model, such information is extracted by collecting evidences with guidance from the pair to be evaluated.

Results on CTB also show a similar result. As demonstrated in table 3, our model outperforms most of our comparators on both UAS and LAS. However, our model lags behind (Dozat and Manning 2017) for these following reasons: First (Dozat and Manning 2017) adopt a much larger neural network with deeper LSTM layers(3 or 4 layers) to produce feature representations for each word in the sentence and compute the score with a biaffine score function while we build our attention mechanism on representations generated from a two-layer LSTM network. Secondly (Dozat and Manning 2017) make choices on a large set of hyper-parameter configurations including different classifiers, different number and dimensionality of layer, as well as different recurrent units to allow their model to outperform others, whereas we just tried a few different network configurations.

Related Work

In this section we first briefly review previous works on transition-based and graph-based models, then discuss how our model is related to previous approaches.

Transition-based parsers transform a sentence into a dependency parse tree by performing a sequence of transition actions, and the parsing problem can be viewed as predicting the optimal sequence of actions until the final dependency tree is obtained. A transition-based model usually includes a stack to store partial parsing results and a buffer to store the rest of the sentence. (Chen and Manning 2014) first attempt to introduce deep learning into transition-based dependency parsing by predicting the transition actions with a neural network classifier. At each step, a neural network computes the probability of each action from current state, which includes words on the top of a stack and a buffer. Some research attempts to augment (Chen and Manning 2014) by beam search or CRF methods, allowing the parser to keep the top k sequences rather than using a greedy algorithm, and to undo previous incorrect actions. (Weiss et al. 2015; Zhou et al. 2015; Andor et al. 2016). (Dyer et al. 2015) and (Kuncoro et al. 2016) use LSTMs to respectively represent the parsing states including the stack, the buffer and the parsing history and get the state-of-the-art performance.

Graph-based models usually consider the words in a sentence as vertices in a graph, and each of the possible arcs as the edges(McDonald et al. 2005). Parsers typically learn to assign a weight to each edge and construct the parse tree with the highest score by running maximum spanning tree algorithms on such a directed graph. The Eisner parsing algorithm (Eisner 1996) is sufficient for finding projective parse trees and the Chu-liu-Edmonds algorithm (Chu and Liu 1965; Edmonds 1967) is to find non-projective parse trees. (Kiperwasser and Goldberg 2016b) propose a bi-directional LSTM-based model in which each word is associated with a bi-directional LSTM vector, representing the word with information from its context. To predict the score of an arc, the representations of the head and the modifier are concatenated as the input of an MLP for scoring this pair. Analogously, a multi-class MLP outputs the label of an arc between each word and its predicted headword based on their LSTM representations.

(Kiperwasser and Goldberg 2016a) propose a model based on hierarchical tree LSTMs, in which the left and right sequences of modifiers are modeled with RNNs, and the tree representations are produced by a greedy bottom-up dependency parser based on an easy-first transition system. In their joint many-task model, (Hashimoto et al. 2016) replace the MLP-based attention mechanism applied in (Kiperwasser and Goldberg 2016b) with a bilinear one. (Zheng 2017) propose an increasing neural dependency parser that forms an

initial parse tree by first-order features and defined high-order features over the initial tree, in order to refine the parse tree in an iterative way.

The proposed model is most related with other parsers based on neural attention mechanism(Kiperwasser and Goldberg 2016b; Zhang, Cheng, and Lapata 2016; Cheng et al. 2016; Dozat and Manning 2017). (Zhang, Cheng, and Lapata 2016) also represent each word in a sentence with a bi-directional LSTM and score each pair with an attention, but their model locally optimizes a set of head-modifier predictions in a greedy way, without to enforce any global consistency during the training process. (Cheng et al. 2016) develop a graph-based neural dependency parser with a bi-directional attention to implicitly capture the high-order parsing history, circumventing the limitation of other graph-based parsers which are unable to take previous parse decisions into account. (Dozat and Manning 2017) build a larger but more thoroughly regularized parser with 3 layers of LSTM networks and replace the traditional MLP-based attention with a biaffine one to improve the performance. Recently, Attention mechanism is also introduced to transition-based parsers(Zhang et al. 2017).

Compare to previous works, instead of directly using attention to predict arc scores or to implicitly model parsing histories, we make use of neural attention mechanism on extracting arc-specific high-order features by respectively collecting evidences for evaluating each head-modifier pair. Our approach supplements rich features for evaluating the arcs and obtains a better score estimation.

Conclusions

Most previous neural graph-based dependency parsers fail to make use of rich features beyond a single arc to estimate the scores. In this study we propose an arc-specific feature extraction mechanism that builds the connection between each particular head-modifier pair and the rest of the sentence. We apply a neural attention mechanism between the union representations of the pair and other words, with which our proposed parser collects evidences for belief and disbelief as high-order features without suffering from high computational complexity, and estimates the certainty score of believing and disbelieving each pair. Experimental results have demonstrated that our attention mechanism successfully captures long-distance dependencies, resulting the proposed parser to achieve a significant performance improvement on most of other attention-based parsers.

Future work will focus on exploring different arc-specific feature extraction approaches in addition to using a neural attention mechanism.

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

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*.

Carreras, X., and Collins, M. 2009. Non-projective parsing for statistical machine translation. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*, 200–209. Association for Computational Linguistics.

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)*, 740–750.

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

Chu, Y. J., and Liu, T. H. 1965. On the shortest arborescence of a directed graph. *Scientia Sinica* 14:1396–1400.

Collobert, R.; Weston, J.; Bottou, L.; Karlen, M.; Kavukcuoglu, K.; and Kuksa, P. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research* 12(Aug):2493–2537.

Dozat, T., and Manning, C. D. 2017. Deep biaffine attention for neural dependency parsing. *Proceedings of the International Conference on Learning Representations (ICLR'17)*.

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*.

Edmonds, J. 1967. Optimum branchings. *Journal of Research of the National Bureau of Standards B* 71(4):233–240.

Eisner, J. M. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of the 16th conference on Computational linguistics-Volume 1*, 340–345. Association for Computational Linguistics.

Fundel, K.; Küffner, R.; and Zimmer, R. 2006. Relex?relation extraction using dependency parse trees. *Bioinformatics* 23(3):365–371.

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*.

Hochreiter, S., and Schmidhuber, J. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780.

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*.

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., and Nivre, J. 2011. Analyzing and integrating dependency parsers. *Computational Linguistics* 37(1):197–230.

McDonald, R. T., and Pereira, F. C. 2006. Online learning of approximate dependency parsing algorithms. In *EACL*, 81–88.

McDonald, R., and Satta, G. 2007. On the complexity of non-projective data-driven dependency parsing. In *Proceedings of the 10th International Conference on Parsing Technologies*, 121–132. Association for Computational Linguistics.

McDonald, R.; Pereira, F.; Ribarov, K.; and Hajič, J. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, 523–530. Association for Computational Linguistics.

McDonald, R.; Crammer, K.; and Pereira, F. 2005. Online large-margin training of dependency parsers. In *Proceedings of the 43rd annual meeting on association for computational linguistics*, 91–98. Association for Computational Linguistics.

Nivre, J., and McDonald, R. 2008. Integrating graph-based and transition-based dependency parsers. *Proceedings of ACL-08: HLT* 950–958.

Pennington, J.; Socher, R.; and Manning, C. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 1532–1543.

Socher, R.; Lin, C. C.; Manning, C.; and Ng, A. Y. 2011. Parsing natural scenes and natural language with recursive neural networks. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, 129–136.

Srivastava, N.; Hinton, G. E.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of machine learning research* 15(1):1929–1958.

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*, 173–180. 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*.

Yamada, H., and Matsumoto, Y. 2003. Statistical dependency analysis with support vector machines. In *Proceedings of IWPT*, volume 3, 195–206.

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*, 562–571. 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*, 188–193. Association for Computational Linguistics.

Zhang, Z.; Liu, S.; Li, M.; Zhou, M.; and Chen, E. 2017. Stack-based multi-layer attention for transition-based dependency parsing. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 1678–1683. Copenhagen, Denmark: Association for Computational Linguistics.

Zhang, X.; Cheng, J.; and Lapata, M. 2016. Dependency parsing as head selection. *arXiv preprint arXiv:1606.01280*.

Zheng, X.; Chen, H.; and Xu, T. 2013. Deep learning for chinese word segmentation and pos tagging. In *EMNLP*, 647–657.

Zheng, X. 2017. Incremental graph-based neural dependency parsing. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 1656–1666. Copenhagen, Denmark: Association for Computational Linguistics.

Zhou, H.; Zhang, Y.; Huang, S.; and Chen, J. 2015. A neural probabilistic structured-prediction model for transition-based dependency parsing. In *ACL (1)*, 1213–1222.