Neural Probabilistic Model for Non-projective MST Parsing

Xuezhe Ma and **Eduard Hovy**
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA 15213, USA
`xuezhem@cs.cmu.edu, hovy@cmu.edu`

Abstract

In this paper, we propose a probabilistic parsing model that defines a proper conditional probability distribution over non-projective dependency trees for a given sentence, using neural representations as inputs. The neural network architecture is based on bi-directional LSTM-CNNs, which automatically benefits from both word- and character-level representations, by using a combination of bidirectional LSTMs and CNNs. On top of the neural network, we introduce a probabilistic structured layer, defining a conditional log-linear model over non-projective trees. By exploiting Kirchhoff's Matrix-Tree Theorem (Tutte, 1984), the partition functions and marginals can be computed efficiently, leading to a straightforward end-to-end model training procedure via back-propagation. We evaluate our model on 17 different datasets, across 14 different languages. Our parser achieves state-of-the-art parsing performance on nine datasets.

1 Introduction

Dependency parsing is one of the first stages in deep language understanding and has gained interest in the natural language processing (NLP) community, due to its usefulness in a wide range of applications. Many NLP systems, such as machine translation (Xie et al., 2011), entity coreference resolution (Ng, 2010; Durrett and Klein, 2013; Ma et al., 2016), low-resource languages processing (McDonald et al., 2013; Ma and Xia, 2014), and word sense disambiguation (Fauceglia et al., 2015), are becoming more sophisticated, in part because of utilizing syntactic knowledge such as dependency parsing trees.

Dependency trees represent syntactic relationships through labeled directed edges between heads and their dependents (modifiers). In the past few years, several dependency parsing algorithms (Nivre and Scholz, 2004; McDonald et al., 2005b; Koo and Collins, 2010; Ma and Zhao, 2012a,b) have been proposed, whose high performance heavily rely on hand-crafted features and task-specific resources that are costly to develop, making dependency parsing models difficult to adapt to new languages or new domains.

Recently, non-linear neural networks, such as recurrent neural networks (RNNs) with long-short term memory (LSTM) and convolution neural networks (CNNs), with as input distributed word representations, also known as word embeddings, have been broadly applied, with great success, to NLP problems like part-of-speech (POS) tagging (Collobert et al., 2011) and named entity recognition (NER) (Chiu and Nichols, 2016). By utilizing distributed representations as inputs, these systems are capable of learning hidden information representations directly from data instead of manually designing hand-crafted features, yielding end-to-end models (Ma and Hovy, 2016). Previous studies explored the applicability of neural representations to traditional graph-based parsing models. Some work (Kiperwasser and Goldberg, 2016; Wang and Chang, 2016) replaced the linear scoring function of each arc in traditional models with neural networks and used a margin-based objective (McDonald et al., 2005a) for model training. Other work (Zhang et al., 2016; Dozat and Manning, 2016) formalized dependency parsing as independently selecting the head of each word with cross-entropy objective, without the guarantee of a general non-projective tree structure output. Moreover, there have yet been no previous work on deriving a neural prob-

abilistic parsing model to define a proper conditional distribution over non-projective trees for a given sentence.

In this paper, we propose a probabilistic neural network-based model for non-projective dependency parsing. This parsing model uses bi-directional LSTM-CNNs (BLSTM-CNNs) as backbone to learn neural information representations, on top of which a probabilistic structured layer is constructed with a conditional log-linear model, defining a conditional distribution over all non-projective dependency trees. The architecture of BLSTM-CNNs is similar to the one used for sequence labeling tasks (Ma and Hovy, 2016), where CNNs encode character-level information of a word into its character-level representation and BLSTM models context information of each word. Due to the probabilistic structured output layer, we can use negative log-likelihood as the training objective, where the partition function and marginals can be computed via Kirchhoff's Matrix-Tree Theorem (Tutte, 1984) to process the optimization efficiently by back-propagation. At test time, parsing trees can be decoded with the maximum spanning tree (MST) algorithm (McDonald et al., 2005b). We evaluate our model on 17 treebanks across 14 different languages, achieving state-of-the-art performance on 9 treebanks. The contributions of this work are summarized as: (i) proposing a neural probabilistic model for non-projective dependency parsing. (ii) giving empirical evaluations of this model on benchmark data sets over 14 languages. (iii) achieving state-of-the-art performance with this parser on nine different treebanks.

2 Neural Probabilistic Parsing Model

In this section, we describe the components (layers) of our neural parsing model. We introduce the neural layers in our neural network one-by-one from top to bottom.

2.1 Edge-Factored Parsing Layer

In this paper, we will use the following notation: $\mathbf{x} = \{x_1, \ldots, x_n\}$ represents a generic input sentence, where x_i is the ith word. $\mathbf{y}$ represents a generic (possibly non-projective) dependency tree, which represents syntactic relationships through labeled directed edges between heads and their dependents. For example, Figure 1 shows a dependency tree for the sentence, "Economic news had
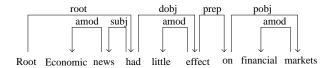

Figure 1: An example labeled dependency tree.

little effect on financial markets", with the sentences root-symbol as its root. $T(\mathbf{x})$ is used to denote the set of possible dependency trees for sentence $\mathbf{x}$.

The probabilistic model for dependency parsing defines a family of conditional probability $p(\mathbf{y}|\mathbf{x}; \Theta)$ over all $\mathbf{y}$ given sentence $\mathbf{x}$, with a log-linear form:

$$P(\mathbf{y}|\mathbf{x}; \Theta) = \frac{\exp\left(\sum_{(x_h, x_m) \in \mathbf{y}} \phi(x_h, x_m; \Theta)\right)}{Z(\mathbf{x}; \Theta)}$$

where Θ is the parameter of this model, $s_{hm} = \phi(x_h, x_m; \Theta)$ is the score function of edge from x_h to x_m, and

$$Z(\mathbf{x}; \Theta) = \sum_{\mathbf{y} \in T(\mathbf{x})} \exp\left(\sum_{(x_h, x_m) \in \mathbf{y}} s_{hm}\right)$$

is the partition function.

Bi-Linear Score Function. In our model, we adopt a bi-linear form score function:

$$\phi(x_h, x_m; \Theta) = \varphi(x_h)^T \mathbf{W} \varphi(x_m) \\ + \mathbf{U}^T \varphi(x_h) + \mathbf{V}^T \varphi(x_m) + \mathbf{b}$$

where $\Theta = \{\mathbf{W}, \mathbf{U}, \mathbf{V}, \mathbf{b}\}$, $\varphi(x_i)$ is the representation vector of x_i, $\mathbf{W}, \mathbf{U}, \mathbf{V}$ denote the weight matrix of the bi-linear term and the two weight vectors of the linear terms in ϕ, and $\mathbf{b}$ denotes the bias vector.

As discussed in Dozat and Manning (2016), the bi-linear form of score function is related to the bi-linear attention mechanism (Luong et al., 2015). The bi-linear score function differs from the traditional score function proposed in Kiperwasser and Goldberg (2016) by adding the bi-linear term. A similar score function is proposed in Dozat and Manning (2016). The difference between their and our score function is that they only used the linear term for head words ($\mathbf{U}^T \varphi(x_h)$) while use them for both heads and modifiers.

Matrix-Tree Theorem. In order to train the probabilistic parsing model, as discussed in Koo et al. (2007), we have to compute the *partition function* and the *marginals*, requiring summation over the set $T(\mathbf{x})$:

$$Z(\mathbf{x};\Theta) = \sum_{\mathbf{y}\in T(\mathbf{x})} \prod_{(x_h,x_m)\in\mathbf{y}} \psi(x_h,x_m;\Theta)$$
$$\mu_{h,m}(\mathbf{x};\Theta) = \sum_{\mathbf{y}\in T(\mathbf{x}):(x_h,x_m)\in\mathbf{y}} P(\mathbf{y}|\mathbf{x};\Theta)$$

where $\psi(x_h,x_m;\Theta)$ is the potential function:

$$\psi(x_h,x_m;\Theta) = \exp\left(\phi(x_h,x_m;\Theta)\right)$$

and $\mu_{h,m}(\mathbf{x};\Theta)$ is the marginal for edge from hth word to mth word for $\mathbf{x}$.

Previous studies (Koo et al., 2007; Smith and Smith, 2007) have presented how a variant of Kirchhoff's Matrix-Tree Theorem (Tutte, 1984) can be used to evaluate the partition function and marginals efficiently. In this section, we briefly revisit this method.

For a sentence $\mathbf{x}$ with n words, we denote $\mathbf{x} = \{x_0, x_1, \ldots, x_n\}$, where x_0 is the root-symbol. We define a complete graph G on $n+1$ nodes (including the root-symbol x_0), where each node corresponds to a word in $\mathbf{x}$ and each edge corresponds to a dependency arc between two words. Then, we assign non-negative weights to the edges of this complete graph with $n+1$ nodes, yielding the weighted adjacency matrix $\mathbf{A}(\Theta) \in \mathbb{R}^{n+1\times n+1}$, for $h, m = 0, \ldots, n$:

$$\mathbf{A}_{h,m}(\Theta) = \psi(x_h,x_m;\Theta)$$

Based on the adjacency matrix $\mathbf{A}(\Theta)$, we have the Laplacian matrix:

$$\mathbf{L}(\Theta) = \mathbf{D}(\Theta) - \mathbf{A}(\Theta)$$

where $\mathbf{D}(\Theta)$ is the weighted degree matrix:

$$\mathbf{D}_{h,m}(\Theta) = \begin{cases} \sum_{h'=0}^{n} \mathbf{A}_{h',m}(\Theta) & \text{if } h = m \\ 0 & \text{otherwise} \end{cases}$$

Then, according to Theorem 1 in Koo et al. (2007), the partition function is equal to the minor of $\mathbf{L}(\Theta)$ w.r.t row 0 and column 0:

$$Z(\mathbf{x};\Theta) = \mathbf{L}^{(0,0)}(\Theta)$$

where for a matrix $\mathbf{A}$, $\mathbf{A}^{(h,m)}$ denotes the *minor* of $\mathbf{A}$ w.r.t row h and column m; i.e., the determinant of the submatrix formed by deleting the hth row and mth column.

The marginals can be computed by calculating the matrix inversion of the matrix corresponding to $\mathbf{L}^{(0,0)}(\Theta)$. The time complexity of computing the partition function and marginals is $O(n^3)$.

Labeled Parsing Model. Though it is originally designed for unlabeled parsing, our probabilistic parsing model is easily extended to include dependency labels.

In labeled dependency trees, each edge is represented by a tuple (x_h, x_m, l), where x_h and x_m are the head word and modifier, respectively, and l is the label of dependency type of this edge. Then we can extend the original model for labeled dependency parsing by extending the score function to include dependency labels:

$$\phi(x_h,x_m,l;\Theta) = \varphi(x_h)^T\mathbf{W}_l\varphi(x_m) \\ +\mathbf{U}_l^T\varphi(x_h) + \mathbf{V}_l^T\varphi(x_m) \\ +\mathbf{b}_l$$

where $\mathbf{W}_l, \mathbf{U}_l, \mathbf{V}_l, \mathbf{b}_l$ are the weights and bias corresponding to dependency label l. Suppose that there are L different dependency labels, it suffices to define the new adjacency matrix by assigning the weight of a edge with the sum of weights over different dependency labels:

$$\mathbf{A}'_{h,m}(\Theta) = \sum_{l=1}^{L} \psi(x_h,x_m,l;\Theta)$$

The partition function and marginals over labeled dependency trees are obtained by operating on the new adjacency matrix $\mathbf{A}'(\Theta)$. The time complexity becomes $O(n^3 + Ln^2)$. In practice, L is probably large. For English, the number of edge labels in Stanford Basic Dependencies (De Marneffe et al., 2006) is 45, and the number in the treebank of CoNLL-2008 shared task (Surdeanu et al., 2008) is 70. While, the average length of sentences in English Penn Treebank (Marcus et al., 1993) is around 23. Thus, L is not negligible comparing to n.

It should be noticed that in our labeled model, for different dependency label l we use the same vector representation $\varphi(x_i)$ for each word x_i. The dependency labels are distinguished (only) by the parameters (weights and bias) corresponding to each of them. One advantage of this is that it significantly reduces the memory requirement comparing to the model in Dozat and Manning (2016) which distinguishes $\varphi_l(x_i)$ for different label l.
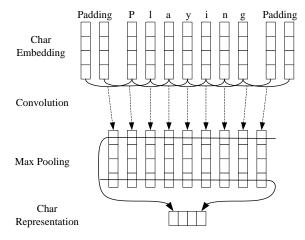
Figure 2: The convolution neural network for extracting character-level representations of words. Dashed arrows indicate a dropout layer applied before character embeddings are input to CNN.

Maximum Spanning Tree Decoding. The decoding problem of this parsing model can be formulated as:

$$
\begin{aligned}
\mathbf{y}^* &= \operatorname*{argmax}_{\mathbf{y} \in T(\mathbf{x})} P(\mathbf{y}|\mathbf{x}; \Theta) \\
&= \operatorname*{argmax}_{\mathbf{y} \in T(\mathbf{x})} \sum_{(x_h, x_m) \in \mathbf{y}} \phi(x_h, x_m; \Theta)
\end{aligned}
$$

which can be solved by using the Maximum Spanning Tree (MST) algorithm described in McDonald et al. (2005b).

2.2 Neural Network for Representation Learning

Now, the remaining question is how to obtain the vector representation of each word with a neural network. In the following subsections, we will describe the architecture of our neural network model for representation learning.

2.2.1 CNNs

Previous work (Santos and Zadrozny, 2014) have shown that CNNs are an effective approach to extract morphological information (like the prefix or suffix of a word) from characters of words and encode it into neural representations, which has been proven particularly useful on Out-of-Vocabulary words (OOV). The CNN architecture our model uses to extract character-level representation of a given word is the same as the one used in Ma and Hovy (2016). The CNN architecture is shown in Figure 2. Following Ma and Hovy (2016), a dropout layer (Srivastava et al., 2014) is applied before character embeddings are input to CNN.

2.2.2 Bi-directional LSTM

LSTM Unit. Recurrent neural networks (RNNs) are a powerful family of connectionist models that have been widely applied in NLP tasks, such as language modeling (Mikolov et al., 2010), sequence labeling (Ma and Hovy, 2016) and machine translation (Cho et al., 2014), to capture context information in languages. Though, in theory, RNNs are able to learn long-distance dependencies, in practice, they fail due to the gradient vanishing/exploding problems (Bengio et al., 1994; Pascanu et al., 2013).

LSTMs (Hochreiter and Schmidhuber, 1997) are variants of RNNs designed to cope with these gradient vanishing problems. Basically, a LSTM unit is composed of three multiplicative gates which control the proportions of information to pass and to forget on to the next time step.

BLSTM. Many linguistic structure prediction tasks can benefit from having access to both past (left) and future (right) contexts, while the LSTM's hidden state $\mathbf{h}_t$ takes information only from past, knowing nothing about the future. An elegant solution whose effectiveness has been proven by previous work (Dyer et al., 2015; Ma and Hovy, 2016) is bi-directional LSTM (BLSTM). The basic idea is to present each sequence forwards and backwards to two separate hidden states to capture past and future information, respectively. Then the two hidden states are concatenated to form the final output. As discussed in Dozat and Manning (2016), there are more than one advantages to apply a multilayer perceptron (MLP) to the output vectors of BLSTM before the score function, eg. reducing the dimensionality and overfitting of the model. We follow this work by using a one-layer perceptron with elu (Clevert et al., 2015) as activation function.

2.3 BLSTM-CNNs

Finally, we construct our neural network model by feeding the output vectors of BLSTM (after MLP) into the parsing layer. Figure 3 illustrates the architecture of our network in detail.

For each word, the CNN in Figure 2, with character embeddings as inputs, encodes the character-level representation. Then the character-level representation vector is concatenated with the word embedding vector to feed into the BLSTM network. To enrich word-level information, we also use POS embeddings. Finally, the output vec-
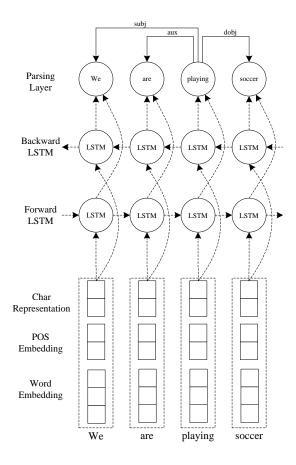
Figure 3: The main architecture of our parsing model. The character representation for each word is computed by the CNN in Figure 2. Then the character representation vector is concatenated with the word and pos embedding before feeding into the BLSTM network. Dashed arrows indicate dropout layers applied on the input, hidden and output vectors of BLSTM.

Layer	Hyper-parameter	Value
CNN	window size	3
	number of filters	50
LSTM	number of layers	2
	state size	256
	initial state	0.0
	peepholes	Hadamard
MLP	number of layers	1
	dimension	100
Dropout	embeddings	0.15
	LSTM hidden states	0.25
	LSTM layers	0.33
Learning	optimizer	Adam
	initial learning rate	0.002
	decay rate	0.5
	gradient clipping	5.0

Table 1: Hyper-parameters for all experiments.

nese, Dutch, English, German and Spanish, we use the structured-skipgram (Ling et al., 2015) embeddings, and for other languages we use the Polyglot (Al-Rfou et al., 2013) embeddings. The dimensions of embeddings are 100 for English, 50 for Chinese and 64 for other languages.

Character Embeddings. Following Ma and Hovy (2016), character embeddings are initialized with uniform samples from $[-\sqrt{\frac{3}{dim}}, +\sqrt{\frac{3}{dim}}]$, where we set $dim = 50$.

POS Embedding. Our model also includes POS embeddings. The same as character embeddings, POS embeddings are also 50-dimensional, initialized uniformly from $[-\sqrt{\frac{3}{dim}}, +\sqrt{\frac{3}{dim}}]$.

Weights Matrices and Bias Vectors. Matrix parameters are randomly initialized with uniform samples from $[-\sqrt{\frac{6}{r+c}}, +\sqrt{\frac{6}{r+c}}]$, where r and c are the number of of rows and columns in the structure (Glorot and Bengio, 2010). Bias vectors are initialized to zero, except the bias $\mathbf{b}_f$ for the forget gate in LSTM , which is initialized to 1.0 (Jozefowicz et al., 2015).

3.2 Optimization Algorithm

Parameter optimization is performed with the Adam optimizer (Kingma and Ba, 2014) with $\beta 1 = \beta 2 = 0.9$. We choose an initial learning rate of $\eta_0 = 0.002$. The learning rate η was adapted using a schedule $S = [e_1, e_2, \ldots, e_s]$, in which the learning rate η is annealed by

tors of the neural netwok are fed to the parsing layer to jointly parse the best (labeled) dependency tree. As shown in Figure 3, dropout layers are applied on the input, hidden and output vectors of BLSTM, using the form of recurrent dropout proposed in Gal and Ghahramani (2016).

3 Network Training

In this section, we provide details about implementing and training the neural parsing model, including parameter initialization, model optimization and hyper parameter selection.

3.1 Parameter Initialization

Word Embeddings. For all the parsing models on different languages, we initialize word vectors with pretrained word embeddings. For Chi-

Model	English				Chinese				German			
	Dev		Test		Dev		Test		Dev		Test	
	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS
Basic	94.51	92.23	94.62	92.54	84.33	81.65	84.35	81.63	90.46	87.77	90.69	88.42
+Char	94.74	92.55	94.73	92.75	85.07	82.63	85.24	82.46	92.16	89.82	92.24	90.18
+POS	94.71	92.60	94.83	92.96	88.98	87.55	89.05	87.74	91.94	89.51	92.19	90.05
Full	94.77	92.66	94.88	92.98	88.51	87.16	88.79	87.47	92.37	90.09	92.58	90.54

Table 2: Parsing performance (UAS and LAS) of different versions of our model on both the development and test sets for three languages.

multiplying a fixed decay rate $\rho = 0.5$ after $e_i \in S$ epochs respectively. We used $S = [10, 30, 50, 70, 100]$ and trained all networks for a total of 120 epochs. While the Adam optimizer automatically adjusts the global learning rate according to past gradient magnitudes, we find that this additional decay consistently improves model performance across all settings and languages. To reduce the effects of "gradient exploding", we use a gradient clipping of 5.0 (Pascanu et al., 2013). We explored other optimization algorithms such as stochastic gradient descent (SGD) with momentum, AdaDelta (Zeiler, 2012), or RMSProp (Dauphin et al., 2015), but none of them meaningfully improve upon Adam with learning rate annealing in our preliminary experiments.

Dropout Training. To mitigate overfitting, we apply the dropout method (Srivastava et al., 2014; Ma et al., 2017) to regularize our model. As shown in Figure 2 and 3, we apply dropout on character embeddings before inputting to CNN, and on the input, hidden and output vectors of BLSTM. We apply dropout rate of 0.15 to all the embeddings. For BLSTM, we use the recurrent dropout (Gal and Ghahramani, 2016) with 0.25 dropout rate between hidden states and 0.33 between layers. We found that the model using the new recurrent dropout converged much faster than standard dropout, while achiving similar performance.

3.3 Hyper-Parameter Selection

Table 1 summarizes the chosen hyper-parameters for all experiments. We tune the hyper-parameters on the development sets by random search. We use the same hyper-parameters across the models on different treebanks and languages, due to time constrains. Note that we use 2-layer BLSTM followed with 1-layer MLP. We set the state size of LSTM to 256 and the dimension of MLP to 100. Tuning these two parameters did not significantly impact the performance of our model.

	Dev		Test	
	UAS	LAS	UAS	LAS
cross-entropy	94.10	91.52	93.77	91.57
global-likelihood	94.77	92.66	94.88	92.98

Table 3: Parsing performance on PTB with different training objective functions.

4 Experiments

4.1 Setup

We evaluate our neural probabilistic parser on the same data setup as Kuncoro et al. (2016), namely the English Penn Treebank (PTB version 3.0) (Marcus et al., 1993), the Penn Chinese Treebank (CTB version 5.1) (Xue et al., 2002), and the German CoNLL 2009 corpus (Hajič et al., 2009). Following previous work, all experiments are evaluated on the metrics of unlabeled attachment score (UAS) and Labeled attachment score (LAS).

4.2 Main Results

We first construct experiments to dissect the effectiveness of each input information (embeddings) of our neural network architecture by ablation studies. We compare the performance of four versions of our model with different inputs — Basic, +POS, +Char and Full — where the Basic model utilizes only the pretrained word embeddings as inputs, while the +POS and +Char models augments the basic one with POS embedding and character information, respectively. According to the results shown in Table 2, +Char model obtains better performance than the Basic model on all the three languages, showing that character-level representations are important for dependency parsing. Second, on English and German, +Char and +POS achieves comparable performance, while on Chinese +POS significantly outperforms +Char model. Finally, the Full model achieves the best accuracy on English and German, but on Chinese +POS obtains the best. Thus, we guess that the POS information is more useful

System	English		Chinese		German	
	UAS	LAS	UAS	LAS	UAS	LAS
Bohnet and Nivre (2012)	–	–	87.3	85.9	91.4	89.4
Chen and Manning (2014)	91.8	89.6	83.9	82.4	–	–
Ballesteros et al. (2015)	91.6	89.4	85.3	83.7	88.8	86.1
Dyer et al. (2015)	93.1	90.9	87.2	85.7	–	–
Kiperwasser and Goldberg (2016): graph	93.1	91.0	86.6	85.1	–	–
Ballesteros et al. (2016)	93.6	91.4	87.7	86.2	–	–
Wang and Chang (2016)	94.1	91.8	87.6	86.2	–	–
Zhang et al. (2016)	94.1	91.9	87.8	86.2	–	–
Cheng et al. (2016)	94.1	91.5	88.1	85.7	–	–
Andor et al. (2016)	94.6	92.8	–	–	90.9	89.2
Kuncoro et al. (2016)	94.3	92.1	88.9	87.3	91.6	89.2
Dozat and Manning (2016)	**95.7**	**94.1**	**89.3**	**88.2**	**93.5**	**91.4**
This work: Basic	94.6	92.5	84.4	81.6	90.7	88.4
This work: +Char	94.7	92.8	85.2	82.5	92.2	90.2
This work: +POS	94.8	93.0	89.1	87.7	92.2	90.1
This work: Full	94.9	93.0	88.8	87.5	92.6	90.5

Table 4: UAS and LAS of four versions of our model on test sets for three languages, together with top-performance parsing systems.

for Chinese than English and German.

Table 3 gives the performance on PTB of the parsers trained with two different objective functions — the cross-entropy objective of each word, and our objective based on likelihood for an entire tree. The parser with global likelihood objective outperforms the one with simple cross-entropy objective, demonstrating the effectiveness of the global structured objective.

4.3 Comparison with Previous Work

Table 4 illustrates the results of the four versions of our model on the three languages, together with twelve previous top-performance systems for comparison. Our Full model significantly outperforms the graph-based parser proposed in Kiperwasser and Goldberg (2016) which used similar neural network architecture for representation learning (detailed discussion in Section 5). Moreover, our model achieves better results than the parser distillation method (Kuncoro et al., 2016) on all the three languages. The results of our parser are slightly worse than the scores reported in Dozat and Manning (2016). One possible reason is that, as mentioned in Section 2.1, for labeled dependency parsing Dozat and Manning (2016) used different vectors for different dependency labels to represent each word, making their model require much more memory than ours.

4.4 Experiments on CoNLL Treebanks

Datasets. To make a thorough empirical comparison with previous studies, we also evaluate our system on treebanks from CoNLL shared task on dependency parsing — the English treebank from CoNLL-2008 shared task (Surdeanu et al., 2008) and all 13 treebanks from CoNLL-2006 shared task (Buchholz and Marsi, 2006). For the treebanks from CoNLL-2006 shared task, following Cheng et al. (2016), we randomly select 5% of the training data as the development set. UAS and LAS are evaluated using the official scorer[1] of CoNLL-2006 shared task.

Baselines. We compare our model with the third-order Turbo parser (Martins et al., 2013), the low-rank tensor based model (Tensor) (Lei et al., 2014), the randomized greedy inference based (RGB) model (Zhang et al., 2014), the labeled dependency parser with inner-to-outer greedy decoding algorithm (In-Out) (Ma and Hovy, 2015), and the bi-direction attention based parser (Bi-Att) (Cheng et al., 2016). We also compare our parser against the best published results for individual languages. This comparison includes four additional systems: Koo et al. (2010), Martins et al. (2011), Zhang and McDonald (2014) and Pitler and McDonald (2015).

---
[1]http://ilk.uvt.nl/conll/software.html

	Turbo	Tensor	RGB	In-Out	Bi-Att	+POS	Full	Best Published	
	UAS	UAS	UAS	UAS [LAS]	UAS [LAS]	UAS [LAS]	UAS [LAS]	UAS	LAS
ar	79.64	79.95	80.24	79.60 [67.09]	80.34 [68.58]	80.05 [67.80]	80.80 [**69.40**]	**81.12**	–
bg	93.10	93.50	93.72	92.68 [87.79]	93.96 [89.55]	93.66 [89.79]	**94.28** [**90.60**]	94.02	–
zh	89.98	92.68	93.04	92.58 [88.51]	–	**93.44** [90.04]	93.40 [**90.10**]	93.04	–
cs	90.32	90.50	90.77	88.01 [79.31]	91.16 [85.14]	91.04 [85.82]	**91.18** [**85.92**]	91.16	85.14
da	91.48	91.39	91.86	91.44 [85.55]	91.56 [85.53]	91.52 [86.57]	91.86 [**87.07**]	**92.00**	–
nl	86.19	86.41	87.39	84.45 [80.31]	87.15 [82.41]	87.41 [84.17]	**87.85** [**84.82**]	87.39	–
en	93.22	93.02	93.25	92.45 [89.43]	–	94.43 [92.31]	**94.66** [**92.52**]	93.25	–
de	92.41	91.97	92.67	90.79 [87.74]	92.71 [89.80]	93.53 [91.55]	**93.62** [**91.90**]	92.71	89.80
ja	93.52	93.71	93.56	93.54 [91.80]	93.44 [90.67]	93.82 [92.34]	**94.02** [**92.60**]	93.80	–
pt	92.69	91.92	92.36	91.54 [87.68]	92.77 [88.44]	92.59 [**89.12**]	92.71 [88.92]	**93.03**	–
sl	86.01	86.24	86.72	84.39 [73.74]	86.01 [75.90]	85.73 [76.48]	86.73 [**77.56**]	**87.06**	–
es	85.59	88.00	88.75	86.44 [83.29]	88.74 [84.03]	88.58 [85.03]	**89.20** [**85.77**]	88.75	84.03
sv	91.14	91.00	91.08	89.94 [83.09]	90.50 [84.05]	90.89 [86.58]	91.22 [**86.92**]	**91.85**	85.26
tr	76.90	76.84	76.68	75.32 [60.39]	**78.43** [**66.16**]	75.88 [61.72]	77.71 [65.81]	78.43	66.16
av	88.73	89.08	89.44	88.08 [81.84]	–	89.47 [84.24]	89.95 [84.99]	89.83	–

Table 5: UAS and LAS on 14 treebanks from CoNLL shared tasks, together with several state-of-the-art parsers. "Best Published" includes the most accurate parsers in term of UAS among Koo et al. (2010), Martins et al. (2011), Martins et al. (2013), Lei et al. (2014), Zhang et al. (2014), Zhang and McDonald (2014), Pitler and McDonald (2015), Ma and Hovy (2015), and Cheng et al. (2016).

Results. Table 5 summarizes the results of our model, along with the state-of-the-art baselines. On average across 14 languages, our approach significantly outperforms all the baseline systems. It should be noted that the average UAS of our parser over the 14 languages is better than that of the "best published", which are from different systems that achieved best results for different languages.

For individual languages, our parser achieves state-of-the-art performance on both UAS and LAS on 8 languages — Bulgarian, Chinese, Czech, Dutch, English, German, Japanese and Spanish. On Arabic, Danish, Portuguese, Slovene and Swedish, our parser obtains the best LAS. Another interesting observation is that the Full model outperforms the +POS model on 13 languages. The only exception is Chinese, which matches the observation in Section 4.2.

5 Related Work

In recent years, several different neural network based models have been proposed and successfully applied to dependency parsing. Among these neural models, there are three approaches most similar to our model — the two graph-based parsers with BLSTM feature representation (Kiperwasser and Goldberg, 2016; Wang and Chang, 2016), and the neural bi-affine attention parser (Dozat and Manning, 2016).

Kiperwasser and Goldberg (2016) proposed a graph-based dependency parser which uses BLSTM for word-level representations. Wang and Chang (2016) used a similar model with a way to learn sentence segment embedding based on an extra forward LSTM network. Both of these two parsers trained the parsing models by optimizing margin-based objectives. There are three main differences between their models and ours. First, they only used linear form score function, instead of using the bi-linear term between the vectors of heads and modifiers. Second, They did not employ CNNs to model character-level information. Third, we proposed a probabilistic model over non-projective trees on the top of neural representations, while they trained their models with a margin-based objective. Dozat and Manning (2016) proposed neural parsing model using bi-affine score function, which is similar to the bi-linear form score function in our model. Our model mainly differ from this model by using CNN to model character-level information. Moreover, their model formalized dependency parsing as independently selecting the head of each word with cross-entropy objective, while our probabilistic parsing model jointly encodes and decodes parsing trees for given sentences.

6 Conclusion

In this paper, we proposed a neural probabilistic model for non-projective dependency parsing, using the BLSTM-CNNs architecture for representation learning. Experimental results on 17 treebanks across 14 languages show that our parser significantly improves the accuracy of both dependency structures (UAS) and edge labels (LAS), over several previously state-of-the-art systems.

References

Rami Al-Rfou, Bryan Perozzi, and Steven Skiena. 2013. Polyglot: Distributed word representations for multilingual nlp. In *Proceedings of CoNLL-2013*. Sofia, Bulgaria, pages 183–192.

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally normalized transition-based neural networks. In *Proceedings of ACL-2016 (Volume 1: Long Papers)*. Berlin, Germany, pages 2442–2452.

Miguel Ballesteros, Chris Dyer, and Noah A. Smith. 2015. Improved transition-based parsing by modeling characters instead of words with lstms. In *Proceedings of EMNLP-2015*. Lisbon, Portugal, pages 349–359.

Miguel Ballesteros, Yoav Goldberg, Chris Dyer, and Noah A. Smith. 2016. Training with exploration improves a greedy stack lstm parser. In *Proceedings of EMNLP-2016*. Austin, Texas, pages 2005–2010.

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *Neural Networks, IEEE Transactions on* 5(2):157–166.

Bernd Bohnet and Joakim Nivre. 2012. A transition-based system for joint part-of-speech tagging and labeled non-projective dependency parsing. In *Proceedings of EMNLP-2012*. Jeju Island, Korea, pages 1455–1465.

Sabine Buchholz and Erwin Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In *Proceeding of CoNLL-2006*. New York, NY, pages 149–164.

Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of EMNLP-2014*. Doha, Qatar, pages 740–750.

Hao Cheng, Hao Fang, Xiaodong He, Jianfeng Gao, and Li Deng. 2016. Bi-directional attention with agreement for dependency parsing. In *Proceedings of EMNLP-2016*. Austin, Texas, pages 2204–2214.

Jason Chiu and Eric Nichols. 2016. Named entity recognition with bidirectional lstm-cnns. *Transactions of the Association for Computational Linguistics* 4:357–370.

Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259* .

Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. 2015. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289* .

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research* 12:2493–2537.

Yann N Dauphin, Harm de Vries, Junyoung Chung, and Yoshua Bengio. 2015. Rmsprop and equilibrated adaptive learning rates for non-convex optimization. *arXiv preprint arXiv:1502.04390* .

Marie-Catherine De Marneffe, Bill MacCartney, Christopher D. Manning, et al. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of LREC-2006*. pages 449–454.

Timothy Dozat and Christopher D. Manning. 2016. Deep biaffine attention for neural dependency parsing. *arXiv preprint arXiv:1611.01734* .

Greg Durrett and Dan Klein. 2013. Easy victories and uphill battles in coreference resolution. In *Proceedings of EMNLP-2013*. Seattle, Washington, USA, pages 1971–1982.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proceedings of ACL-2015 (Volume 1: Long Papers)*. Beijing, China, pages 334–343.

Nicolas R Fauceglia, Yiu-Chang Lin, Xuezhe Ma, and Eduard Hovy. 2015. Word sense disambiguation via propstore and ontonotes for event mention detection. In *Proceedings of the The 3rd Workshop on EVENTS: Definition, Detection, Coreference, and Representation*. Denver, Colorado, pages 11–15.

Yarin Gal and Zoubin Ghahramani. 2016. A theoretically grounded application of dropout in recurrent neural networks. In *Advances in Neural Information Processing Systems*.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *International conference on artificial intelligence and statistics*. pages 249–256.

Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, et al. 2009. The conll-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proceedings of CoNLL-2009: Shared Task*. pages 1–18.

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

Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. 2015. An empirical exploration of recurrent network architectures. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*. pages 2342–2350.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* .

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional lstm feature representations. *Transactions of the Association for Computational Linguistics* 4:313–327.

Terry Koo and Michael Collins. 2010. Efficient third-order dependency parsers. In *Proceedings of ACL-2010*. Uppsala, Sweden, pages 1–11.

Terry Koo, Amir Globerson, Xavier Carreras, and Michael Collins. 2007. Structured prediction models via the matrix-tree theorem. In *Proceedings of EMNLP-2007*. Prague, Czech Republic, pages 141–150.

Terry Koo, Alexander M. Rush, Michael Collins, Tommi Jaakkola, and David Sontag. 2010. Dual decomposition for parsing with non-projective head automata. In *Proceedings of EMNLP-2010*. Cambridge, MA, pages 1288–1298.

Adhiguna Kuncoro, Miguel Ballesteros, Lingpeng Kong, Chris Dyer, and Noah A. Smith. 2016. Distilling an ensemble of greedy dependency parsers into one mst parser. In *Proceedings of EMNLP-2016*. Austin, Texas, pages 1744–1753.

Tao Lei, Yu Xin, Yuan Zhang, Regina Barzilay, and Tommi Jaakkola. 2014. Low-rank tensors for scoring dependency structures. In *Proceedings of ACL-2014 (Volume 1: Long Papers)*. Baltimore, Maryland, pages 1381–1391.

Wang Ling, Chris Dyer, Alan W Black, and Isabel Trancoso. 2015. Two/too simple adaptations of word2vec for syntax problems. In *Proceedings of NAACL-2015*. Denver, Colorado, pages 1299–1304.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of EMNLP-2015*. Lisbon, Portugal, pages 1412–1421.

Xuezhe Ma, Yingkai Gao, Zhiting Hu, Yaoliang Yu, Yuntian Deng, and Eduard Hovy. 2017. Dropout with expectation-linear regularization. In *Proceedings of the 5th International Conference on Learning Representations (ICLR-2017)*. Toulon, France.

Xuezhe Ma and Eduard Hovy. 2015. Efficient inner-to-outer greedy algorithm for higher-order labeled dependency parsing. In *Proceedings of EMNLP-2015*. Lisbon, Portugal, pages 1322–1328.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *Proceedings of ACL-2016 (Volume 1: Long Papers)*. Berlin, Germany, pages 1064–1074.

Xuezhe Ma, Zhengzhong Liu, and Eduard Hovy. 2016. Unsupervised ranking model for entity coreference resolution. In *Proceedings of NAACL-2016*. San Diego, California, USA.

Xuezhe Ma and Fei Xia. 2014. Unsupervised dependency parsing with transferring distribution via parallel guidance and entropy regularization. In *Proceedings of ACL-2014*. Baltimore, Maryland, pages 1337–1348.

Xuezhe Ma and Hai Zhao. 2012a. Fourth-order dependency parsing. In *Proceedings of COLING 2012: Posters*. Mumbai, India, pages 785–796.

Xuezhe Ma and Hai Zhao. 2012b. Probabilistic models for high-order projective dependency parsing. *Technical Report, arXiv:1502.04174* .

Mitchell Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics* 19(2):313–330.

Andre Martins, Miguel Almeida, and Noah A. Smith. 2013. Turning on the turbo: Fast third-order non-projective turbo parsers. In *Proceedings of ACL-2013 (Volume 2: Short Papers)*. Sofia, Bulgaria, pages 617–622.

Andre Martins, Noah Smith, Mario Figueiredo, and Pedro Aguiar. 2011. Dual decomposition with many overlapping components. In *Proceedings of EMNLP-2011*. Edinburgh, Scotland, UK., pages 238–249.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005a. Online large-margin training of dependency parsers. In *Proceedings of ACL-2005*. Ann Arbor, Michigan, USA, pages 91–98.

Ryan McDonald, Joakim Nivre, Yvonne Quirmbach-Brundage, Yoav Goldberg, Dipanjan Das, Kuzman Ganchev, Keith Hall, Slav Petrov, Hao Zhang, Oscar Täckström, Claudia Bedini, Núria Bertomeu Castelló, and Jungmee Lee. 2013. Universal dependency annotation for multilingual parsing. In *Proceedings of ACL-2013*. Sofia, Bulgaria, pages 92–97.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajic. 2005b. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of HLT/EMNLP-2005*. Vancouver, Canada, pages 523–530.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Interspeech*. volume 2, page 3.

Vincent Ng. 2010. Supervised noun phrase coreference research: The first fifteen years. In *Proceedings of ACL-2010*. Association for Computational Linguistics, Uppsala, Sweden, pages 1396–1411.

Joakim Nivre and Mario Scholz. 2004. Deterministic dependency parsing of English text. In *Proceedings of COLING-2004*. Geneva, Switzerland, pages 64–70.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *Proceedings of ICML-2013*. pages 1310–1318.

Emily Pitler and Ryan McDonald. 2015. A linear-time transition system for crossing interval trees. In *Proceedings of NAACL-2015*. Denver, Colorado, pages 662–671.

Cicero D Santos and Bianca Zadrozny. 2014. Learning character-level representations for part-of-speech tagging. In *Proceedings of ICML-2014*. pages 1818–1826.

David A. Smith and Noah A. Smith. 2007. Probabilistic models of nonprojective dependency trees. In *Proceedings of EMNLP-2007*. Prague, Czech Republic, pages 132–140.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research* 15(1):1929–1958.

Mihai Surdeanu, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. 2008. The conll-2008 shared task on joint parsing of syntactic and semantic dependencies. In *Proceedings of CoNLL-2008*. pages 159–177.

William Thomas Tutte. 1984. *Graph theory*, volume 11. Addison-Wesley Menlo Park.

Wenhui Wang and Baobao Chang. 2016. Graph-based dependency parsing with bidirectional lstm. In *Proceedings of ACL-2016 (Volume 1: Long Papers)*. Berlin, Germany, pages 2306–2315.

Jun Xie, Haitao Mi, and Qun Liu. 2011. A novel dependency-to-string model for statistical machine translation. In *Proceedings of EMNLP-2011*. Edinburgh, Scotland, UK., pages 216–226.

Nianwen Xue, Fu-Dong Chiou, and Martha Palmer. 2002. Building a large-scale annotated chinese corpus. In *Proceedings of COLING-2002*. pages 1–8.

Matthew D Zeiler. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701* .

Hao Zhang and Ryan McDonald. 2014. Enforcing structural diversity in cube-pruned dependency parsing. In *Proceedings of ACL-2014 (Volume 2: Short Papers)*. Baltimore, Maryland, pages 656–661.

Xingxing Zhang, Jianpeng Cheng, and Mirella Lapata. 2016. Dependency parsing as head selection. *arXiv preprint arXiv:1606.01280* .

Yuan Zhang, Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2014. Greed is good if randomized: New inference for dependency parsing. In *Proceedings of EMNLP-2014*. Doha, Qatar, pages 1013–1024.