Discourse Parsing with Attention-based Hierarchical Neural Networks

Qi Li **Tianshi Li** **Baobao Chang**
Key Laboratory of Computational Linguistics, Ministry of Education
School of Electronics Engineering and Computer Science, Peking University
No.5 Yiheyuan Road, Haidian District, Beijing, 100871, China
Collaborative Innovation Center for Language Ability, Xuzhou, 221009, China
qi.li@pku.edu.cn lts_417@hotmail.com chbb@pku.edu.cn

Abstract

RST-style document-level discourse parsing remains a difficult task and efficient deep learning models on this task have rarely been presented. In this paper, we propose an attention-based hierarchical neural network model for discourse parsing. We also incorporate tensor-based transformation function to model complicated feature interactions. Experimental results show that our approach obtains comparable performance to the contemporary state-of-the-art systems with little manual feature engineering.

1 Introduction

A document is formed by a series of coherent text units. Document-level discourse parsing is a task to identify the relations between the text units and to determine the structure of the whole document the text units form. Rhetorical Structure Theory (RST) (Mann and Thompson, 1988) is one of the most influential discourse theories. According to RST, the discourse structure of a document can be represented by a Discourse Tree (DT). Each leaf of a DT denotes a text unit referred to as an Elementary Discourse Unit (EDU) and an inner node of a DT represents a text span which is constituted by several adjacent EDUs. DTs can be utilized by many NLP tasks including automatic document summarization (Louis et al., 2010; Marcu, 2000), question-answering (Verberne et al., 2007) and sentiment analysis (Somasundaran, 2010) etc.

Much work has been devoted to the task of RST-style discourse parsing and most state-of-the-art approaches heavily rely on manual feature engineering (Joty et al., 2013; Feng and Hirst, 2014; Ji and Eisenstein, 2014). While neural network models have been increasingly focused on for their ability to automatically extract efficient features which reduces the burden of feature engineering, there is little neural network based work for RST-style discourse parsing except the work of Li et al. (2014a). Li et al. (2014a) propose a recursive neural network model to compute the representation for each text span based on the representations of its subtrees. However, vanilla recursive neural networks suffer from gradient vanishing for long sequences and the normal transformation function they use is weak at modeling complicated interactions which has been stated by Socher et al. (2013). As many documents contain more than a hundred EDUs which form quite a long sequence, those weaknesses may lead to inferior results on this task.

In this paper, we propose to use a hierarchical bidirectional Long Short-Term Memory (bi-LSTM) network to learn representations of text spans. Comparing with vanilla recursive/recurrent neural networks, LSTM-based networks can store information for a long period of time and don't suffer from gradient vanishing problem. We apply a hierarchical bi-LSTM network because the way words form an EDU and EDUs form a text span is different and thus they should be modeled separately and hierarchically. On top of that, we apply attention mechanism to attend over all EDUs to pick up prominent semantic information of a text span. Besides, we use tensor-based transformation function to model complicated feature interactions and thus it can produce combinatorial features.

We summarize contributions of our work as follows:

- We propose to use a hierarchical bidirectional LSTM network to learn the compositional semantic representations of text spans, which naturally matches and models the intrinsic hierarchical structure of text spans.
- We extend our hierarchical bi-LSTM network with attention mechanism to allow the network to focus on the parts of input containing prominent semantic information for the compositional representations of text spans and thus alleviate the problem caused by the limited memory of LSTM for long text spans.
- We adopt a tensor-based transformation function to allow explicit feature interactions and apply tensor factorization to reduce the parameters and computations.
- We use two level caches to intensively accelerate our probabilistic CKY-like parsing process.

The rest of this paper is organized as follows: Section 2 gives the details of our parsing model. Section 3 describes our parsing algorithm. Section 4 gives our training criterion. Section 5 reports the experimental results of our approach. Section 6 introduces the related work. Conclusions are given in section 7.

2 Parsing Model

Given two successive text spans, our parsing model evaluates the probability to combine them into a larger span, identifies which one is the nucleus and determines what is the relation between them. As with the work of Ji and Eisenstein (2014), we set three classifiers which share the same features as input to deal with those problems. The whole parsing model is shown in Figure 1. Three classifiers are on the top. The semantic representations of the two given text spans which come from the output of attention-based hierarchical bi-LSTM network with tensor-based transformation function is the main part of input to the classifiers. Additionally, following the previous practice of Li et al. (2014a), a small set of handcrafted features is introduced to enhance the model.

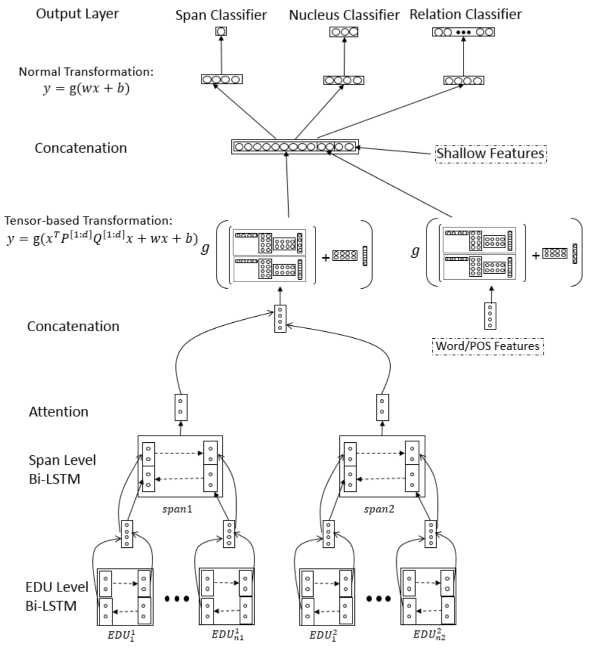

Figure 1: Schematic structure of our parsing model.

2.1 Hierarchical Bi-LSTM Network for Text Span Representations

Long Short-Term Memory (LSTM) networks have been successfully applied to a wide range of NLP tasks for the ability to handle long-term dependencies and to mitigate the curse of gradient vanishing (Hochreiter and Schmidhuber, 1997; Bahdanau et al., 2014; Rocktäschel et al., 2015; Hermann et al., 2015). A basic LSTM can be described as follows. A sequence $\{x_1, x_2, ..., x_n\}$ is given as input. At each time-step, the LSTM computation unit takes in one token x_t as input and it keeps some information in a cell state C_t and gives an output h_t. They are calculated in this way:

$$i_t = \sigma(W_i[h_{t-1}; x_t] + b_i) \tag{1}$$
$$f_t = \sigma(W_f[h_{t-1}; x_t] + b_f) \tag{2}$$
$$\tilde{C}_t = tanh(W_C[h_{t-1}; x_t] + b_C) \tag{3}$$
$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t \tag{4}$$
$$o_t = \sigma(W_o[h_{t-1}; x_t] + b_o) \tag{5}$$
$$h_t = o_t \odot tanh(C_t) \tag{6}$$

where $W_i, b_i, W_f, b_f, W_c, b_C, W_o, b_o$ are LSTM parameters, $\odot$ denotes element-wise product and σ denotes sigmoid function. The output at the last token, i.e., h_n is taken as the representation of the whole sequence.

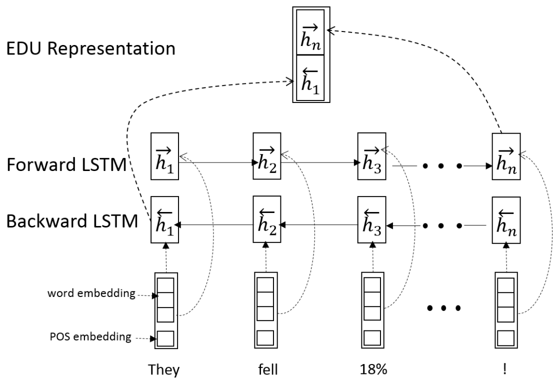
EDU Representation
$\overrightarrow{h_n}$
$\overleftarrow{h_1}$
Forward LSTM
Backward LSTM
$\overrightarrow{h_1}$ $\overrightarrow{h_2}$ $\overrightarrow{h_3}$ $\overrightarrow{h_n}$
$\overleftarrow{h_1}$ $\overleftarrow{h_2}$ $\overleftarrow{h_3}$ $\overleftarrow{h_n}$
word embedding
POS embedding
They
fell
18%
!

Figure 2: Bi-LSTM for computing the compositional semantic representation of an EDU.

Since an EDU is a sequence of words, we derive the representation of an EDU from the sequence constituted by concatenation of word embeddings and the POS tag embeddings of the words as Figure 2 shows. Previous work on discourse parsing tends to extract some features from the beginning and end of text units partly because discourse clues such as discourse markers(e.g., because, though) are often situated at the beginning or end of text units(Feng and Hirst, 2014; Ji and Eisenstein, 2014; Li et al., 2014a; Li et al., 2014b; Heilman and Sagae, 2015). Considering the last few tokens of a sequence normally have more influence on the representation of the whole sequence learnt with LSTM because they get through less times of forget gate from the LSTM computation unit, to effectively capture the information from both beginning and end of an EDU, we use bidirectional LSTM to learn the representation of an EDU. In other words, one LSTM takes the word sequence in forward order as input, the other takes the word sequence in reversed order as input. The representation of a sequence is the concatenation of the two vector representations calculated by the two LSTMs.

Since a text span is a sequence of EDUs, its meaning can be computed from the meanings of the EDUs. So we use another bi-LSTM to derive the compositional semantic representation of a text span from the EDUs it contains. The two bi-LSTM networks form a hierarchical structure as Figure 1 shows.

2.2 Attention

The representation of a sequence computed by bi-LSTMs is always a vector with fixed dimension despite the length of the sequence. Thus when dealing with a text span with hundreds of EDUs, bi-LSTM may not be enough to capture the whole semantic information with its limited output vector dimension. Attention mechanism can attend over the output at every EDU with global context and pick up prominent semantic information and drop the subordinate information for the compositional representation of the span, so we employ attention mechanism to alleviate the problem caused by the limited memory of LSTM networks. The attention mechanism is inspired by the work of Rocktäschel et al. (2015). Our attention-based bi-LSTM network is shown in Figure 3.

We combine the last outputs of the span level bi-LSTM to be $h_s = [\overrightarrow{h}_{e_n}, \overleftarrow{h}_{e_1}]$. We also combine the outputs of the two LSTM at every EDU of the span: $h_t = [\overrightarrow{h}_t, \overleftarrow{h}_t]$ and thus get a matrix $H = [h_1; h_2; ...; h_n]^T$. Taking $H \in \mathbb{R}^{d \times n}$ and $h_s \in \mathbb{R}^d$ as inputs, we get a vector $\alpha \in \mathbb{R}^n$ standing for weights of EDUs to the text span and use it to get a weighted representation of the span $r \in \mathbb{R}^d$:

$$M = tanh(W_y H + W_l h_s \otimes e_n) \quad (7)$$
$$\alpha = softmax(w_\alpha^T M) \quad (8)$$
$$r = H\alpha \quad (9)$$

where $\otimes$ denotes Cartesian product , $M \in \mathbb{R}^{k \times n}$, e_n is a n dimensional vector of all 1s and we use the Cartesian product $W_l h_s \otimes e_n$ to repeat the result of $W_l h_s$ n times in column to form a matrix and $W_y \in \mathbb{R}^{k \times d}$, $W_l, \in \mathbb{R}^{k \times d}$, $w_\alpha \in \mathbb{R}^k$ are parameters.

We synthesize the information of r and h_s to get the final representation of the span:

$$w_h = \sigma(W_{hr} r + W_{hh} h_s) \quad (10)$$
$$h = w_h \odot h_s + (1 - w_h) \odot r \quad (11)$$

where $W_{hr}, W_{hh} \in \mathbb{R}^{d \times d}$ are parameters, $w_h \in \mathbb{R}^d$ is a computed vector representing the element-wise weight of h_s and the element-wise weighted summation $h \in \mathbb{R}^d$ is the final representation of the text span computed by the attention-based bidirectional LSTM network.

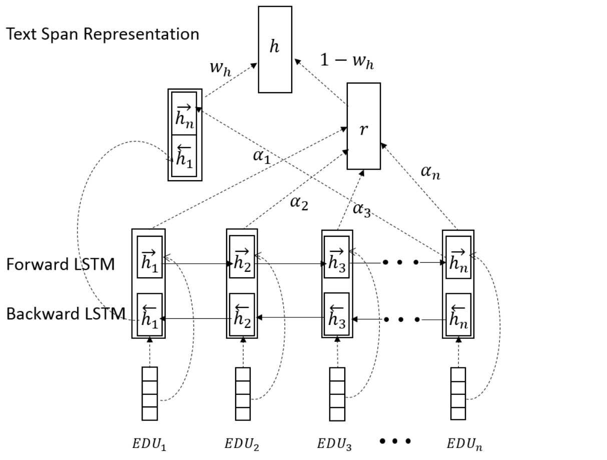

Figure 3: Attention-based bi-LSTM for computing the compositional semantic representation of a text span.

2.3 Classifiers

We concatenate the representations of the two given spans: $h = [h_{s1}, h_{s2}]$ and feed h into a full connection hidden layer to obtain a higher level representation v which is the input to the three classifiers:

$$v = Relu(W_h[h_{s1}, h_{s2}] + b_h) \quad (12)$$

For each classifier, we firstly transform $v \in \mathbb{R}^l$ into a hidden layer:

$$v_{sp} = Relu(W_{hs}v + b_{hs}) \quad (13)$$
$$v_{nu} = Relu(W_{hn}v + b_{hn}) \quad (14)$$
$$v_{rel} = Relu(W_{hr}v + b_{hr}) \quad (15)$$

where $W_{hs}, W_{hn}, W_{hr} \in \mathbb{R}^{h \times l}$ are transformation matrices and $b_{hs}, b_{hn}, b_{hr} \in \mathbb{R}^h$ are bias vectors.

Then we feed these vectors into the respective output layer:

$$y_{sp} = \sigma(w_s v_{sp} + b_s) \quad (16)$$
$$y_{nu} = softmax(W_n v_{nu} + b_n) \quad (17)$$
$$y_{rel} = softmax(W_r v_{rel} + b_r) \quad (18)$$

where $w_s \in \mathbb{R}^h, b_s \in \mathbb{R}, W_n \in \mathbb{R}^{3 \times h}, W_n \in \mathbb{R}^{3 \times h}, b_n \in \mathbb{R}^3, W_r \in \mathbb{R}^{n_r \times h}, b_n \in \mathbb{R}^{n_r}$ are parameters and n_r is the number of different discourse relations.

The first classifier is a binary classifier which outputs the probability the two spans should be combined. The second classifier is a multiclass classifier which identifies the nucleus to be span 1, span 2 or both. The third classifier is also a multiclass classifier which determines the relation between the two spans.

2.4 Tensor-based Transformation

Tensor-based transformation function has been successfully utilized in many tasks to allow complicated interaction between features (Sutskever et al., 2009; Socher et al., 2013; Pei et al., 2014). Based on the intuition that allowing complicated interaction between the features of the two spans may help to identify how they are related, we adopt tensor-based transformation function to strengthen our model.

A tensor-based transformation function on $x \in \mathbb{R}^{d_1}$ is as follows:

$$y = Wx + x^T T^{[1:d_2]} x + b \quad (19)$$
$$y_i = \sum_j W_{ij} x_j + \sum_{j,k} T^{[i]}_{j,k} x_j x_k + b_i \quad (20)$$

where $y \in \mathbb{R}^{d_2}$ is the output vector, $y_i \in \mathbb{R}$ is the ith element of y, $W \in \mathbb{R}^{d_2 \times d_1}$ is the transformation matrix, $T^{[1:d_2]} \in \mathbb{R}^{d_1 \times d_1 \times d_2}$ is a 3rd-order transformation tensor. A normal transformation function in neural network models only has the first term Wx with the bias term. It means for normal transformation function each unit of the output vector is the weighted summation of the input vector and this only allows additive interaction between the units of the input vector. With the tensor multiplication term, each unit of the output vector is augmented with the weighted summation of the multiplication of the input vector units and thus we incorporate multiplicative interaction between the units of the input vector.

Inevitably, the incorporation of tensor leads to side effects which include the increase in parameter number and computational complexity. To remedy this, we adopt tensor factorization in the same way as Pei et al. (2014): we use two low rank matrices to approximate each tensor slice $T^{[i]} \in \mathbb{R}^{d_1 \times d_1}$:

$$T^{[i]} \Rightarrow P^{[i]} Q^{[i]} \quad (21)$$

where $P^{[i]} \in \mathbb{R}^{d_1 \times r}$, $Q^{[i]} \in \mathbb{R}^{r \times d_1}$ and $r \ll d_1$. In this way, we drastically reduce parameter number and computational complexity.

We apply the factorized tensor-based transformation function to the combined text span representation $h = [h_{s1}, h_{s2}]$ to make the features of the two spans explicitly interact with each other:

$$v = Relu(W_h[h_{s1}, h_{s2}] + [h_{s1}, h_{s2}]^T P_h^{[1:d]} Q_h^{[1:d]} [h_{s1}, h_{s2}] + b_h) \quad (22)$$

Comparing with Eq. 12, the transformation function is added with a tensor term.

2.5 Handcrafted Features

Most previously proposed state-of-the-art systems heavily rely on handcrafted features (Hernault et al., 2010; Feng and Hirst, 2014; Joty et al., 2013; Ji and Eisenstein, 2014; Heilman and Sagae, 2015). Li et al. (2014a) show that some basic features are still necessary to get a satisfactory result for their recursive deep model. Following their practice, we adopt minimal basic features which are utilized by most systems to further strengthen our model. We list these features in Table 1. We apply the factorized tensor-based transformation function to Word/POS features to allow more complicated interaction between them.

3 Parsing Algorithm

In this section, we describe our parsing algorithm which utilizes the parsing model to produce the global optimal DT for a segmented document.

3.1 Probabilistic CKY-like Algorithm

We adopt a probabilistic CKY-like bottom-up algorithm which is also adopted in (Joty et al., 2013; Li et al., 2014a) to produce a DT for a document. This parsing algorithm is a dynamic programming algorithm and produces the global optimal DT with our parsing model. Given a text span which is constituted by $[e_i, e_{i+1}, ..., e_j]$ and the possible subtrees of $[e_i, e_{i+1}, ..., e_k]$ and $[e_{k+1}, e_{k+2}, ..., e_j]$ for all $k \in \{i, i{+}1, ..., j{-}1\}$ with their probabilities, we choose k and combine the corresponding subtrees to form a combined DT with the following recurrence formula:

$$k = \arg\max_k \{P_{sp}(i,k,j) P_{i,k} P_{k+1,j}\} \quad (23)$$

where $P_{i,k}$ and $P_{k+1,j}$ are the probabilities of the most probable subtrees of $[e_i, e_{i+1}, ..., e_k]$ and $[e_{k+1}, e_{k+2}, ..., e_j]$ respectively, $P_{sp}(i,k,j)$ is the probability which is predicted by our parsing model to combine those two subtrees to form a DT.

The probability of the most probable DT of $[e_i, e_{i+1}, ..., e_j]$ is:

$$P_{i,j} = \max_k \{P_{sp}(i,k,j) P_{i,k} P_{k+1,j}\} \quad (24)$$

3.2 Parsing Acceleration

Computational complexity of the original probabilistic CKY-like algorithm is $O(n^3)$ where n is the number of EDUs of the document. But in this work, given each pair of text spans, we compute the representations of them with hierarchical bi-LSTM network at the expense of an additional $O(n)$ computations. So the computational complexity of our parser becomes $O(n^4)$ and it is unacceptable for long documents. However, most computations are duplicated, so we use two level caches to drastically accelerate parsing.

Firstly, we cache the outputs of the EDU level bi-LSTM which are the semantic representations of EDUs. As for the forward span level LSTM, after we get the semantic representation of a span, we cache it too and use it to compute the representation of an extended span. For example, after we get the representation of span constituted by $[e_1, e_2, e_3]$, we take it with semantic representation of e_4 to compute the representation of the span constituted by $[e_1, e_2, e_3, e_4]$ in one LSTM computation step. For the backward span level LSTM, we do it the same way just in reversed order. Thus we decrease the computational complexity of computing the semantic representations for all possible span pairs which is the most time-consuming part of the original parsing process from $O(n^4)$ to $O(n^2)$.

Secondly, it can be seen that before we apply $Relu$ to the tensor-based transformation function, many calculations from the two spans which include a large part of tensor multiplication are independent. The multiplication between the elements of the representations of the two spans caused by the tensors and the element-wise non-linear activation function $Relu$ terminate the independence between them. So we can further cache the independent calculation results before $Relu$ operation for each span. Thus we decrease the computational complexity of a large part of tensor-based transformation from $O(n^3)$ to

Word/POS Features
One-hot representation of the first two words and of the last word of each span.
One-hot representation of POS tags of the first two words and of the last word of each span.
Shallow Features
Number of EDUs of each span.
Number of words of each span.
Predicted relations of the two subtrees' roots.
Whether each span is included in one sentence.
Whether both spans are included in one sentence.

Table 1: Handcrafted features used in our parsing model.

$O(n^2)$ which is the second time-consuming part of the original parsing process.

The remaining $O(n^3)$ computations include a little part of tensor-based transformation computations, $Relu$ operation and the computations from the three classifiers. These computations take up only a little part of the original parsing model computations and thus we greatly accelerate our parsing process.

4 Max-Margin Training

We use Max-Margin criterion for our model training. We try to learn a function that maps: $X \rightarrow Y$, where X is the set of documents and Y is the set of possible DTs. We define the loss function for predicting a DT $\hat{y_i}$ given the correct DT y_i as:

$$\triangle(y_i, \hat{y_i}) = \sum_{r \in \hat{y_i}} \kappa \mathbf{1}\{r \notin y_i\} \tag{25}$$

where r is a span specified with nucleus and relation in the predicted DT, κ is a hyperparameter referred to as discount parameter and **1** is indicator function. We expect the probability of the correct DT to be a larger up to a margin to other possible DTs:

$$Prob(x, y_i) \geq Prob(x_i, \hat{y_i}) + \triangle(y_i, \hat{y_i}) \tag{26}$$

The objective function for m training examples is as follows:

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} l_i(\theta), where \tag{27}$$

$$l_i(\theta) = \max_{\hat{y_i}}(Prob(x_i, \hat{y_i}) + \triangle(y_i, \hat{y_i})) - Prob(x_i, y_i) \tag{28}$$

where θ denotes all the parameters including our neural network parameters and all embeddings.

The probabilities of the correct DTs increase and the probabilities of the most probable incorrect DTs decrease during training. We adopt Adadelta (Zeiler, 2012) with mini-batch to minimize the objective function and set the initial learning rate to be 0.012.

5 Experiments

We evaluate our model on RST Discourse Treebank[1] (RST-DT) (Carlson et al., 2003). It is partitioned into a set of 347 documents for training and a set of 38 documents for test. Non-binary relations are converted into a cascade of right-branching binary relations. The standard metrics of RST-style discourse parsing evaluation include blank tree structure referred to as span (S), tree structure with nuclearity (N) indication and tree structure with rhetorical relation (R) indication. Following other RST-style discourse parsing systems, we evaluate the relation metric in 18 coarse-grained relation classes. Since our work focus does not include EDU segmentation, we evaluate our system with gold-standard EDU segmentation and we apply the same setting on this to other discourse parsing systems for fair comparison.

5.1 Experimental Setup

The dimension of word embeddings is set to be 50 and the dimension of POS embeddings is set to be 10. We pre-trained the word embeddings with GloVe (Pennington et al., 2014) on English Gigaword[2] and we fine-tune them during training. Considering some words are pretrained by GloVe but

[1]https://catalog.ldc.upenn.edu/LDC2002T07
[2]https://catalog.ldc.upenn.edu/LDC2011T07

don't appear in the RST-DT training set, we want to use their embeddings if they appear in test set. Following Kiros et al. (2015), we expand our vocabulary with those words using a matrix $W \in \mathbb{R}^{50\times 50}$ that maps word embeddings from the pre-trained word embedding space to the fine-tuned word embedding space. The objective function for training the matrix W is as follows:

$$\min_{W,b} ||V_{tuned} - V_{pretrained}W - b||_2^2 \qquad (29)$$

where $V_{tuned}, V_{pretrained} \in \mathbb{R}^{|V|\times 50}$ contain fine-tuned and pre-trained embeddings of words appearing in training set respectively, $|V|$ is the size of RST-DT training set vocabulary and b is the bias term also to be trained.

We lemmatize all the words appeared and represent all numbers with a special token. We use Stanford CoreNLP toolkit (Manning et al., 2014) to preprocess the text including lemmatization, POS tagging etc. We use Theano library (Bergstra et al., 2010) to implement our parsing model. We randomly initialize all parameters within (-0.012, 0.012) except word embeddings. We adopt dropout strategy (Hinton et al., 2012) to avoid overfitting and we set the dropout rate to be 0.3.

5.2 Results and Analysis

To show the effectiveness of the components incorporated into our model, we firstly test the performance of the basic hierarchical bidirectional LSTM network without attention mechanism (ATT), tensor-based transformation (TE) and handcrafted features (HF). Then we add them successively. The results are shown in Table 2.

The performance is improved by adding each component to our basic model and that shows the effectiveness of attention mechanism and tensor-based transformation function. Even without handcrafted features, the performance is still competitive. It indicates that the semantic representations of text spans produced by our attention-based hierarchical bi-LSTM network are effective and the handcrafted features are complementary to semantic representations produced by the network.

We also experiment without mapping the OOV word embeddings and use the same embedding for all OOV words. The result is shown in Table 3. Without mapping the OOV word embeddings the performance decreases slightly, which demonstrates that the relation between pre-trained embedding space and the fine-tuned embedding space can be learnt and it is beneficial to train a matrix to transform OOV word embeddings from the pre-trained embedding space to the fine-tuned embedding space.

System Setting	S	N	R
Basic	82.7	69.7	55.6
Basic+ATT	83.6*	70.2*	56.0*
Basic+ATT+TE	84.2*	70.4	56.3*
Basic+ATT+TE+HF	85.8*	71.1*	58.9*

Table 2: Performance comparison for different settings of our system on RST-DT. 'Basic' denotes the basic hierarchical bidirectional LSTM network; '+ATT' denotes adding attention mechanism; '+TE' denotes adopting tensor-based transformation; '+HF' denotes adding handcrafted features. * indicates statistical significance in t-test compared to the result in the line above ($p < 0.05$).

System Setting	S	N	R
Without OOV mapping	85.1	70.7	58.2
Full version	85.8*	71.1*	58.9*

Table 3: Performance comparison for whether to map OOV embeddings.

We compare our system with other state-of-the-art systems including (Joty et al., 2013; Ji and Eisenstein, 2014; Feng and Hirst, 2014; Li et al., 2014a; Li et al., 2014b; Heilman and Sagae, 2015). Systems proposed by Joty et al. (2013), Heilman (2015) and Feng and Hirst (2014) are all based on variants of CRFs. Ji and Eisenstein (2014) use a projection matrix acting on one-hot representations of features to learn representations of text spans and build Support Vector Machine (SVM) classifier on them. Li et al. (2014b) adopt dependency parsing methods to deal with this task. These systems are all based on handcrafted features. Li et al. (2014a) adopt a recursive deep model and use some basic handcrafted features to improve their performances which has been stated before.

Table 4 shows the performance for our system and those systems. Our system achieves the best result in span and relatively lower performance in nucleus and relation identification comparing with the corresponding best results but still better than

System	S	N	R
Joty et al. (2013)	82.7	68.4	55.7
Ji and Eisenstein (2014)	82.1	71.1	**61.6**
Feng and Hirst (2014)	**85.7**	71.0	58.2
Li et al. (2014a)	84.0	70.8	58.6
Li et al. (2014b)	83.4	**73.8**	57.8
Heilman and Sagae (2015)	83.5	68.1	55.1
Ours	**85.8**	71.1	58.9
Human	88.7	77.7	65.8

Table 4: Performance comparison with other state-of-the-art systems on RST-DT.

System	S	N	R
Li et al. (2014a) (no feature)	82.4	69.2	56.8
Ours (no feature)	84.2	70.4	56.3

Table 5: Performance comparison with the deep learning model proposed in Li et al. (2014a) without handcrafted features.

most systems. No system achieves the best result on all three metrics. To further show the effectiveness of the deep learning model itself without handcrafted features, we compare the performance between our model and the model proposed by Li et al. (2014a) without handcrafted features and the results are shown in Table 5. It shows our overall performance outperforms the model proposed by Li et al. (2014a) which illustrates our model is effective.

Table 6 shows an example of the weights (W) of EDUs (see Eq. 8) derived from our attention model. For span1 the main semantic meaning is expressed in EDU32 under the condition described in EDU31. Besides, it is EDU32 that explicitly manifests the contrast relation between the two spans. As can be seen, our attention model assigns less weight to EDU30 and focuses more on EDU32 which is reasonable according to our analysis above.

Span1 (EDU30∼EDU32)	W
That means that	0.13
if the offense deals with one part of the business,	0.38
you don't attempt to seize the whole business;	0.49
Span2 (EDU33)	W
you attempt to seize assets related to the crime,	1.0

Table 6: An example of the weights derived from our attention model. The relation between span1 and span2 is Contrast.

6 Related Work

Two most prevalent discourse parsing treebanks are RST Discourse Treebank (RST-DT) (Carlson et al., 2003) and Penn Discourse TreeBank (PDTB) (Prasad et al., 2008). We evaluate our system on RST-DT which is annotated in the framework of Rhetorical Structure Theory (Mann and Thompson, 1988). It consists of 385 Wall Street Journal articles and is partitioned into a set of 347 documents for training and a set of 38 documents for test. 110 fine-grained and 18 coarse-grained relations are defined on RST-DT. Parsing algorithms published on RST-DT can mainly be categorized as shift-reduce parsers and probabilistic CKY-like parsers. Shift-reduce parsers are widely used for their efficiency and effectiveness and probabilistic CKY-like parsers lead to the global optimal result for the parsing models. State-of-the-art systems belonging to shift-reduce parsers include (Heilman and Sagae, 2015; Ji and Eisenstein, 2014). Those belonging to probabilistic CKY-like parsers include (Joty et al., 2013; Li et al., 2014a). Besides, Feng and Hirst (2014) adopt a greedy bottom-up approach as their parsing algorithm. Lexical, syntactic, structural and semantic features are extracted in these systems. SVM and variants of Conditional Random Fields (CRFs) are mostly used in these models. Li et al. (2014b) distinctively propose to use dependency structure to represent the relations between EDUs. Recursive deep model proposed by Li et al. (2014a) has been the only proposed deep learning model on RST-DT.

Incorporating attention mechanism into RNN (e.g., LSTM, GRU) has been shown to learn better representation by attending over the output vectors and picking up important information from relevant positions of a sequence and this approach has been utilized in many tasks including neural machine translation (Kalchbrenner and Blunsom, 2013; Bahdanau et al., 2014; Hermann et al., 2015), text entailment recognition (Rocktäschel et al., 2015) etc. Some work also uses tensor-based transformation function to make stronger interaction between features and learn combinatorial features and they get performance boost in their tasks (Sutskever et al., 2009; Socher et al., 2013; Pei et al., 2014).

7 Conclusion

In this paper, we propose an attention-based hierarchical neural network for discourse parsing. Our attention-based hierarchical bi-LSTM network produces effective compositional semantic representations of text spans. We adopt tensor-based transformation function to allow complicated interaction between features. Our two level caches accelerate parsing process significantly and thus make it practical. Our proposed system achieves comparable results to state-of-the-art systems. We will try extending attention mechanism to obtain the representation of a text span by referring to another text span at minimal additional cost.

Acknowledgments

We thank the reviewers for their instructive feedback. We also thank Jiwei Li for his helpful discussions. This work is supported by National Key Basic Research Program of China under Grant No.2014CB340504 and National Natural Science Foundation of China under Grant No.61273318. The Corresponding author of this paper is Baobao Chang.

References

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

James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio. 2010. Theano: a CPU and GPU math expression compiler. In *Proceedings of the Python for Scientific Computing Conference (SciPy)*, June. Oral Presentation.

Lynn Carlson, Daniel Marcu, and Mary Ellen Okurowski. 2003. Building a discourse-tagged corpus in the framework of rhetorical structure theory. In *Current and new directions in discourse and dialogue*, pages 85–112. Springer.

Vanessa Wei Feng and Graeme Hirst. 2014. A linear-time bottom-up discourse parser with constraints and post-editing. In *ACL (1)*, pages 511–521.

Michael Heilman and Kenji Sagae. 2015. Fast rhetorical structure theory discourse parsing. *CoRR*, abs/1505.02425.

Karl Moritz Hermann, Tomá s Kociský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. *CoRR*, abs/1506.03340.

Hugo Hernault, Helmut Prendinger, David A DuVerle, and Mitsuru Ishizuka. 2010. Hilda: a discourse parser using support vector machine classification. *Dialogue and Discourse*, 1(3):1–33.

Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, abs/1207.0580.

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

Yangfeng Ji and Jacob Eisenstein. 2014. Representation learning for text-level discourse parsing. In *ACL (1)*, pages 13–24.

Shafiq R. Joty, Giuseppe Carenini, Raymond T. Ng, and Yashar Mehdad. 2013. Combining intra- and multi-sentential rhetorical parsing for document-level discourse analysis. In *ACL*.

Daniel Jurafsky and James H Martin. 2008. Speech and language processing, chapter 14. In *Prentice Hall*.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *EMNLP*.

Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S. Zemel, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. 2015. Skip-thought vectors. *CoRR*, abs/1506.06726.

Jiwei Li, Rumeng Li, and Eduard H Hovy. 2014a. Recursive deep models for discourse parsing. In *EMNLP*, pages 2061–2069.

Sujian Li, Liang Wang, Ziqiang Cao, and Wenjie Li. 2014b. Text-level discourse dependency parsing. In *ACL (1)*, pages 25–35.

Annie Louis, Aravind Joshi, and Ani Nenkova. 2010. Discourse indicators for content selection in summarization. In *Proceedings of the 11th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 147–156. Association for Computational Linguistics.

William C Mann and Sandra A Thompson. 1988. Rhetorical structure theory: Toward a functional theory of text organization. *Text-Interdisciplinary Journal for the Study of Discourse*, 8(3):243–281.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *ACL*.

Daniel Marcu. 2000. *The theory and practice of discourse parsing and summarization*. MIT press.

Wenzhe Pei, Tao Ge, and Baobao Chang. 2014. Max-margin tensor neural network for chinese word segmentation. In *ACL (1)*, pages 293–303.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*.

Rashmi Prasad, Nikhil Dinesh, Alan Lee, Eleni Miltsakaki, Livio Robaldo, Aravind K Joshi, and Bonnie L Webber. 2008. The penn discourse treebank 2.0. In *LREC*. Citeseer.

Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomá s Kociský, and Phil Blunsom. 2015. Reasoning about entailment with neural attention. *CoRR*, abs/1509.06664.

Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*, volume 1631, page 1642. Citeseer.

Swapna Somasundaran. 2010. *Discourse-level relations for Opinion Analysis*. Ph.D. thesis, University of Pittsburgh.

Ilya Sutskever, Ruslan Salakhutdinov, and Joshua B. Tenenbaum. 2009. Modelling relational data using bayesian clustered tensor factorization. In *NIPS*.

Suzan Verberne, Lou Boves, Nelleke Oostdijk, and Peter-Arno Coppen. 2007. Evaluating discourse-based answer extraction for why-question answering. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 735–736. ACM.

Matthew D. Zeiler. 2012. Adadelta: An adaptive learning rate method. *CoRR*, abs/1212.5701.