Sentence Representation Learning for Text-Level Discourse Parsing

Abstract

Discourse structures have a central role in several NLP tasks, such as dialogue generation or discourse translation. Also, Text-Level discourse parsing is notoriously difficult for the long distance of discourse and deep structure of discourse trees. In this paper, we take discourse parsing task back to tree's representation problem and build a tree-structured model for discourse parsing. We compare those popular methods for sentence representation along with a tree structured neural network in RST discourse parsing task.

Key word: sentence representation, RST-DT, discourse parsing, tree LSTM

Introduction

Documents are usually formed as a long sequence text which could also be analyzed as constituency trees, as shown in figure 1. Discourse structures can describe the organization of a document detailedly which is central to a number of NLP applications like sentiment analysis (Voll et al., 2007), text summarization (Louise et al., 2010) and question answering (Ferrucci et al., 2010).

Rhetorical Structure Theory (RST) (Mann and Thompson, 1988) is a representative linguistic theory of discourse structures, This theory guided the annotation of the RST Discourse Treebank (RST-DT) (Carlson et al., 2001) , from which several text level discourse parsers have been proposed (Hernault et al., 2010; Feng and Hirst, 2014; Wang and Li., 2017; Braud, 2017).

However, the RST-DT discourse corpora are limited in size, since annotation is time consuming and complex. Many measures have been taken to solve the problem. For example, Braud (2017) harmonized some existing corpora to leverage information by combining dataset in different languages. Due to the limitation of training data, the methods of discourse parsing are still weak: those state-of-the-art jobs at present like DPLP (2014) and Wang (2017) are still based on traditional methods like support vector machine (SVM) instead of neural networks. We think it meaningful to do this research over neural network based discourse parsing for the fast development of Deep Learning.

In this paper, our model is implemented as shift-reduce discourse parser. The core idea of our work is to learn better representation for each subtree, so we present a tree-structured neural network to discourse parsing. What's more, sentence representation is inevitable an essential part in deep models. We will compare those popular methods for sentence representation along with this neural network strictly. We make the code and preprocessing scripts available for download at https://github.com/ArlenZhang/Data_share.

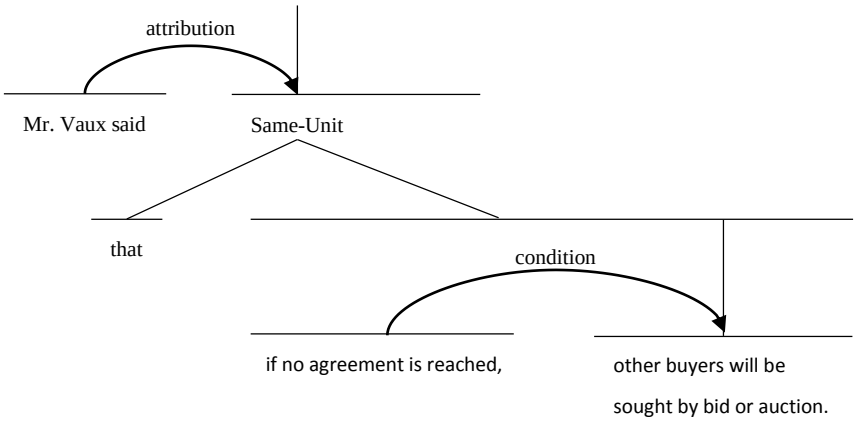

Figure 1

2 Related Work

The task of discourse parsing are mainly divided into two aspects. The first one focus on relation recognition, the other is full discourse parsing that identifies discourse relation and structure. The RST discourse parsing focus on the later one and many RST discourse parsers have been proposed up to now. The first text-level discourse parser relays mainly on heuristics and hand-crafted rules (Marcu, 2000a; Carlson et al., 2001). Hernaulth et al.(HILDA, 2010) built a greedy model using SVM to transform this task into a labeling decision problem. Feng and Hirst (2014) built a bottom-up, two stage(sentence- then document-level), greedy parser with linear-chain CRF models.

Recently, many more researchers have focused on building models with good representations of the limited data. Li et al. (2014) used a recursive neural network that builds a representation for each clause based on the syntactic tree, and then apply two classifieds as in Hernault et al.(HILDA, 2010). The system presented by Ji and Eisenstein (2014 DPLP) jointly learns the representation of the discourse units and a shift-reduce parser. This system, however, uses Tree LSTM to represent each discourse tree node and focus on the representation learning for each EDUs.

3 Parsing Model

3.1 Background: Shift-reduce parsing

A shift-reduce discourse parser maintains two data structures: a stack S of partially completed subtrees and a buffer B of EDUs yet to be parsed. The parser is initialized with the stack empty and the buffer contains the EDUs $e = (e_1, e_2 \dots e_n)$ of the document in order. During parsing process, the parser consumes transitions $a = (a_1, a_2, a_k, \dots a_{2n-1})$ constantly, where $a_k \in \{\text{Shift}, \text{Reduce}\}$. The state of buffer and stack will change according to the predicted action label. The shift-reduce parsing procedure is detailedly described in table 1.

Algorithm 1 Shift-reduce discourse parsing

Input: EDUs of a discourse $[e_1, e_2 \dots e_n]$

$Initial_{state} = [\text{Stack}, \text{Queue}] = [[\Theta], [e_1, e_2 \dots e_n]]$

while Queue has more than 0 element **or** Stack has more than 1 element **do**
  Action = Predict the action label according to c_s
  if Action is SHIFT **then**
    Pop an element from Queue and push it onto Stack
  else
    Pop tow elements e_{-1} and e_{-2} from Stack
    Merge e_{-1} and e_{-2} into e_{new}
    Push e_{new} onto Stack
  endif
endwhile
Pop the last element e from Stack
Output: A discourse parsing tree e

Table 1:

3.2 Elements in Buffer

We use a learned linear transformation to map all these encoded EDUs of a document $\vec{x}_{E\_encoded}$ into a vector pair $\langle \overrightarrow{h_b}, \overrightarrow{c_b} \rangle$ that stored in the buffer. The vector $\vec{x}_{E\_encoded}$ of those encoded EDUs will be display in section 4.

$$\begin{bmatrix} \overrightarrow{h_b} \\ \overrightarrow{c_b} \end{bmatrix} = W_{EDU}\vec{x}_{E\_encoded} + \vec{b}_{EDU}$$

3.3 Composition Function for Reducing

When the reduce action is performed, the vector representations of two tree nodes are popped off the stack and fed into the following composition function. This function is a neural network function that produces a representation for the parent node. The parent node is then pushed on to the stack to form a new state.

The composition function of us function like the Tree LSTM composition function(Tai 2015). It generalizes the LSTM neural network to tree. It still maintains a memory cell c and a hidden state

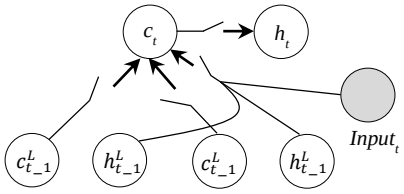

Figure 2:

vector h as what we do in LSTM. The bottom-up Tree-LSTM (Zhu et al., 2015) extends the sequence LSTM by splitting the previous state vector $\vec{h}_{t-1}$ into a left child state vector $\vec{h}^L_{t-1}$ and a right child state vector $\vec{h}^R_{t-1}$, calculating $\vec{h}_t$ as :

$$\vec{c}_t = \sum_{N\in\{L,\,R\}} \vec{f}^N_{t-1} \odot \vec{c}^L_{t-1} + \vec{\imath}_t \odot (\boldsymbol{W_{xg}}\vec{\boldsymbol{x}}_{\boldsymbol{t}} + \sum_{N\in\{L,\,R\}} W^N_{hg}\vec{h}^N_{t-1} + b_g)$$

$$\vec{h}_t = \vec{o}_t \odot \tanh(\vec{c}_t)$$

where $\vec{\boldsymbol{x}}_{\boldsymbol{t}}$ is an optional vector valued input argument which is either empty or comes from an external source like the trackers (see section 3.4), $\vec{f}^N_{t-1}$ is the forget gate, $\vec{o}_t$ is the output gate, and $\odot$ is the elementwise product. The new pair $< \vec{h}_t, \vec{c}_t >$ which represents a subtree is placed on

the stack.

3.4 Trackers

State tracker (Samuel, 2016) and connective tracker are two simple sequence-based LSTM. The state tracker's inputs at the m^{th} step are the top element of the buffer $\vec{h}_b^1$ and the top two elements of the stack $\vec{h}_s^1$ and $\vec{h}_s^2$. The connective tracker's input at the m^{th} step is the connectives in the EDUs to be shift in the buffer. So, the connective works only when the next transition is SHIFT. We combine the outputs of two trackers into $h_{composed}$. We use these two tracker in two purpose: the composed hidden states $h_{composed}$ gives a representation of the $< Stack, Buffer >$ for transition classifier and it is also used as a input vector $\vec{x}_t$ for the composition function (see 3.3).

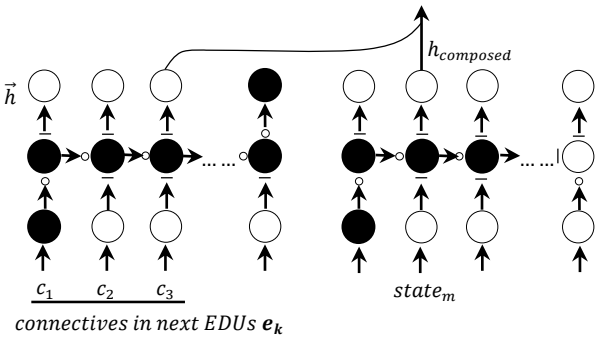

Figure 3:

4 Methods for EDUs Representation

We use word representation based on the 100D vectors provided with GloVe (Pennington et al., 2014) and Siamese(Tom et al., 2016). We do not update these representations during training for we aim at comparing these encoders strictly. We still use POS embedding which is obtained over a whole sentence as a supplement representation for word. In this paper, we focus on finding out a most appropriate sentence encoder for EDUs representation. These most popular encoders used in our experiments are listed as follows.

4.1 CNN based encoder

CNN () is an efficient model for encoding, it can capture some important local information of a sentence which could be helpful in structure building. As we know, sometimes, it is lexical information like connectives, specific words and phrase that counts. As we care about word level vector information when we apply CNN in NLP, so we set the filter size as $(k, embedding\_size)$. The structure of CNN encoder is designed in figure 2.

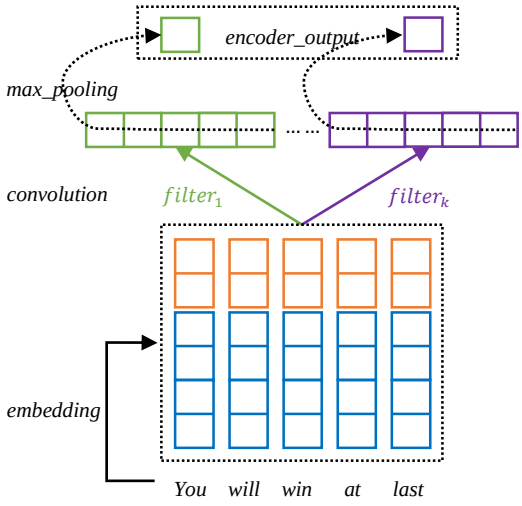

Figure 4:

4.2 Bi LSTM & Self-Attention based encoder

Bi LSTM is a variant of RNN, it is meant to maintain a low-resolution summary of the portion of the sentence has been processed so far. However, it only provides a final hidden state to represent the sentence which will be invalid for we need to capture some decisive local hidden state. Self-attention mechanism will put attention on all the hidden states of a sequence and can extract relevant information of the sequence (Lin et al., 2017). It performs well on many tasks like Reading comprehension (Cheng et al., 2016)、Text inheritance (textual entailment/Parikh et al., 2016) [39]、Text summary (Paulus et al., 2017) [40] and so on. So, we introduce Self-Attention into this Bi LSTM encoder in that we think it should be helpful in EDUs encoding for discourse parsing. We formulate our vision of attention as:

$$\mathrm{H} = \mathrm{BiLSTM}(EDUs\_embedding)$$

$$encoder\_output = \mathrm{softmax}(\mathrm{V_a}\tanh(\mathrm{W_a}\mathrm{H^T}))\,\mathrm{H}$$

where H is the hidden state values of the Bi LSTM model whose input is an EDUs, $\mathrm{W_a}$ and $\mathrm{V_a}$ are weights to learn.

4.3 Bag of word model

Continuous Bag of Word model is a neural network for efficient estimation of high-quality sentence embedding proposed by Siamese (2016). **Average** the embedding of words in a sentence has proven to be a surprisingly successful and efficient way of obtaining sentence embedding. In this paper, we use the Bag of Word (BOW) model to represent EDUs only to find that it is really helpful in our data driven discourse parser. We describe the representation of EDU_i as:

$$EDUs_i = \mathrm{average}(\sum_{j=1}^{padding_{length}} w_{ij})$$

where w_{ij} is the $\mathrm{j^{th}}$ word embedding weight of the $\mathrm{i^{th}}$ EDUs.

4.4 Features for Data Driven Model

Feature engineer is a traditional way of discourse parsing and many classic methods are based on this. Nowadays, with the rapid development of Deep Learning, features are still useful in face of limited data. The most state-of-art parsers all depend on features like position, length, dependency and so on. Wang (2017) extract these statistic features to learn a SVM classifier which achieves the

highest score at the level of Span. Ji (DPLP, 2014) proposed a discourse parser which is still a state-of-art parser at the level of Relation. They represent these extracted feature by multiplying with a specific weights matrix which looks just like a data driven model. However, they use these representations to learn a SVM classifier which is still a traditional method. Braud et al., (2017) uses almost the same features as DPLP and use these feature to learn a feed-forward neural network. They proposed a cross-lingual discourse parser which extends the data we use. So, it is hard to say that their highest score at the level of Nuclearity due to the neural network they use.

Feature
Words at the beginning and end of the EDU
POS tag at the beginning and end of the EDU
The head words the first three words in the EDU
Boolean value of whether the center word of temporary sentence is in the temporary EDUs
Length of EDU in tokens **(l1<=5, 5<l2<=12, 12<l3)**
Position of EDUs in a discourse **(percentage)**

Table 2: Additional features for RST parsing

These successful parsers have a similarity that they all use some easy but effective feature to represent EDUs. So, we select some classical features in table2. The usual way of using these feature in neural network is embedding learning. We combine these features and obtain **18** feature labels. We assign these feature with different embedding vectors and let the model to learn these vectors themselves.

5 Experiments

5.1 Data

The RST-DT (Carlson and Marcu, 2001) annotates 385 documents from the Wall Street Journal. It was mainly divided into 2 data sets (347 for training and 38 for testing). Conventionally, we binarize those non-binary sub-trees with right-branching () and use 18 coarse-grained relations as our relation set. We use the evaluation metrics defined by Marcu (2000), i.e. the precision, recall and F-score with respect to span and nuclearity.

5.2 Results and Analysis

In this paper, we aim at finding a better way of EDUs representation for neural network based discourse parsing. We hold the purpose that … is better than … in discourse parsing, so we put our attention only at the level Span and Nuclearity in the paper.

Encoding Model	S	N
CNN		
Bi LSTM + Self-attention		
BOW		
Features Embedding Learning		

Table 3:

We choose the Ji (DPLP, 2014) to compare with in three reasons: 1. the DPLP has a state-of-art result in discourse parsing; 2. it is also a data driven model different from others; 3. Our parsers are all built on the limited data RST-DT without other external data.

Comparing with Ji (DPLP, 2014), our discourse parsing based on tree-structured neural

network is a dynamic process, we don’t need to prepare all data before. We introduce a novel Tree structured neural network instead of SVM in DPLP. What’s more, DPLP learns the representation matrix and the SVM classifier at the same time which is more complex than ours. From the F-score at the Span level, our parser are better than theirs.

Model	S	N
CNN		
Bi LSTM + Self-attention		
BOW		
DPLP 2014		

Table 4:

6 Conclusion and future work

In this paper, we build a

Reference

Yangfeng Ji and Tacob Eisenstein. 2014. Representation learning for text-level discourse parsing. In Proceedings of ACL.

Chloe Braud, Maximin Coavoux, and Anders Sogaard. 2017. Cross-lingual RST Discourse Parsing.

Jiwei Li, Rumeng Li, and Eduard Hovy. 2014 Recursive Deep Models for Dsicourse Parsing.

Yizhong Wang, Sujian Li, and Houfeng Wang. 2017. A Two-Stage Parsing Method for Text-Level Discourse Analysis.

Samuel R. Bowman, Jon Gauthier, and Abhinav Rastogi. 2016. A Fast Unified Model for Parsing and Sentence Understanding.

Tom Kenter, Alexey Borisov, and Maarten de Rijke. 2016. Siamese CBOW: Optimizing Word Embeddings for Sentence Representations.

William C. Mann and Sandra A. Thompson. 1988. Rhetorical Structure Theory: Toward a functional theory of text organization. Text, 8:243–281.

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.

Kimberly Voll and Maite Taboada. 2007. Not all words are created equal: Extracting semantic orientation as a function of adjective relevance. In Proceedings of Australian Conference on Artificial Intelligence.

David Ferrucci, Eric Brown, Jennifer Chu-Carroll, James Fan, David Gondek, Aditya A Kalyanpur, Adam Lally, J William Murdock, Eric Nyberg, John Prager, et al. 2010. Building Watson: An overview of the DeepQA project. AI magazine, 31(3):59–79.

Lynn Carlson and Daniel Marcu. 2001. Discourse tagging reference manual. Technical report, University of Southern California Information Sciences Institute.

Hugo Hernault, Helmut Prendinger, David A. duVerle, and Mitsuru Ishizuka. 2010. HILDA: A discourse parser using support vector machine classification. Dialogue and Discourse, 1:1–33.

Vanessa Wei Feng and Graeme Hirst. 2014. A linear-time bottom-up discourse parser with constraints and post-editing. In Proceedings of ACL.

Daniel Marcu. 2000a. The rhetorical parsing of unrestricted texts: A surface-based approach. Computational Linguistics.

Xiaodan Zhu, Parinaz Sobhani, and Hongyu Guo. 2015. Long short-term memory over recursive structures. In *Proceedings of the 23nd International Conference on Machine Learning*, page 1604-1612, Lille, France, July.