EDUs Representation Learning for RST Discourse Parsing

**Abstract.** Discourse structure has 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 RST discourse parsing. What’s more, we compare those popular methods for sentence representation along with a tree structured neural network to find a best EDUs encoder in RST discourse parsing task.

**Keywords:** Discourse Parsing, RST-DT, Tree LSTM, EDUs Representation.

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 detailedly describe the organization of a document which is central to a number of NLP applications like sentiment analysis [1], text summarization [2] and question answering [3].

Rhetorical Structure Theory (RST) [4] is a representative linguistic theory of discourse structures, This theory guided the annotation of the RST Discourse Treebank (RST-DT) [5], from which several text level discourse parsers have been proposed [6, 7, 8, 9].

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 et al. [9] 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 Ji et al. [10] and Wang et al. [8] 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 compare those popular methods for sentence representation along with our neural network based discourse parser strictly to find a best way of EDUs representation. We make the main codes available for download at <https://github.com/ArlenZhang/Data_share>.

**Fig. 1.** An example of RST discourse structure

Mr. Vaux said

that

if no agreement is reached,

other buyers will be

sought by bid or auction.

condition

Same-Unit

attribution

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 [11, 22]. Hernaulth et al. [6] built a greedy model using SVM to transform this task into a labeling decision problem. Feng and Hirst [7] 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. [12] used a recursive model that builds a representation for each EDUs based on the syntactic tree, and then trained two classifiers as Hernault et al. [6]**.** The system presented by Ji and Eisenstein [10] jointly learns the representation of the discourse units and a shift-reduce parser. This system, however, uses bottom-up Tree LSTM [13] to represent each discourse tree node and focus on the representation learning for each EDU.

3 Parsing Model

3.1 Background: Shift-reduce parsing

A shift-reduce discourse parser [maintain](file:///D:\%E6%9C%89%E9%81%93%E8%AF%8D%E5%85%B8%E5%AE%89%E8%A3%85%E8%B7%AF%E5%BE%84\Dict\7.2.0.0703\resultui\dict\?keyword=maintain)s two data structures: a stack of partially completed subtrees and a buffer of EDUs yet to be parsed. The parser is initialized with the stack empty and the buffer contains the EDUs of the document in order. During parsing process, the parser consumes transitions constantly, where . 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.

**Table 1.** Shift-reduce discourse parsing

|  |
| --- |
| **Algorithm 1** Shift-reduce discourse parsing |
| **Input:** EDUs of a discourse    **while** buffer has more than 0 element **or** stack has more than 1 element **do**  = Predict the action label according to  **if** is **then**  Pop an element from and push it onto  **else**  Pop tow elements and from  Merge and into  Push onto  **endif**  **endwhile**  Pop the last element from  **Output:** A discourse parsing tree |

3.2 Elements in Buffer

We use a learned linear transformation to map all these encoded EDUs of a document into a vector pair that stored in the buffer. The vector of those encoded EDUs will be display in section 4.

|  |  |  |
| --- | --- | --- |
|  |  | (1) |

3.3 Composition Function

Figure 2 shows a derivative of the bottom-up Tree-LSTM, illustrating the input (), memory cell () and hidden state () at the time. It extends the sequence LSTM by splitting the previous state vector into a left child state vector and a right child state vector , splitting the previous cell vector into and .

*ct*

*ht*

**Fig. 2.** Topology of tree LSTM. Shaded nodes represent the input vectors from our two trackers. White nodes represent hidden state vectors and memory cell vectors.

When the action is performed, the vector representations of two child nodes are popped off the stack and fed into a *composition function* which is also a neural network function that produces a representation for the new parent node at time step . The composition function of us calculating and as :

|  |  |  |
| --- | --- | --- |
|  |  | (2) |
|  |  | (3) |

where is an optional vector valued input argument which is either empty or comes from an external source like the trackers (see section 3.4), is the forget gate, is the input gate, is the output gate, and is the elementwise product. The new pair which represents a subtree is pushed onto the stack.

3.4 Trackers

State tracker [14] and connective tracker are two simple sequence-based LSTM. The state tracker’s inputs at the step is the top element of the buffer and the top two elements of the stack and . The connective tracker’s input at the step is the connectives in the next EDU to be popped in the buffer. So, the connective works only when the next transition is . The connectives we use are those artificially tagged connectives in PDTB [19]. We combine the outputs of two trackers into as shown in figure 3.We use these two tracker in two purpose: the composed hidden states supplies a representation of the to the transition classifier and it is also used as a input vector for the composition function (see 3.3).

**Fig. 3.** Network architecture of two trackers we use. The left one is the connective tracker. The right one is the state tracker. We use two colors to reflect the switch of the LSTM gate.

4 Methods for EDU Representation

We use word representation based on the 100D vectors provided by GloVe [15]. We do not update the weights of these vectors during training for we aim at comparing these encoders strictly. What’s more, we use the embeddings of part-of-speech (POS) tags as a supplement representation for each word in a sentence. In this section, we focus on finding out a most appropriate sentence encoder to represent each EDU. These popular encoders used in our experiments are listed below.

4.1 CNN based encoder

Convolutional Neural Network (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 size of filters as . The structure of CNN encoder with is designed in figure 2.

*You will win at last*

*embedding*

*convolution*

*max\_pooling*

POS

Words

**Fig. 4.** EDUs encoder based on CNN

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. It performs well on many tasks like reading comprehension [16]、natural language inference [17]、text summary [18] and so on. We introduce self-attention into this Bi-LSTM encoder with the purpose of obtaining decisive local information for discourse parsing. We formulate our vision of attention as:

|  |  |  |
| --- | --- | --- |
|  |  | (4) |

where as shown in figure 3, and are weights to learn.

**……**

**……**

**…**

**……**

Attention layer

Bi-LSTM layer

embedding

**Fig. 5.** EDUs encoder based on Bi-LSTM & Self-Attention

4.3 Encode EDUs with Features

Feature engineering is a traditional method in discourse parsing task. Many classical and effective discourse parsers are based on this. Nowadays, features are still useful in face of limited data. Wang et al. [8] extracted a lot of features from the top 2 elements in the stack and firstEDU in the queue to train a SVM classifier which achieves the highest score at the level of Span. Ji et al. [10] transformed these extracted features by multiplying them with a specific weight matrix which makes their model work as a data driven model. However, they use these representations to train a SVM classifier which is still a traditional method. Braud et al. [9] extracted almost the same features as DPLP and trained a feed-forward neural network as their parser model. They proposed a cross-lingual discourse parser which improves the performance with lots of external data.

**Table 2.** Features selected for RST parsing

|  |
| --- |
| 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 EDU |
| Length of tokens in the EDU (l1<=5, 5<l2<=12, 12<l3)  Position of EDU in a discourse (position of the EDU divided by the total number of EDU) |

These successful parsers have a similarity that they all use some simple but effective features to represent EDUs. To make the experiment more complete, we select some typical features to encode each EDU, as shown in table 2. The usual way of using these features in neural network is to automatically learn vectors for these features. In this paper, we combine the last 3 sets of features into 18 groups and assign different feature vectors to each group. We describe the representation of a EDU as:

|  |  |  |
| --- | --- | --- |
|  |  | (6) |

5 Experiments

5.1 Experimental Setup

The RST-DT 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 subtrees with right-branching [20] and use 18 coarse-grained relations as our relation set. We use the evaluation metrics defined by Marcu [21], i.e. the precision, recall and F-score with respect to span and nuclearity.

In our experiments we optimized on the development set (randomly choose 20 documents from training set) the following parameters: the learning rate = 0.003, dropout = 0.2, the size of hidden layers of Tree LSTM is 64, the size of hidden layers of Bi-LSTM is 128. Our models are trained with RMSProp to minimize a cross-entropy loss objective with an regularization term where the regularization hyper-parameter is 10-5. The final model was tested on the testing set after parameter tuning.

5.2 Results and Analysis

In this paper, we aim at finding a better way of EDU representation for neural network based discourse parsing. We hold the purpose that relation labeling should be done separately due to data sparsity and thus we put our attention only on span (S) and nuclearity (N) in this paper. As is shown in Table 3, this parser get a better result when using Bi-LSTM & Self-attention as EDUs encoder. The way of using CNN as EDUs encoder in our experiment is proved to be ineffective.

**Table 3.** Performance of our parser based on different encoders

|  |  |  |
| --- | --- | --- |
| Encoding Model | S | N |
| CNN | 78.32 | 55.63 |
| Bi-LSTM & Self-attention | **84.01** | **68.97** |
| Feature Vectors Learning | 83.56 | 67.92 |

We choose the Ji and Eisenstein’s discourse parser to compare with in three reasons: 1. Their parser has a state-of-the-art result in discourse parsing; 2. Their parser is also a data driven model different from others; 3. Their parser was built on the RST-DT only without using external data. The comparison results are shown in Table 4.

**Table 4.** Comparison results

|  |  |  |
| --- | --- | --- |
| Approach | S | N |
| Ji and Eisenstein | 82.1 | 71.1 |
| Ours | 84.01 | 68.97 |
| Human | 88.7 | 77.7 |

Compared with Ji, our parser is a pure dynamic processor which dose not need to prepare training data before for static oracle. We introduce a novel parser based on tree-structured neural network instead of SVM. What’s more, their parser learns the representation matrix and the SVM classifier at the same time which is harder to train than ours. From the F-score at the Span level, our parser are better than theirs.

6 Conclusion

We introduce connective tracker into constituent tree LSTMs to build a novel RST discourse parser. We conduct a series of experiments to compare the performance when using different EDU encoders. We have presented a framework to perform discourse parsing while jointly training the EDUs encoders to output better representations for EDUs. Our shift-reduce parsing system substantially outperforms Ji's data driven parser on span detection.

One big problem of our model is data sparsity which has limited the performance of the model. Our future work will focus on simplifying our model and introducing external data to alleviate this problem.

Reference

1. 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.
2. 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.
3. 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.
4. William C. Mann and Sandra A. Thompson. 1988. Rhetorical Structure Theory: Toward a functional theory of text organization. Text, 8:243–281.
5. Lynn Carlson and Daniel Marcu. 2001. Discourse tagging reference manual. Technical report, University of Southern California Information Sciences Institute.
6. 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.
7. Vanessa Wei Feng and Graeme Hirst. 2014. A linear-time bottom-up discourse parser with constraints and post-editing. In Proceedings of ACL.
8. Yizhong Wang, Sujian Li, and Houfeng Wang. 2017. A Two-Stage Parsing Method for Text-Level Discourse Analysis.
9. Chloé Braud, Maximin Coavoux, and Anders Søgaard. 2017. Cross-lingual RST Discourse Parsing.
10. Yangfeng Ji and Tacob Eisenstein. 2014. Representation learning for text-level discourse parsing. In Proceedings of ACL.
11. Daniel Marcu. 2000a. The rhetorical parsing of unrestricted texts: A surface-based approach. Computational Linguistics.
12. Jiwei Li, Rumeng Li, and Eduard Hovy. 2014 Recursive Deep Models for Dsicourse Parsing.
13. 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.
14. Samuel R. Bowman, Jon Gauthier, and Abhinav Rastogi. 2016. A Fast Unified Model for Parsing and Sentence Understanding.
15. Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global vectors for word representation. In Proc. EMNLP.
16. Jianpeng Cheng, Li Dong ,and Mirella Lapata. 2016. Long Short-Term Memory-Networks for Machine Reading. *In Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 551–561.
17. Ankur P. Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. Decomposable Attention Model for Natural Language Inference.
18. Romain Paulus, Caiming Xiong, and Richard Socher. 2017. A DEEP REINFORCED MODEL FOR ABSTRACTIVE SUMMARIZATION.
19. 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.
20. Radu Soricut and Daniel Marcu. 2003. Sentence level discourse parsing using syntactic and lexical information. In *Proceedings of NAACL*.
21. Daniel Marcu. 2000. The theory and practice of discourse parsing and summarization. MIT press.
22. Lynn Carlson, Daniel Marcu, and Mary Ellen Okurowski. 2001. Building a discourse-tagged corpus in the framework of Rhetorical Structure Theory. In *Proceedings of the Second SIGdial Workshop on Discourse and Dialogue.*