Event Temporal Relation Classification Based On Graph Convolutional Networks *

Qianwen Dai[1], Fang Kong[1] and Qianying Dai[1]

[1] Natural Language Processing Laboratory，School of Computer Science and Technology，Soochow University，Suzhou，Jiangsu 215006，China
kongfang@suda.edu.cn

Abstract. Classifying temporal relations between events is an important step of understanding natural language, and a significant subsequent study of event extraction. With the development of deep learning, various neural network frameworks have been applied to the task of event temporal relation classification. However, current studies only consider semantic information in local contexts of two events and ignore the syntactic structure information. To solve this problem, this paper proposes a neural architecture combining LSTM and GCN. This method can automatically extract features from word sequences and dependency syntax. A series of experiments on the Timebank-Dense corpus also show the superiority of the model presented in this paper.

Keywords: Temporal Relation; Graph Convolutional Networks; Syntactic Dependency .

1 Introduction

An event is a description of a certain behavior or state in a specific time and environment [1]. From the perspective of time, events may occur in a time interval or last for a period of time. Therefore, these events usually follow a sequential order. Event temporal relation classification is to identify the order of events according to the characteristics of time clues. Classifying temporal relations between events is a basic task of natural language processing. It has direct application in tasks such as question answering, event timeline generation and document summarization.

Temporal relations between events are various, such as "BEFORE", "AFTER", etc. Temporal relation classification aims to classify these relations correctly. Fig. 1 shows some examples of event temporal relations. S1 is an intra-sentence event pair (***boom***, ***layoffs***) with 'BEFORE' relation. S2 is a cross-sentence event pair (***break*** , ***visit***) with 'AFTER' relation.

Previous works studied this task as the classifification problem based on pattern matching and statistical machine learning. However, these methods need external

*This work is supported by Project 61876118 under the National Natural Science Foundation of China, and Key Project 61836007 under the National Natural Science Foundation of China.

resources to obtain semantic information, and depend on a large number of annotated entity attributes, which are difficult to be obtained in the practical scenarios. With the development of deep learning, temporal relation classification no longer depends on manual features and external resources. However, existing studies only consider semantic information in local contexts of two events and ignore syntactic information. How to make better use of syntactic structure information becomes the key.

S1: And at the big brokerage houses, after ten years of ***boom***, they're talking about ***layoffs***.

S2: a. The main negative is the risk that the pope's visit will persuade a great many more cubans to ***break*** loose of the cuban government.
b. If so, then the pope's ***visit*** would really open up a new chapter in the government's relations with its own society.

Fig.1. Examples of event temporal relation

In this paper, we proposes a neural architecture combining LSTMs and GCNs to better compute the representation for each word based on the syntactic dependency tree. Compared with existing studies, our model can achieve competitive results.

2 Related Work

Previous works of temporal relation classification are based on pattern matching and statistical machine learning. Mani et al. [2] built MaxEnt classififier on hand-tagged features for classifying temporal relations. Later Chambers et al. [3] used a two-stage classififier which first learned imperfect event attributes and then combined them with other linguistic features obtained from WordNet [4] and VerbOcean [5] in the second stage to perform the classifification. The following works mostly expanded the feature sets(Cheng et al.[6]; Bethard and Martin[7]; Kolomiyets et al.[8]).

In recent years, neural networks has been widely applied in the temporal relation classification. Xu et al. [9] perform LSTM with max pooling separately on each feature channel along dependency path. Cheng et al. [10] extracted the shortest dependency path from the dependency syntax tree, obtained good results through Bi-LSTMs. Choubey et al. [11] introduced more event context information based on the shortest dependency path and used Bi-LSTMs for Classifying Temporal Relations between Intra-Sentence Events.

In addition, Ning et al. [12] developed a probabilistic knowledge base acquired from the news domain, existing temporal extraction systems can be improved via this resource. Then Ning et al. [13] presented a joint inference framework using constrained conditional models (CCMs) for temporal and causal relations.

Compared with feature-based methods, neural networks based on dependency path achieves more advanced performance. But it causes lack of information and doesn't take the syntactic structure into consideration. In this paper, we present a sequential model with LSTMs and GCNs that can better compute the representation for each

word by combining semantic and syntactic information. Thus, the overall performance of event temporal relation classification is improved.

3 Graph Convolutional Networks

3.1 Graph Convolutional Networks

GCNs(Kipf and Welling(2017)[14]) are neural networks that operate directly on graph structures. It convolves the features of neighboring nodes and also propagates the information of a node to its nearest neighbors. Let $G=(V,E)$ be a undirected graph where V is the set of nodes ($|V|=n$), E indicates the edge set. We can define a matrix $X \in \mathbf{R}^{d\times n}$ denoting d-dimensional input node features. GCNs retrieve new node features at layer k+1 by encoding neighboring nodes' features with the following equation:

$$\boldsymbol{h}_v^{(k+1)} = RELU\left(\frac{\sum_{u\in N_{(v)}} \boldsymbol{W}\boldsymbol{h}_u^{(k)} + \boldsymbol{b}}{|N_{(v)}|}\right) \quad (1)$$

Here, $W \in \mathbf{R}^{d\times d}$ and $b \in \mathbf{R}^{d}$ are a weight matrix and a bias, respectively; $RELU$ is the rectifier linear unit activation function. v is the target node and N_v represents the neighborhood of v, including v itself. h_v^k is hidden representation of node u at layer k and $h_v^1 = x_v$.

3.2 Syntactic Graph Convolutional Networks

The above GCNs can only be used for the topological structure of undirected graph, while syntactic dependency trees are directed and there are various types of edges. This paper refers to the Syntactic GCNs proposed by Marcheggiani[15] and modifies the computation in order to incorporate label information.

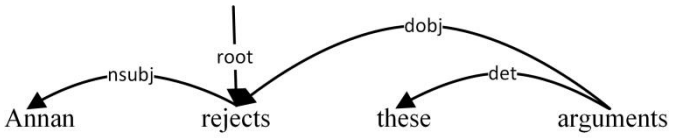

Fig.2 An example sentence annotated with syntactic dependencies

We add an arc in the opposite direction for each arc in the syntactic dependency tree. The syntactic dependency tree can be viewed as a graph whose edges are labeled. The label $L(u,v)$ for edge $(u,v)\in E$ contains two pieces of information. It describes the syntactic dependency type and indicates whether the edge is in the same or opposite direction as the syntactic dependency arc. For example, Fig. 2 shows the dependency graph of sentence "Annan rejects these arguments". After adding arcs in

the opposite direction, both (rejects, Annan) whose label is nsubj and (Annan, rejects) whose label is nsubj' belong to the edge set. We correspond label types to different weight matrices and bias terms to represent different combinations of directions and dependency types. In other words, the syntatic GCN parameters are label-specific. The computation can be written as shown below. The specific model is shown in Fig. 3 (the bias terms are omitted in the figure).

$$\boldsymbol{h}_v^{(k+1)} = \mathbf{ReLU}\left(\frac{\sum_{u\in N_{(v)}} \boldsymbol{W}_{L_{(u,v)}} \boldsymbol{h}_u^{(k)} + \boldsymbol{b}_{L_{(u,v)}}}{|N_{(v)}|}\right) \quad (2)$$

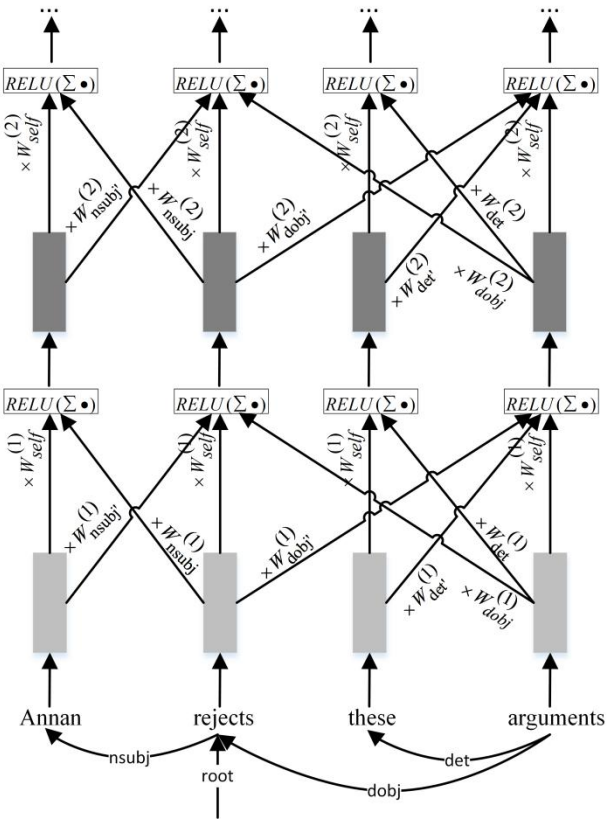

Fig.3 Syntactic GCN

Due to various dependency types and the arcs added, there are too many parameters in the model. To avoid over-fitting, we only keep the direction of each label and do not care about specific categories. So the label $L(u,v)$ can be reduced to three types: (1) the same direction as the syntactic dependency arc, (2) the opposite direction to the syntactic dependency arc, or (3) point to itself.

4 Event Temporal Relation Model Based on GCNs

The overall architecture of the proposed event temporal relation system is illustrated in Fig. 4. The system contains two identical network modules, corresponding to the processing process of two event sentence sequences. Each moudle is composed of four components: word embedding layer, Bi-LSTM layer, Syntactic GCN layer and max pooling layer. At the top of the model, there is the hidden layer, followed by the softmax layer for classification. The model is described in detail below.

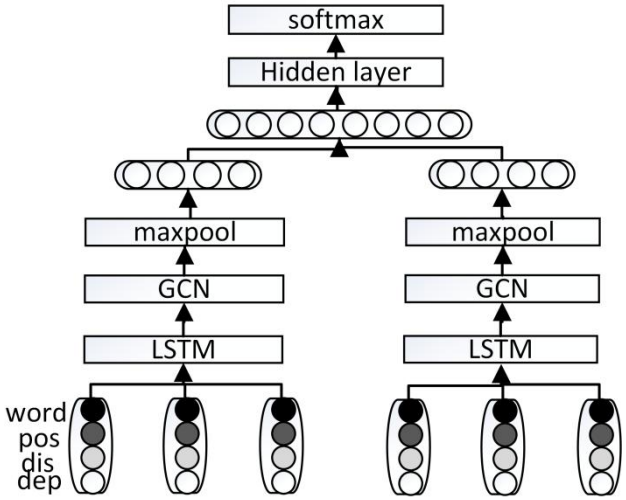

Fig.4 Event temporal relation classification model based on GCN

4.1 Word representations

Given the sentences of an event pair, for each word in the sentences, we create a word representation X_t . The word representation is the concatenation of four vectors:(i)a pre-trained word embedding x_{word} (ii) a randomly initialized part-of-speech tag embedding x_{pos} (iii) a randomly initialized embedding x_{dis} which represents the distance between the word and the event word (iiii) a randomly initialized dependency relation type embedding x_{dep} . It is noteworthy that x_{dep} cover the shortage of syntactic GCNs that ignoring dependency relation types to some extent. The final word representation as follows:

$$\boldsymbol{X}_t = \boldsymbol{x}_{word} \oplus \boldsymbol{x}_{pos} \oplus \boldsymbol{x}_{dis} \oplus x_{dep} \quad (3)$$

4.2 Bidirectional LSTM layer

Recurrent neural network is one of the most effective ways to represent sentences. In order to further improve the representation ability of our model, we use Bi-LSTM to transform the underlying input. At the same time, Bi-LSTM can make up the inability of GCNs to capture dependencies between nodes far away from each other in the

graph. For each word representation X_t, Bi-LSTM layer encodes it in two directions respectively. By concatenating outputs of two directions, we create a complete context-aware representation of a word:

$$\overrightarrow{\boldsymbol{h}_t} = \mathrm{LSTM}(\boldsymbol{X}_t, \overrightarrow{\boldsymbol{h}_{t-1}}), \quad (4)$$

$$\overleftarrow{\boldsymbol{h}_t} = \mathrm{LSTM}(\boldsymbol{X}_t, \overleftarrow{\boldsymbol{h}_{t+1}}), \quad (5)$$

$$\boldsymbol{h}_t = \overrightarrow{\boldsymbol{h}_t} \oplus \overleftarrow{\boldsymbol{h}_t} \quad (6)$$

4.3 Syntactic GCN layer

GCN is used to solve the problem of general neural networks that not easy to deal with topological structure. We use GCN to extract syntactic features from dependency trees. The representation calculated by Bi-LSTM layer is fed to a GCN defined in Equation (2). The output h_t^k of GCN is obtained. So the vector representation of event sentences are $H_1^k = \{h_{1_1}^k, h_{1_2}^k, ...h_{1_n}^k\}$ and $H_2^k = \{h_{2_1}^k, h_{2_2}^k, ...h_{2_n}^k\}$.

4.4 Max Pooling layer

Max pooling layer is added to select the maximum value of each column of the GCN outputs and the most informative data to form the final representation of event sentences.

$$\boldsymbol{l} = \max_{\mathrm{dim}=1}(\boldsymbol{H}^k) \quad (7)$$

4.5 Hidden layer

Finally, the representation of two event event sentences produced by max pooling layer are concatenated and fed into the hidden layer.

$$\boldsymbol{L} = \mathrm{concat}(\boldsymbol{l}_1 + \boldsymbol{l}_2), \quad (8)$$

$$\boldsymbol{Y} = \tanh(\boldsymbol{L}\boldsymbol{W}_h + \boldsymbol{b}_h) \quad (9)$$

Where l_1, l_2 are the representations of two event sentences, respectively. W_h is the weight matrix and b_h is bias of tanh function.

4.6 Temporal relation classifier

After the hidden layer, a softmax classifiifier predicts probabilites for each of the six classes:

$$o = \mathrm{softmax}(\boldsymbol{Y}\boldsymbol{W}_o + \boldsymbol{b}_o) \quad (10)$$

Where Y represents the output of the event sentences vector through the hidden layer, and W_o is the weight matrix and b_o is bias of softmax function.

In this paper, stochastic gradient descent algorithm is used to minimize the negative logarithmic likelihood function for model training. The objective function is defined as follows:

$$j(\theta) = -\sum_{i=1}^{n} \log p(y_i \mid x_i, \theta) \quad (11)$$

Where θ is the trainable parameter set of the model; n Represents the number of training samples; x_i Represents the i th sample of the training sample and y_i is the corresponding label.

5 Experiments

5.1 Datasets

We tested the proposed model on the TimeBank-Dense(TB-D) dataset to certify its validity and correctness.

TimeBank-Dense contains 36 documents annotated with 6 temporal relation types, including "AFTER", "BEFORE", "SIMULTANEOUS", "INCLUDES", "IS_INCLUDED", "VAGUE". "VAGUE" indicates event pairs whose temporal relations are unclear or missing. TB-D annotates 6088 event pairs and their specific distribution is shown in table 1.

Table 1. Event pairs distribution of TB-D corpus

Relation	Inter-sentence	Cross-sentence
AFTER	436	684
BEFORE	542	806
SIMULTANEOUS	49	44
INCLUDE	94	182
IS_INCLUDED	174	173
VAGUE	793	2111
Overall	2088	4000

5.2 Cross-validation and Hyper-parameters

We use a sentence-level 5-fold cross validation on the TB-D corpus and take the micro-average overall F1-score as the final result. We randomly sampled 15% of the training set acted as validation set. Early stopping is used to save the best model based on the validation data.The patience is set as 10. We use 200-dimensions pre-trained word embeddings from GloVe (Pennington et al., 2014)[16]. For POS and relative distance, we adopt the 50-dimensions look up table initialized randomly. For dependency relation, we adopt the 30-dimensions.

The batch size is 64. Adam optimization algorithm is adopted and the initial learning rate is 0.001. In order to prevent neural networks from over-fitting, we adopt dropout separately after embedding, Bi-LSTM, and hidden layer. The dropout ratio is 0.5. We set each single LSTM output with 128 dimensions, layer number is 3. GCN output is 256 dimensions, layer number is 2. The hidden layer is set as 200-dimensions.

5.3 Results and discussion

Comparison with previous studies. We compare the proposed system with two best performing feature-based systems and the best system in the neural network architecture.

- **CAEVO:** Chambers et al.[17](2014) proposed a hybrid system based on filter architecture. The system combines hand-crafted rules and hand-tagged features.
- **MIRZA**： Mirza and Tonelli[18](2016) mined the value of low dimensions word embeddings by concatenating them with sparse traditional features. Their traditional features includes entity attributes, temporal signals, semantic information of WordNet, etc..
- **Cheng**: Cheng et al.[10](2017) built a LSTM classifier based on dependency path . They extracted the shortest dependency path and take the word, part of speech and dependency relation as input features.

Table 2. Comparison with the existing methods

Relation	CAEVO	MIRZA	Cheng	Ours
AFTER	-	43.0	44.0	57.4
BEFORE	-	47.1	46.0	54.2
SIMULTANEOUS	-	-	-	-
INCLUDE	-	4.9	2.5	27.8
IS_INCLUDED	-	25.0	17.0	32.3
VAGUE	-	61.3	62.4	62.5
Overall	49.4	51.9	52.9	**56.8**

Table 2 shows the detail results. Compared with MIRZA, F1 increased by 4.5% , proving that neural networks has a significant effect on mining deep information of sentences. The comparison with Cheng also shows the superiority of our work.

Effect of GCNs. In order to confirm the impact of GCN, we compare our model against its version which lacks GCN layers and which lacks LSTM layers. Table 3 lists the detail comparisions. The last two columns represent our model, and k represents the layer number of GCN.

Experimental results show the efficiency of our model in temporal relation classification. GCN performs better than LSTM, no matter its layer number is 1 or 2. The classifier with LSTM and one GCN layers (K = 1) performs the best, which proves the complementarity of GCNs and LSTMs. Compared with the LSTM, F1 increased by 3.0%. The reason why the improvements is clear. Bidirectional LSTM only encodes semantic information while GCN takes the features of neighboring

nodes into consideration and models syntactic information. The experimental results prove the necessity of syntactic structure for temporal relation classification.

2 GCN layers performs better than 1 GCN layer when LSTM layers are dropped altogether. But when combined with the LSTM, F1 of 1 GCN layer increased by 2.8% while F1 of 2 GCN layers decreased by 1.4%. This suggests that extra GCN layers are effective but when there are too many layers it is largely redundant with respect to what LSTMs already capture.

In addition, to measure the influence of ignoring dependency relation types , we delete x_{dep} from the word vector X_t . Removing dependency types leads to a drop of 0.4% F1. It shows that the loss of syntactic dependency types has an effect on the results. We still can't throw out the type information completely.

Table 3. The effect of GCN on experimental performance

Relation	LSTM	GCN (k=1)	GCN (k=2)	LSTM +GCN - x_{dep} (k=1)	LSTM +GCN (k=1)	LSTM +GCN (k=2)
AFTER	53.6	54.3	58.6	59.1	57.4	55.0
BEFORE	48.4	48.4	51.0	53.7	54.2	49.1
SIMULTANEOUS	-	-	-	-	-	-
INCLUDE	12.9	16.3	16.4	29.6	27.8	18.6
IS_INCLUDED	28.5	28.7	31.3	26.9	32.3	29.1
VAGUE	60.9	61.2	62.1	61.6	62.5	61.7
Overall	53.8	54.0	56.0	56.4	**56.8**	54.6

6 Conclusion

In this paper, we propose an event temporal relation model based on graph convolutional networks (GCN). We combine LSTM and GCN to extract not only sequential features but also regional dependency features for each word. It overcomes the inability of traditional neural network model to process syntactic structure information of text. Compared with most previously proposed methods, our model is competitive. However, there is room for improvement. For example, the current researches mainly focus on event pairs, which may lead to inconsistent situations when constructing time chains later. In future studies, we can introduce linear programming to improve the overall performance of the model.

References

1. ZHENG X, LI P F, ZHU Q M, et al. Annotation and classification of temporal relation between chinese events[J]. Computer Science, 2015, 42(7): 276-279.

2. ZHENG X, LI P F, ZHU Q M, et al. Annotation and classification of temporal relation between chinese events[J]. Computer Science, 2015, 42(7): 276-279.
3. Mani I, Verhagen M, Wellner B, et al. Machine learning of temporal relations//Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics. Association for Computational Linguistics, 2006: 753-760..
4. Chambers N, Wang S, Jurafsky D. Classifying temporal relations between events// Meeting of the ACL on Interactive Poster and Demonstration Sessions. Association for Computational Linguistics, 2007:173-176.
5. Miller G A. WordNet: a lexical database for English. Communications of the ACM, 1995, 38(11): 39-41.
6. Chklovski T, PVerbocean P. mining the Web for fine-grained semantic verb relations//Conference on empirical methods in natural language processing. 2004.
7. Cheng Y, Asahara M, Matsumoto Y. NAIST. Japan: Temporal relation identification using dependency parsed tree//Proceedings of the 4th International Workshop on Semantic Evaluations. Association for Computational Linguistics, 2007: 245-248.
8. Bethard S, Martin J H. CU-TMP: Temporal relation classification using syntactic and semantic features//Proceedings of the 4th International Workshop on Semantic Evaluations. Association for Computational Linguistics, 2007: 129-132.
9. Kolomiyets O, Bethard S, Moens M F. Extracting narrative timelines as temporal dependency structures//Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1. Association for Computational Linguistics, 2012: 88-97.
10. Xu Y, Mou L, Li G, et al. Classifying relations via long short term memory networks along shortest dependency paths//proceedings of the 2015 conference on empirical methods in natural language processing. 2015: 1785-1794.
11. Cheng F, Miyao Y. Classifying temporal relations by bidirectional lstm over dependency paths//Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers). 2017: 1-6.
12. Choubey P K, Huang R. A sequential model for classifying temporal relations between intra-sentence events. arXiv preprint arXiv:1707.07343, 2017.
13. Ning Q, Wu H, Peng H, et al. Improving temporal relation extraction with a globally acquired statistical resource. arXiv preprint arXiv:1804.06020, 2018.
14. Ning Q, Feng Z, Wu H, et al. Joint reasoning for temporal and causal relations//Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). 2018: 2278-2288.
15. Kipf T N, Welling M. Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907, 2016.
16. Marcheggiani D, Titov I. Encoding sentences with graph convolutional networks for semantic role labeling[J]. arXiv preprint arXiv:1703.04826, 2017.
17. Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In Proc. of EMNLP.
18. Chambers N, Cassidy T, McDowell B, et al. Dense event ordering with a multi-pass architecture. Transactions of the Association for Computational Linguistics, 2014, 2: 273-284.
19. Paramita M, Tonelli S. On the contribution of word embeddings to temporal relation classification//26th International Conference on Computational Linguistics (Coling 2016): Technical Papers. 2016: 2818-2828.