Bidirectional Long Short-Term Memory Networks for Relation Classification

Shu Zhang[1], Dequan Zheng[2], Xinchen Hu[2] and Ming Yang[1]
[1] Fujitsu Research and Development Center, Beijing, China
{zhangshu, yangming}@cn.fujitsu.com
[2] School of Computer Science and Technology, Harbin Institute of Technology, Harbin, China
{dqzheng, xchu}@mtlab.hit.edu.cn

Abstract

Relation classification is an important semantic processing, which has achieved great attention in recent years. The main challenge is the fact that important information can appear at any position in the sentence. Therefore, we propose bidirectional long short-term memory networks (BLSTM) to model the sentence with complete, sequential information about all words. At the same time, we also use features derived from the lexical resources such as WordNet or NLP systems such as dependency parser and named entity recognizers (NER). The experimental results on SemEval-2010 show that BLSTM-based method only with word embeddings as input features is sufficient to achieve state-of-the-art performance, and importing more features could further improve the performance.

1 Introduction

The automatic classification of semantic relations is an important task, which could offer useful information for many applications, such as question answering, information extraction, the construction and completion of semantic or relational knowledge base.

In this work, we focus on the classification of semantic relations between pairs of nominals (Hendrickx et al., 2010). Given a sentence S with annotated pairs of nominal e_1 and e_2, the task is to classify which of the following nine semantic relations holds between the nominals: *Cause-Effect*, *Instrument-Agency*, *Product-Producer*, *Content-Container*, *Entity-Origin*, *Entity-Destination*, *Component-Whole*, *Member-Collection*, *Message-Topic*, or *Other* if it does not belongs to any of the nine annotated relations.

For example, *News* and *commotion* are connected in a *Cause-Effect* relation in the sentence "The *news* brought about a *commotion* in the office." In this instance, the relation between *news* and *commotion* could be inferred by the meaning of the two nominals and the context of "brought about" around them. Therefore, how to grasp and represent the lexical and context information are the key research points for semantic relation classification.

Supervised methods with carefully handcrafted features from lexical and semantic resources have achieved high performance (Hendrickx *et al.*, 2010; Rink and Harabagiu, 2010). However, the selection of features and the effective integration of knowledge sources into relation classification seem to be difficult.

Recently, deep neural networks has been applied with the aim of reducing the number of handcrafted features, and getting effective features from lexical and sentence level (Socher et al., 2012; Zeng et al., 2014; Yu et al., 2014).

Different from previous work, we propose bidirectional long short-term memory networks (BLSTM) to solve the relation classification. For every word in a given sentence, BLSTM has complete, sequential information about all words before and after it. Long distance relationship may be solved in some extent in this networks. At the same time, we also use features derived from the lexical resources such as WordNet or NLP tools such as dependency parser and named entity recognizers (NER). The experimental results show that only using word embedding as input features is enough to achieve state-of-the-art results. Importing more features could further improve the performance of the relation classification.

29th Pacific Asia Conference on Language, Information and Computation pages 73 - 78
Shanghai, China, October 30 - November 1, 2015

2 Related Work

SemEval-2010 task 8 focused on semantic relation classification, it provides a standard testbed to evaluate and compare the performance of different approaches.

SVM (Rink and Harabagiu, 2010): Using SVM classifier and a number of features derived from NLP tools and many external resources, it achieves the highest performance among the participating systems (10 teams, 28 runs).

Neural network has got great achievement in many applications, it has also been utilized in relation classification as shown in the followings:

MV-RNN (Socher et al., 2012): They propose a recursive neural network model to learn compositional vector representations for phrases and sentences of arbitrary syntactic type and length.

CNN (Zeng et al. (2014): Sentence level features are learned using a convolutional model, and concatenated with lexical features to form the final extracted feature vector.

FCM (Yu et al., 2014): They decompose the sentence into substructures, and extract features for each substructure. Finally they combine these features with the embeddings of words in this substructure to form a substructure embedding.

CR-CNN (Santos et al., 2015): They propose network to learn a distributed vector representation for each relation class. A ranking loss function is proposed to reduce the impact of artificial classes.

DepNN (Liu et al., 2015): Using a recursive neural network to model the subtrees, and a convolutional neural network to capture the most important features on the shortest path.

From the above works, we can see that many different neural network models have been applied to solve relation classification recently. The main target is to learn the effective features in lexical and sentence level to represent the latent relation between the given nominals.

Our work has the same target, and we try to apply BLSTM to mine the sentence level features with its advantage of capturing long distance relationship in a sentence. We also study the influence of adding features obtained from NLP tools and resources on the final classification performance.

3 Long Short Term Memory

The Long Short Term Memory architecture was proposed and extended (Hochreiter and Schmidhuber, 1997; Gers et al., 2002) with the motivation on an analysis of Recurrent Neural Nets (Hochreiter et al., 2001), which found that long time lags were inaccessible to existing architectures, because backpropagated error either blows up or decays exponentially.

A LSTM layer consists of a set of recurrently connected blocks, known as memory blocks. Each one contains one or more recurrently connected memory cells and three multiplicative units - the input, output and forget gates - that provide continuous analogues of write, read and reset operations for the cells. LSTM has achieved the best known results in handwriting recognition (Graves et al., 2009) and speech recognition (Graves et al., 2013).

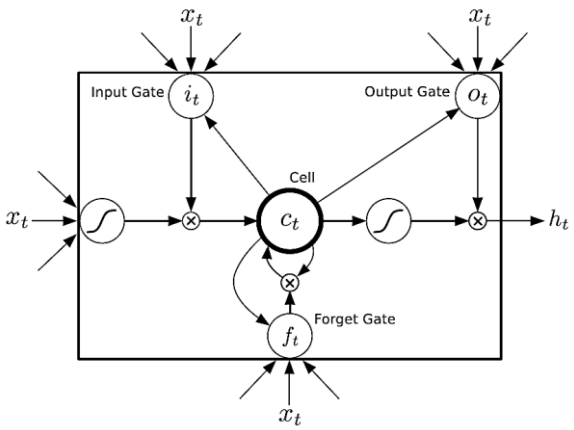

Fig. 1. LSTM memory block with one cell

Figure 1 shows one cell of LSTM memory block. More precisely, the input x_t to the cells is multiplied by the activation of the input gate, the output to the net is multiplied by that of the output gate, and the previous cell values are multiplied by the forget gate. The net can only interact with the cells via the gates.

The basic idea of bidirectional LSTM is to present each training sequence forwards and backwards to two separate recurrent nets, both of which are connected to the same output layer. This means that for every point in a given sequence, the network has complete, sequential information about all points before and after it. The structure of BLSTM is shown in Figure 2.

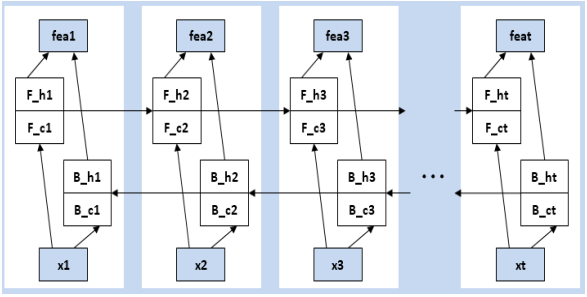

Fig. 2. Bidirectional LSTM

4 Methodology

We propose bidirectional long short-term memory networks (BLSTM) to solve the relation classification. It includes the following parts:

(1) Initial feature extraction: extract from the input sentence.
(2) Features embedding: transform all initial features into real-valued vector representation.
(3) BLSTM-based sentence level representation: get high level feature representation from step (2).
(4) Constructing feature vector: get lexical level and sentence level features from step (2) and step (3), and concatenate them to form the final feature vector.
(5) Classifying: feed final feature vector into a multilayer perceptron (MLP) and softmax layer to get the probability distribution of relation labels.

4.1 Initial Feature Extraction

Besides word and position features, we utilize NLP tools and resources to get POS, NER, dependency parse and hypernyms features. We aim to grasp more features which may indicate the relationship of the pair of two nominals. All these features could be classified into two types: lexical features and relative position relationship features.

We extract word, POS, NER and hypernyms as lexical features. The WordNet hypernyms are adopted as MVRNN (Socher et al., 2012).

Three different relative position relationship features are extracted and shown in Figure 3.

In this work we also utilize the relative word position proposed by Zeng et al. (2014). The position feature (**PF**) is derived from the relative distances of the current word to the target nominals e_1 and e_2. For instance, the word *sat* in the sentence shown in Figure 3, its relative distance to the target nominal *cat* (e_1) and *mat* (e_2) are 1 and -3.

We also chose the Stanford dependency parser to capture long distance relationships between two nominals in a sentence. Our dependency features are based on paths in the dependency tree. Here, we extract two types of features:

Relative dependency features:

- Relative root feature: *r_r* (root node), *r_c* (child node of root), *r_o* (others)
- Relative e_1 feature: e_1_e_1 (e_1 node), e_1_*c* (child node of e_1), e_1_*p* (parent node of e_1), e_1_*o* (others)
- Relative e_2 feature: e_2_e_2 (e_2 node), e_2_*c* (child node of e_2), e_2_*p* (parent node of e_2), e_2_*o* (others)

Dep features: the tag of the current word to its parent node on the dependency tree

The above features represent the relationship between the current word and the target node, including the root, e_1, e_2 and their parent node. Figure 4 gives an example of dependency parser results.

	The	cat(e1)	sat	on	the	mat(e2
Relative root	r_o	r_c	r_r	r_o	r_o	r_c
Relative e1 feature	e1_c	e1_e	e1_p	e1_o	e1_o	e1_o
Relative e2 feature	e2_o	e2_o	e2_p	e2_c	e2_c	e2_e
dep	det	nsubj	root	case	det	nmo
PF	-1 -5	0 -4	1 -3	2 -2	3 -1	4 0

Fig. 3. Example of relative position relationship features

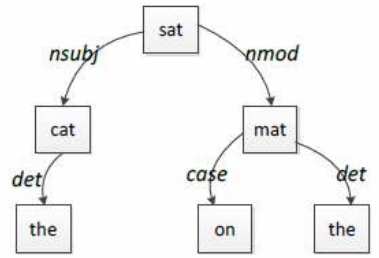

Fig. 4. Example of dependency parser results

4.2 Feature Embedding

Word Embedding is to map each word into a real-valued vector to represent syntactic and semantic information about the words.

Given an embedding matrix $W^{wrd} \in \mathbf{R}^{d^w \times |V|}$, where V is the size of word vocabulary. Each word w has its embedding by using the matrix-vector product:

$$r^w = W^{wrd} v^w$$

where v^w is one-hot represenation, to get one column of the matrix W^{wrd}.

The size of the word embedding d^w is a hyperparameter, which is usually set 50 or 100.

For other kinds of initial features, we also transform them into a vector representation r^{kj}, where j means the *jth* type of feature, the dimension is d^{kj}. The initial value of the vector is random generated with the method proposed by Glorot and Bengio (2010).

Given a sentence $x=\{w_1, w_2, \ldots, w_n\}$, all the initial feature embeddings are concatenated according to the following format to represent each word:

$$x_i = \left[r_i^w, r_i^{k1}, r_i^{k2} \ldots, r_i^{km}\right]$$

where r_i^w is the word embedding of word x_i, r_i^{kj} is embedding of the *jth* types of features.

The parameter m is the size of features. Its value is 6 in this paper, because we choose the following six kinds of features: POS, NER, hypernyms(WNSYN), position feature (PF), dependency feature (Dep), relative-dependency feature (Relative-Dep).

4.3 BLSTM-based Sentence Level Representation

It is well known that humans can exploit longer context to mine the relationship of two nominals in a sentence. LSTM has shown its merit on capturing long distance relationship in different fields. With this motivation, we adopt BLSTM to get the sentence level representation.

The LSTM equations are given for a single memory block.

Input Gates:

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i)$$

Forget Gates:

$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f)$$

Cells:

$$c_t = f_t c_{t-1} + i_t \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c)$$

Output gates:

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o)$$

Cell Outputs:

$$h_t = o_t \tanh c_t$$

where σ is the activation function, and i, f, o and c are respectively the input gate, forget gate, output gate and memory cell.

As shown in Figure 2, the network contained two sub-networks for the left and right sequence context. The outputs of these subnets for the *ith* word are integrated in the following way:

$$F_i = [F\_h_i, F\_c_i, B\_h_i, B\_h_i]$$

where F and B refer to forward and backward directions.

4.4 Constructing Feature Vector

Inspired by the work from Zeng et al. (2014), we extract and concatenate sentence level features and lexical level features to form the finally extracted feature vector.

Lexical level features are focused on the two target nominals $e_1$and e_2. We concatenate the vector got from feature embeddings and BLSTM layer to represent the two nominals as [x_{e1}, F_{e1}, x_{e2}, F_{e2}].

Sentence level features are focused on the context information, which are constructed from the output of BLSTM layer. As shown in Figure 5, the matrix got from BLSTM could be divided into A, B and C parts by e_1 and e_2. Max pooling operation is adopted to extract the vector from A and B parts, B and C parts respectively. The vector m_1 and m_2 is concatenated to form the sentence level representation.

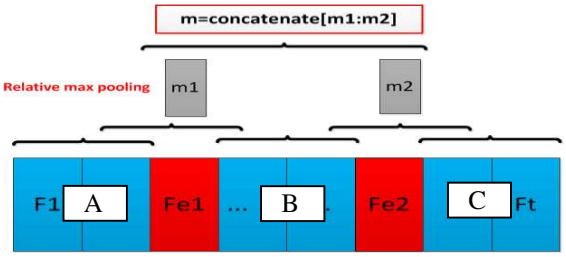

Fig. 5. Constructing sentence level feature vector

The motivation of constructing sentence level in this way is to strengthen the influence of the context between two entities, which are usually contained more information for indicating the relationship.

4.5 Classifying

A multilayer perceptron (MLP) will be used for combining sentence level feature and lexical feature into the final extracted feature vector. Finally, the final extracted features are fed into a softmax classifier to predict the sematic relation labels.

5 Experiments

5.1 Data and metrics

Experiments are conducted on the SemEval-2010 task 8 dataset (Hendrickx *et al.*, 2010). It includes 8,000 training instances and 2,717 test instances. There are 9 relation types, and each type has two directions. If the instance could not refer to any of 9 relation types, there is a type *Other*.

We adopt the official evaluation metric to evaluate our systems, which is based on macro-averaged F1-score for the nine proper relations and others.

5.2 Experiments setting

The dimension of feature embeddings used in the experiments are listed in the following.

Features	Embedding Dimension
WF	50, 100
PF	2*5
POS	20
NER	20
WNSYN	20
DEP	20
RELATIVE-DEP	3*10

Table 1. Embedding dimension

We select two available trained word embeddings to see its influence to the classification performance. One is from Turian et al. (2010), the dimension of word embedding is 50. The other is from Jeffrey Pennington et al. (2014), the dimension of word embedding is 100.

As shown in the above, position feature (PF) contains two elements, and relative-dependency feature (Relative-Dep) contains three elements. Therefore, embedding dimension of PF is 2*5, that of RELATIVE-DEP is 3*10.

The BLSTM layer contains 400 units for each direction, and MLP layer contains 1000 units.

5.3 Results and Analysis

Firstly, we testify the performance of proposed BLSTM-based method with two feature set. One only uses word embedding as input, the other uses all features shown in section 4.1. We also list the results of CNN and CR-CNN methods as reference.

Model	Feature Set	F1
CNN (Zeng et al., 2014)	Only word embeddings	69.7
	word embeddings, word position embeddings,word pair, words around word pair, WordNet	82.7
CR-CNN (Santos et al., 2015)	Only word embeddings	82.8
	word embeddings, word position embeddings	84.1
BLSTM	Only word embedding (100)	82.7
	All features	84.3

Table 2. Comparison with previously published results

In table 2, only using word embedding as input features, BLSTM-based method achieves F1 of 82.7, which is similar to the results of CNN with multiple features, and CR-CNN with only word embedding features. However, CR_CNN use word embeddings of size 400, our method use word embeddings of size 100. It proves that BLSTM-based method is effective to mine the relationship between two nominals. With more features, the performance achieves F1 of 84.3, which testifies general features gotten from NLP tools could improve the classification performance.

Secondly, we testify the influence of different features for the classification by removing one type of features from feature set in each time.

From Table 3, we see that the performance has very slight change by removing position and NER features. It shows that BLSTM has better representation on sentence level relationship without position features. The information of position features is already contained in BLSTM networks. The whole features are considered from lexical and sentence level. The performances of removing PF or NER feature don't change obviously, maybe the information they contained is represented by other features.

Removed Feature	F1
PF	84.2
POS	83.9
NER	84.2
WNSYN	83.2
DEP	83.5

Table 3. Results of removing one kind of feature

Finally, we compare the results in different word embedding size. In Table 4 we give the result with using word embedding of size 50. It achieves a F1 of 83.6, about 0.7% less than that with using word embedding of size 100, which shows larger size of dimension of word embedding may contain more information, and it could improve the performance.

We also compare the LSTM based method with only one direction such as forward or backward. The results shows BLSTM has a slight advantage over unidirectional LSTM.

Compared with proposed constructing sentence level feature vector in figure 5, we use Max pooling operation directly from A+B+C parts. The result shows F1 of 83.1, which is lower than our method with F1 of 83.6. It proves that our proposed method is effective.

Model (word embedding 50)	F1
BLSTM	83.6
Forward-LSTM	82.1
Backward-LSTM	82.4
Single-max model	83.1

Table 4. Results of removing one kind of feature

6 Conclusion

In this paper, we propose bidirectional long short-term memory networks (BLSTM) to solve the relation classification. BLSTM is proposed to mine the sentence level representation. The experiment results show that only using word embedding as input features is enough to achieve state-of-the-art results. Importing more features could further improve the performance of the relation classification.

Reference

Bryan Rink and Sanda Harabagiu. 2010. UTD: Classifying Semantic Relations by Combining Lexical and Semantic Resources. In Proceedings of 5th International Workshop on Semantic Evaluation, pages 256–259.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. 2014. Relation Classification via Convolutional Deep Neural Network. In Proceedings of the 25th International Conference on Computational Linguistics (COLING), pages 2335–2344.

Iris Hendrickx, Su Nam Kim, Zornitsa Kozareva, Preslav Nakov, Diarmuid Ó Séaghdha, Sebastian Padó, Marco Pennacchiotti, Lorenza Romano, and Stan Szpakowicz. 2010. Semeval-2010 Task 8: Multi-way Classification of Semantic Relations Between Pairs of Nominals. In Proceedings of the 5th International Workshop on Semantic Evaluation, pages 33–38.

Richard Socher, Brody Huval, Christopher D. Manning, and Andrew Y. Ng. 2012. Semantic Compositionality through Recursive Matrix-Vector Spaces. In Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pages 1201–1211.

Yang Liu, Furu Wei, Sujian Li, Heng Ji, Ming Zhou, and Houfeng Wang. 2015. A Dependency-based Neural Network for Relation Classification. In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Short Papers), pages 285-290.

Cícero Nogueira dos Santos, Bing Xiang, and Bowen Zhou. 2015. Classifying Relations by Ranking with Convolutional Neural Networks. In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing, pages 626–634.

Alex Graves, Marcus Liwicki, Santiago Fernández, Roman Bertolami, Horst Bunke, Jürgen Schmidhuber. 2009. A Novel Connectionist System for Improved Unconstrained Handwriting Recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence, 31(5): 855-868.

Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech Recognition with Deep Recurrent Neural Networks. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 6645–6649.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word Representations: A Simple and General Method for Semi-Supervised Learning. In Proceedings of the 48th annual meeting of the association for computational linguistics, pages 384-394.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global Vectors for Word Representation. In Proceedings of the Empirical Methods in Natural Language Processing, pages 1532-1543.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the Difficulty of Training Deep Feedforward Neural Networks. International conference on artificial intelligence and statistics, pages 249-256.

Mo Yu, Matthew R. Gormley, and Mark Dredze. 2014. Factor-based Compositional Embedding Models. In NIPS Workshop on Learning Semantics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. Neural Computation, 9(8):1735–1780.

Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, and Jürgen Schmidhuber. 2001. Gradient Flow in Recurrent Nets: the Difficulty of Learning Long-Term Dependencies. In Kremer, S. C.and Kolen, J. F., editors, A Field Guide to Dynamical Recurrent Neural Networks. IEEE Press.

Felix A. Gers, Nicol N. Schraudolph, and Jürgen Schmidhuber. 2002. Learning Precise Timing with LSTM Recurrent Networks. Journal of Machine Learning Research, 3:115–143.