Improving Relation Classification by Incorporating Dependency and Semantic Information

1st Kun Deng
School of Computer Engineering and Science
Shanghai University
Shanghai, China
dengkun@shu.edu.cn

2nd Shaochun Wu*
School of Computer Engineering and Science
Shanghai University
Shanghai, China
scwu@shu.edu.cn

Abstract—Relation classification is the task of identifying relations between two entities in a sentence, which is an essential step in the standard NLP pipeline. Most of the previous models only make use of dependency or semantic features, which may result in the loss of vital information. In this paper, we propose a novel model that incorporates dependency and semantic information for relation classification. This is a neural network model using long short-term memory(LSTM), graph convolutional networks(GCN), and convolutional neural networks(CNN), named LGCNN. Concretely, it utilizes self-attention and LSTM to capture the local context in sentences. What's more, it uses graph convolution network to encode dependency information and takes advantage of convolution neural network to encode semantic information from the local context. Experiments on the SemEval-2010 Task 8 and KBP37 dataset demonstrate that our model is very effective in relation classification.

Index Terms—Relation classification, dependency features, semantic features

I. Introduction

Relation Extraction(RE) is a significant task in Natural Language Processing (NLP) [1]. The task is to extract semantic relations between entity pairs in a sentence based on their context. Use the following sentence to give an example: "Forward <e1> motion </e1> of the vehicle through the air caused a <e2> suction </e2> on the road draft tube." The marked entity pair "motion" and "suction" are of the relation "Cause-Effect(e1,e2)". Many tasks such as dialog generation, information retrieval and question answering require relation extraction as an intermediate step.

Traditional relation extraction methods are based on either hand-crafted features [1] or elaborately designed kernels [2], which are time-consuming and challenging to apply to novel domains. Recently, neural network-based methods are proposed to solve the shortcoming of traditional methods, which can automatically extract semantic features from text and reduces manual intervention. Neural network-based methods include convolutional neural networks [3], long short-term memory based recurrent neural networks [4], [8], graph convolutional networks [5], and so on.

*Corresponding author.

While CNNs are able to learn local semantic features, GCNs have been effective in learning dependency features. In this paper, we propose a novel model that incorporates CNN and GCN to extract the semantic and dependency features of sentences for relation classification. The contributions of our work can be summarized as follows: (1) We propose an attention-based GCN method to learn dependency sentence features. Compared with previous methods, our method utilizes attention and local context for learning word representations. (2) We also present CNN, a framework with multiple types of filters in capturing the semantic features on the context vector. (3) We combine the GCN and CNN to preserve the full relation information. Our proposed model proved to be very useful on the SemEval-2010 Task 8 and the KBP37 dataset.

II. Related Work

A. Relation Extraction

Recently, neural network models have shown superior performance in many areas compared to traditional models with hand-crafted features. Zeng [3] first applied the CNN model to relation extraction, which improves the effect of relation extraction by automatically capture relevant lexical and sentence-level features. Variants of CNN-based methods include convolutional deep neural network [3], ranking with CNN(CRCNN) [6], and Attention-CNN [7]. The RNN-based method is another popular choice in relation extraction. The most recently used RNN models are bidirectional long short-term memory networks(Bi-LSTM) [8], Attention-LSTM [9], and LSTM with entity-aware attention [10].

Dependency tree-based features were applied to relation extraction and found to be very efficient in extracting dependency information [11], [12]. Yang [24] utilizing dependency information of dependency parse tree and present a position encoding convolutional neural network for relation classification. Le [13] obtained the shortest dependency path(SDP) between two entities based on dependency parse, and extracted the relation by incorporating CNN and two-channel RNN with LSTM units. Some recent work combines SDP with graph convolutional

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

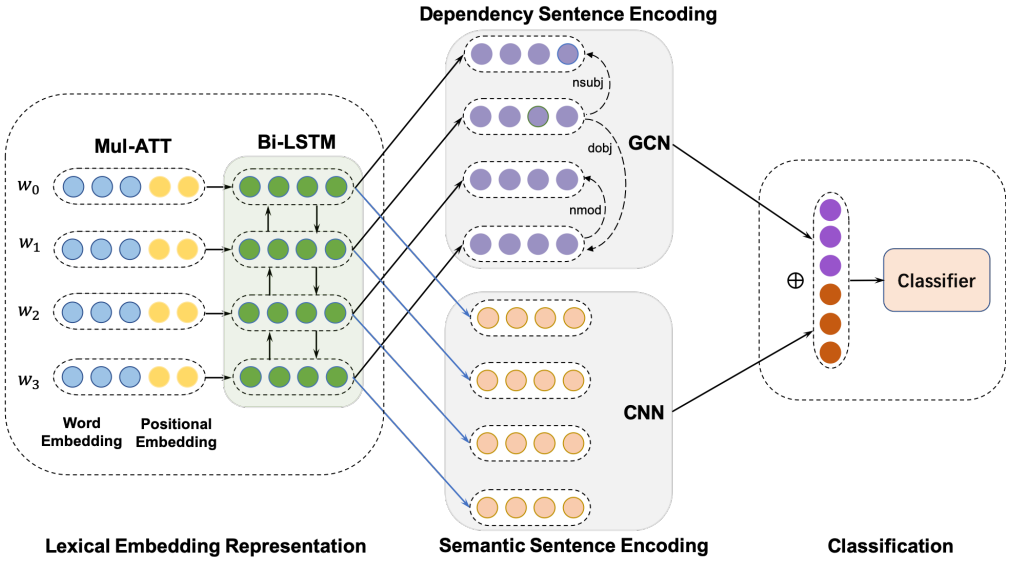

Fig. 1. Overview of LGCNN. LGCNN can be divided into four parts: Lexical Embedding Representation, Dependency Sentence Encoding, Semantic Sententce Encoding and Classification.

network(GCN). The paper [5] suggested incorporating pruning strategies and SDP to improve the performance of dependency tree-based methods, which used GCN to model the pruned-tree.

B. Graph Convolution Networks

The graph convolutional network [14] encodes the nodes in the graph according to its graph structure and an application of the convolutional neural network. Since graph convolution network can effectively encode sentence semantic information, it is widely applied in Neural Language Process such as text generation [15], question answering [16], and machine translation [17]. In some recent studies, graph convolution network was applied to relation extraction [5], [18]. However, they either used the pruned dependency path to build the graph or relied on the shortest dependent path to construct a graph, which may lose some significant information. We use the attention of the shortest dependency path to assign different weights to the nodes in the graph, and use the graph convolution network to extract the dependency information.

III. Methods Overview

Given a word sequence $S = \{w_1, w_2, ...\}$ contains two entities e_1 and e_2, the aim of relation extraction is to extract the semantic relation between two entities. Since the relation set $C = \{c_1, c_2, ...\}$ is given, the problem can be converted into a classification task. LGCNN consists of four components, and we will briefly introduce each part in this section. The overall framework of LGCNN is shown in Fig. 1.

1) Lexical Embedding Representation: LGCNN uses multi-head attention over the concatenated word and position embedding for capturing the meaning of the correlation between words. For capturing the local context of each token, Bi-LSTM is employed to encode the representations of the output of the multi-head attention. In this component, to get a better lexical representation, we use multi-head attention and Bi-LSTM. More details in section IV-A.

2) Dependency Sentence Encoding: In this part, GCN is applied to encode the representation of each token on the dependency tree to capture long-range dependencies. To capture the most critical information about the relation classification in the dependency tree, we use word attention based on the shortest dependency path. In section IV-B, we will describe the component in detail.

3) Semantic Sentence Encoding: In this part, we use CNN over lexical representation to get the uppermost semantic information of the sentence. Please refer to section IV-C for more details.

4) Classification: Finally, sentence representation is acquired by stitching the output of dependency sentence embedding and semantic sentence embedding. Sentence representation is used to match the most likely relation through a fully connected layer in the candidate relations. We will elaborate on this component in section IV-D.

IV. Model Details

In this section, we introduce each components of LGCNN in detail.

A. Lexical Embedding Representation

Given a sentence $S = \{w_1, w_2, ...\}$ containing an entity pair, we encode each token in the sentence into better lex-

ical representation, which consists of an embedding layer, a self-attention layer, and a Bi-LSTM layer. The input sentence contains position indicators(PI): $e_{11}, e_{12}, e_{21}, e_{22}$, which will be treated as a single word. The position indicator has been proven to be very useful in relation extraction [19].

1) Embedding Layer: The embedding layer aims at mapping words in sentence into continuous input embedding. Firstly, we map each token w_i to a k-dimensional GloVe [20] embedding $w_i \in \mathbb{R}^{d_w}$. Then, we map the relative distance of each token to two entities into two p-dimensional position embeddings, which is following Zeng [3]. Finally, we get the final input embedding for each token in the sentence by stitching word embedding and two position embeddings.

2) Self-attention Layer: To solve the long-term dependency problems, we utilize the multi-head attention mechanism to capture the meaning of the correlation between words. Multi-head attention [21] is able to learn the implicit contextual information for words, which applies the self-attention mechanism multiple times over the same input. The following formula can express the process:

$$a_{ijh} = softmax(q_{jh}k_{ih}^T/\sqrt{d}) \tag{1}$$

$$o = \sum_j \alpha_{ijh}\nu_{jh} \tag{2}$$

$$o_i = [o_{i1}; ...; o_{ih}] \tag{3}$$

$$O = w^m o + X^{k-1} \tag{4}$$

In the attention mechanism, q, k, and v are equal, which are the input vectors after being segmented, where a_{ijh} represents the weight of the j^{th} token in the h^{th} attention mechanism for the i^{th} token. And d is denoting the vector length of q, o_{ih} indicating the weighted representation of the i^{th} word in the h^{th} head attention mechanism. With $[...; ...]$ denoting the concatenation operation, and o_i is the representation of the i^{th} word after concatenation through the multi-head attention mechanism. We use the residual connection method to get the final representation O, which is the sum of concatenation outputs of self-attention and the input X^{k-1} of the previous layer, where $w^m \in \mathbb{R}^{d\times d}$ is a learnable variable of a linear transformation.

3) Bi-LSTM layer: In this layer, we use Bi-LSTM network to encode the context representation of each word. Bi-LSTM network consists of two parts: a forward LSTM network which encodes the context of sentence in sequential order and a backward LSTM network which encodes the context of sentence in reverse order. The strategy of the model is to connect the forward LSTM hidden state $\overrightarrow{h_t}$ and the backward LSTM hidden state $\overleftarrow{h_t}$, so that each token contains a forward context and backward context. Bi-LSTM hidden state can be represented by the following:

$$h_t = \overrightarrow{h_t} \oplus \overleftarrow{h_t} \tag{5}$$

where $\oplus$ denotes concatenate operation.

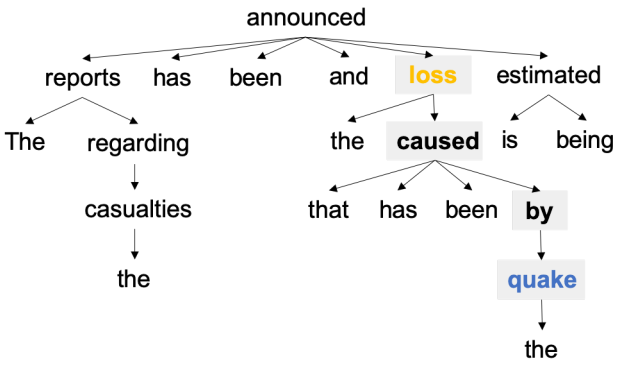

Fig. 2. is the dependency tree of the sentence: "The reports regarding the casualties has been announced and the loss that has been caused by the quake is being estimated".

B. Dependency Sentence Encoding

As we all know, the dependency path contains a large amount of dependency information, which will facilitate the relation extraction. We use a graph convolution network(GCN) to extract the dependency information of the token in the dependency tree. The shortest dependency path(SDP) of the two entities can effectively extract the most important information between two entities, but only using SDP may lose some significant information. Therefore, we use the attention mechanism based on SDP. First, we get a weighted representation of the word by SDP-based attention and then encode it using GCN.

Since the different words are of varying importance in the dependency tree, and SDP contains important information between the two entities, so we use SDP-based attention for the words. The SDP-based attention is to assign a higher weight to words on SDP, and words that are not on SDP give lower weight. The weight of each word in a sentence can be derived from the following formula:

$$a_i = \begin{cases} a_{high}, & \text{if } i \in S_{sdp} \\ a_{low}, & \text{if } i \notin S_{sdp} \end{cases} \tag{6}$$

where a_i refers to the weight of the i^{th} word, S_{sdp} represents the set of words on the SDP path, a_{high} and a_{low} are two weight representations, indicating the high weight and low weight respectively.

For example, Fig. 2 is a dependency tree of sentences, where "loss" and "quake" are two entities, and the shortest dependency path is "loss-caused-by-quake". Through the SDP-based attention mechanism, we assign the weights of "loss", "caused", "by", and "quake" to a_{high}, and the weights of other words to a_{low}.

Given a sentence $S = \{w_1, w_2, ...\}$, we use the Spacy tool to generate the dependency tree and extract SDP between the two entities. The dependency tree can be represented as a directed graph G, where the nodes N of the graph represent the words in the sentence, and the edges Σ in the graph represent the dependencies between

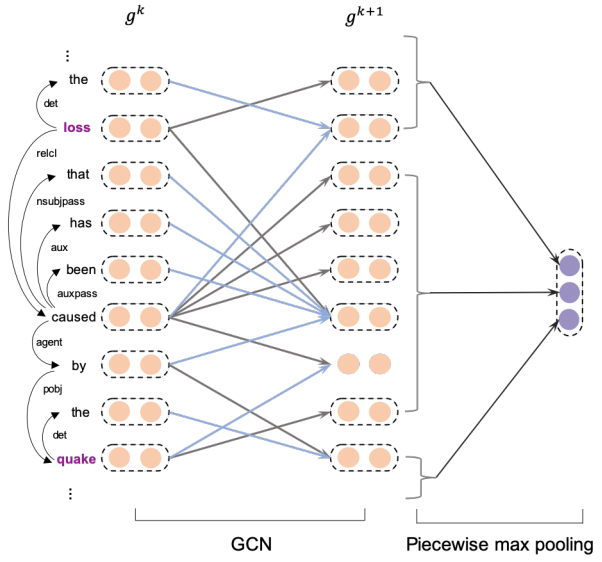

Fig. 3. is the detail of dependency sentence embedding.

the words. The graph convolution network proved to be very effective in the classification task [14], which captures information about neighboring nodes or neighbor nodes with a distance of k. We construct an $N * N$ adjacency matrix A where $A_{ij} = 1$, if word i to j have an edge on the dependency tree, otherwise $A_{ij} = 0$. Motivated by Marcheggiani and Titov et al [22], we employ three edge labels which include in forward edges(e_{ij}), backword edges(e_{ij}^{-1}) and self-loops($\circ$), where e_{ij} denote the forward edge from node i to node j, e_{ij}^{-1} represents the backword edge from node i to node j, and $\circ$ as a special symbol for self-loop. For each token w_{si} in sentence s, we employ GCN to get its hidden representation. The specific formula is as follows:

$$g_i^{k+1} = f(\sum_{j=1}^{N} A_{ij}(W^k g_i^k + b_k)) \tag{7}$$

Here, g_i^{k+1}denotes the hidden representation of the i^{th} word in a sentence in the $k+1$ layer GCN embedding, f refers to any non-linear activation function, $A = A + I$ with A being the $N * N$ adjacency matrix and I being the $N * N$ identity matrix, A_k is the model parameter that needs to be learned, and b_k is a bias term. Note that when $k = 0$, g_i^0 represents the initial representation of the token w_i, which is the output representation of the LSTM in our model.

1) Piecewise Max Pooling: As Fig. 3 shown, each hidden representation H is divided into three segments (H_1, H_2, H_3) by the two entities, where $H = (g_1, g_2, ..., g_N)$. We obtain an embedding H^{gcn} for the sentence s by max pooling of three segments separately.

$$H^{gcn} = max(H_1) \oplus max(H_2) \oplus max(H_3) \tag{8}$$

where $\oplus$ denotes vector concatenation.

C. Semantic Sentence Encoding

Furthermore, we employ CNN to extract the global features among $h_{1:N} = h_1, h_2, ..., h_N$, where $h_i \in \mathbb{R}^d$ represents the LSTM hidden state vector of the i^{th} word in the sentence. We apply a window of p hidden state vectors to a filter $w \in \mathbb{R}^{p \times d}$, which will generate a new feature. The feature c_i can be obtained by the following formula:

$$c_i = f(w \odot h_{i:i+p-1} + b) \tag{9}$$

Here f refers to a non-linear activation function, and we use the rectified linear unit(ReLU) in this paper. b is a bias term and $\odot$ is the dot product.

Apply this filter to the entire sentence to generate global feature $c = [c_2, c_2, ..., c_{N-p+1}]$. In general, CNN needs to extract features using multiple filters. Afterward, we utilized max-pooling to get the most important feature in the global feature c. The output of max pooling generated by i^{th} filter is as follows:

$$p_i = max(c) \tag{10}$$

To better explore relation indicate information, we use multi-gram CNN, which uses multiple types of filters to get the features. Finally, CNN's sentence embedding H^{cnn} can be obtained by the following formula:

$$H^{cnn} = (p_1, p_2, ..., p_k) \tag{11}$$

where k is the total number of filters. In this paper, we combine tri-gram, four-gram, and five-gram features with a concatenation operation.

D. Classification

The final sentence representation is obtained by concatenating the GCN's sentence encoding and CNN's sentence encoding, which contains not only the dependency information but also the deep semantic information.

$$H = H^{gcn} \oplus H^{cnn} \tag{12}$$

The final sentence representation can be directly classification by feeding it into a fully connected softmax layer. In order to prevent over-fitting, we utilized dropout [23] in the connected layer. The probability of target label:

$$p(y|x) = softmax(w \odot H + b) \tag{13}$$

where x is the input sentence, and y is the target label. We utilize the cross entropy and the L_2 regularization to define the objective function as follows:

$$J(\theta) = -\sum_{i=1}^{s}(y_i|x_i, \theta) + \beta\|\theta\|^2 \tag{14}$$

where s indicates the total sentence; x^i and y^i represent the sentence and label of the i^{th} training example; β is L_2 regularization hyperparameter. The θ is the whole network parameter, which can be learnable. In order to train the network parameter θ, we minimize the objective function by using the Adadelta optimizer.

TABLE I
Details of the SemEval-2010 task 8 and KBP37 dataset.

Datasets	Split	Sentences
SemEval-2010 task 8 (Relations:10)	Train	8000
	Test	2717
KBP37 (Relations:19)	Train	15917
	Test	3045

V. Experiments

In this section, we present our experiments in detail. Subsection V-A introduces the dataset and evaluation metrics. Subsection V-B describes the hyperparameter setting. Subsection V-C compares our LGCNN model with other methods in the literature and evaluate various versions of our model with removed some components.

A. Datasets

In our experiments, we evaluate the models on the SemEval-2010 task 8 and KBP37 dataset. The statistics of the two datasets are shown in Table I. Below we described each in detail.

1) SemEval: The SemEval-2010 task 8 dataset is an acknowledged benchmark for relation classification [1], which contains 8000 sentences for training and 2717 sentences for testing. The dataset includes ten categories, nine of which are directional, and one is non-directional.

2) KBP37: The KBP37 is a relatively large dataset in relation extraction. The KBP32 dataset contains 19322 sentences that 15917 sentences for training and 3405 sentences for testing. Because the dataset contains many special entities and relations, its classification is more complicated and confusing. In KBP37, a large number of entities are names of persons, places or organizations.

B. Parameter Settings

For different datasets, we use different parameters to achieve the best performance. In this work, we use Adadelta to update model parameters for the SemEval-2010 task 8 and KBP37 dataset. The parameters of the LGCNN model are summarized in Table II.

TABLE II
Hyper-parameter of LGCNN.

Parameter	SemEval-2010 8	KBP37
Word dimension	300	300
Position dimension	50	50
Batch size	20	20
Hidden size	500	300
Head size	4	2
Window size	$\{3, 4, 5\}$	$\{3, 4, 5\}$
Convolution size	128	200
GCN Layer dropout rate	0.8	0.8
Other dropout rate	0.5	0.5
GCN dimension	300	300
Regularization parameter	1e-5	1e-5
Learning rate	0.8	1
Adadelta decay rate	0.6	0.6

TABLE III
Details of the SemEval-2010 task 8 and KBP37 dataset.

Model	Features	SemEval-2010 8	KBP37
CNN [3]	WE,PF	78.3	51.3
CNN [3]	WE,PI	77.4	55.1
BLSTM+ATT [8]	WE,PI	84.0	61.2
LSTM+ATT [9]	WE,PF	85.2	61.7
SDP_LSTM [4]	WE,PF,WN,DEP	83.7	60.5
SPTree [26]	WE,PF,WN,DEP	84.4	–
PA_LSTM [25]	WE,PF,DEP	82.7	–
C_GCN [5]	WE,PF,DEP	84.8	62.0
LGCNN	WE,PF,PI,DEP	85.5	63.2

TABLE IV
Perfomance comparison of different version of LGCNN on two datasets

Model	SemEval-2010 8	KBP37
LSTM_CNN(w/o GCN)	84.9	61.7
LSTM_GCN(w/o CNN)	84.6	62.3
LGCNN	85.5	63.2

C. Overall Evaluation Results

We compare the F1-score of our model with some previous RE models in Table III.

In Table III, the WE, PF, PI, WN, DEP denote word vectors, position features, position indicators, wordnet, and dependency features respectively. The first four rows in the table represent the high performance achieved in RE by the End-to-End Model. There are CNN-based models such as CNN [3] and RNN-based models such as BLSTM-ATT [8] and LSTM-ATT [9] for this task. The next four rows in the table are neural network models based on the dependency tree. Since the SPTree and PA_LSTM models are not reproduced on the KBP dataset, the results are not presented. The last row is our model, which has achieved batter results on the two datasets.

To analyze the effect of the various components of LGCNN, we separately removed some of the components in LGCNN for evaluation. In Table IV, LSTM_CNN indicates that the classification is not performed using dependency sentence encoding in the LGCNN, and achieve the F1-score of 84.9% and 61.7%. LSTM_GCN represents that the model after removing the semantic sentence encoding component and its F1-score on the two datasets is 84.6% and 62.3%, respectively. The performance of LGCNN combining dependency features and semantic features is 85.5% and 63.2%. This shows that the combination of dependency features and semantic features is very effective in relation classification.

VI. Conclusion

This paper proposes a neural network, which combines dependency information and semantic information for relation classification. LGCNN makes use of SDP-based attention on GCN to extract dependency information, which can focus on the token in a sentence. Furthermore,we also

improves CNN with multiple types of filters to capture the semantic features on the context vector. Experimental results demonstrate that the proposed methods is superior to previous strategies, which effectively combines the features in the dependency tree with the features in the original sentence. By analyzing our results further, it can be found that our model enables two independent models to reinforce each other.

References

[1] I. Hendrickx, S. Nam Kim, Z. Kozareva, P. Nakov, M. Pennacchiotti, L. Romano, and S. Szpakowicz, "Semeval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals," in Proceedings of the 5th International Workshop on Semantic Evaluation, 2010, pp. 33–38.

[2] B. Plank, A. Moschitti, "Embedding semantic similarity in tree kernels for domain adaptation of relation extraction," in Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, 2013, pp. 1498–1507.

[3] D. Zeng, K. Liu, S. Lai, G. Zhou, and J. Zhao, "Relation classification via convolutional deep neural network," in Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers, 2014, pp. 2335–2344.

[4] Y. Xu, L. Mou, G. Li, Y. Chen, H. Peng, and Z. Jin, "Classifying relations via long short term memory networks along shortest dependency paths," in Proceedings of Conference on Empirical Methods in Natural Language Processing, 2015.

[5] Y. Zhang, P. Qi, and D. Christopher, "Graph Convolution over Pruned Dependency Trees Improves Relation Extraction," in Proceedings of Conference on Empirical Methods in Natural Language Processing, 2018.

[6] C. dos Santos, B. Xiang, and B. Zhou, "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, 2015, pp. 626–634.

[7] Y. Shen, X. Huang, "Attention-based convolutional neural network for semantic relation extraction," in Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, 2016, pp. 2526–2536.

[8] S. Zhang, D. Zheng, X. Hu, and M. Yang, "Bidirectional long short-term memory networks for relation classification," in Proceedings of the 29th Pacific Asia Conference on Language, Information and Computa- tion, 2015, pp. 73–78.

[9] P. Zhou, W. Shi, J. Tian, Z. Qi, B. Li, H. Hao, and B. Xu, "Attention-based bidirectional long short-term memory networks for relation classification," in Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, 2016, pp. 207–212.

[10] J. Lee, S. Se, and Y. Suk Choi, "Semantic Relation Classification via Bidirectional LSTM Networks with Entity-aware Attention using Latent Entity Typing," Symmetry, 2019.

[11] Y. Liu, F. Wei, S. Li, H. Ji, M. Zhou, and H. Wang, "A dependency-based neural network for relation classification," arXiv preprint arXiv:1507.04646, 2015.

[12] Y. Xu, R. Jia, L. Mou, G. Li, Y. Chen, and Y. Lu, "Improved Relation Classification by Deep Recurrent Neural Networks with Data," arXiv preprint arXiv:1601.03651, 2016.

[13] H. Q. Le, D. C. Can, Q. T. Ha, and N. Collier, "A Richer-but-Smarter Shortest Dependency Path with Attentive Augmentation for Relation Extraction," in 2019 Annual Conference of the North American Chapter of the Association for Computational Linguistics, 2019, pp. 2902-2912.

[14] T. N. Kipf and M.Welling, "Semi-supervised classification with graph convolutional networks," International Conference on Learning Representations, 2017.

[15] L. Song, Y. Zhang, Z. Wang, and D. Gildea, "A Graph-to-Sequence Model for AMR-to-Text Generation," ACL, 2018.

[16] D. Sorokin, I. Gurevych, "Modeling semantics with gated graph neural networks for knowledge base question answering," COLING, 2018.

[17] J. Bastings, I. Titov, and W. Aziz, "Graph Convolutional Encoders for Syntax-aware Neural Machine Translation," in Proceedings of Conference on Empirical Methods in Natural Language Processing, 2017.

[18] S. Vashishth, R. Joshi, and S. Prayaga, "RESIDE: Improving Distantly-Supervised Neural Relation Extraction using Side Information," in Proceedings of Conference on Empirical Methods in Natural Language Processing, 2018.

[19] D. Zhang, D. Wang, "Relation classification via recurrent neural network," arXiv preprint arXiv:1508.01006, 2015.

[20] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in Empirical Methods in Natural Language Processing, 2014.

[21] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, and I. Polosukhin, "Attention is all you need," in Advances in neural information processing systems, 2017, pp. 5998-6008.

[22] D. Marcheggiani and I. Titov, "Encoding sentences with graph convolutional networks for semantic role labeling," CoRR, 2017.

[23] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," Journal of machine learning research, 2014, pp. 1929–1958.

[24] Y. Yang, Y. Tong, S. Ma, and Z. Deng, "A position encoding convolutional neural network based on dependency tree for relation classification," in Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, 2016, pp. 65–74.

[25] P. Wang, Z. Xie, and J. Hu, "Relation classification via cnn, segmented max-pooling, and sdp-blstm," in International Conference on Neural Information Processing, 2017, pp. 144–154.

[26] M. Miwa and M. Bansal, "End-to-end relation extraction using LSTMs on sequences and tree structures," in Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, 2016.