Relation Extraction Model Based on Keywords Attention

Yu Chen
School of Computer Science
Hubei University of Technology
Hubei, China
1148848330@qq.com

Jianxia Chen
School of Computer Science
Hubei University of Technology
Hubei, China
chenjianxiawh@gmail.com

Chang Liu
School of Computer Science
Hubei University of Technology
Hubei, China
1323821376@qq.com

Qi Liu
School of Computer Science
Hubei University of Technology
Hubei, China
260129443@qq.com

***Abstract*—Recently, most relational extraction models usually mitigate the adverse effects of noise in sentences for the prediction results, utilizing different tools of natural language processing that to capture high-level features in sentences combined. However, these attention mechanisms do not manage to exploit as much as possible the semantic information of certain keywords that have relational expressive information in the sentence. Therefore, this paper proposes a model based on the keyword's attention mechanism, which is a novel attention mechanism based on the keywords of relational expression related. In particular, the proposed attention mechanism utilizes a linear-chain conditional random field that combines entity-pair features, similarity features between entity-pair features, and its hidden vectors to compute each word's marginal distribution defined as the attention weight. Experimental results show that the method can focus on keywords with relational expression semantics in sentences without using sophisticated tools and achieves performance improvements on the SemEval-2010 Task 8 dataset.**

Keywords-relation extraction; keywords attention; Hidden similarity; Bi-GRU

I. INTRODUCTION

RE(Relation extraction) is critical to NLP (natural language processing). To improve the performance of the models, researchers have tried several methods to remove the effects of noise [1, 2], including the removal of irrelevant words and methods based on attention mechanism. In contrast to traditional classification tasks, this task has to deal with noise in sentences.

In this paper, we propose an end-to-end bidirectional recurrent neural network-based model [3], called the REKA (RE Based on Keyword Attention) model, which uses a keyword-based attention mechanism. At the same time, our model avoids the accumulation of errors by not using any NLP tools, we use Gate Recurrent Unit (GRU) [4] to build a recurrent neural network to get the contextual information of the sentences. The keywords attention in the REKA model consists of two components: entity pair attention and segment attention, respectively. The paper utilized a linear-chain CRF (Conditional Random Field) [5] incorporating entity pair similarity calculations to calculate the marginal distribution of each state variable and consider it as an attention weight.

II. RELATED WORK

Recently, the models based on CNN (convolutional neural networks) [6, 7, 8] and RNN (recurrent neural network) [9] have become a major method for the RE research. The purpose of a CNN is to capture the local and continuous contextual content of a target, whereas an RNN accumulates contextual information in the input sentences via storage units. Socher *et al.* [10] proposed a RNN method that allows each node in the analysis tree to have a vector and a matrix, where the vector captures the intrinsic meaning of the component and the matrix captures how it changes the meaning of adjacent words or phrases. This matrix-vector RNN[10] can learn the meaning of operators in propositional logic and natural language, solving the problem that single word vector space models cannot capture the compositional meaning of long phrases, which prevents them from under-standing language in greater depth.

Hashimoto *et al*. [11] proposed a RNN model based on syntactic trees in 2013. Unlike the model proposed by Socher *et al*. [10], Hashimoto *et al*. did not use word dependency matrices, which are computationally expensive, but used additional features such as lexical (Part-of-Speech) labels, phrase categories, and syntactic heads, and introduced into the RNN model Hashimoto's model demonstrates the effectiveness of adding features and introducing averaging parameters to the RNN model to add weight to important phrases for the target task.

III. METHODOLOGY

As shown in Fig. 1, the proposed REKA is composed of the following four layers:

A. Input Layer

The input layer of the REKA model aims to convert the semantic and positional information of the input sentences into vectors. We uses $\{w_1, w_2, \ldots, w_n\}$ to denotes the input sentences and $\{p_1^{e_j}, p_2^{e_j}, \ldots, p_n^{e_j}\}$ denotes a vector of the relative position of every words to the entity pair e_j where $j \in \{1, 2\}$.

To enable the model to capture more accurate semantic information, the paper used d_w dimensional ELMo (Embedding from Language Model) word embedding pre-training model. Unlike previous work in which one word corresponds to a

DOI reference number: 10.18293/SEKE2021-073

vector that is stationary [12, 13], ELMo word vectors are no more just a vector correspondence, a real trained model [14].

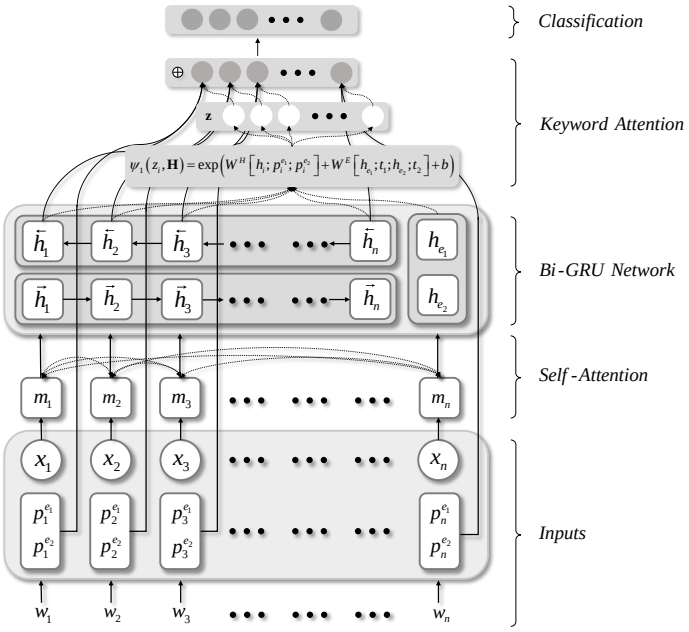

Figure 1. The systematic architecture of REKA model.

B. Multi-Head Attention Layer

We add a multi-headed attention mechanism after the input layer to better allow the model to understand the meaning of the context and to improve the performance of the model to solve the long-term dependencies problem [15, 16].

The calculation process for the multi-headed attention module is shown in the following equation:

$$\text{MultiHead}\ (Q,K,V) = W^M \text{Concat}[\text{head}_1;\ldots;\text{head}_r] \quad (1)$$

$$\text{where head}_i = \text{Attention}\left(W_i^Q Q, W_i^K K, W_i^V V\right) \quad (2)$$

$$\text{Attention}\ (Q,K,V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_w}}\right)V \quad (3)$$

Where ere $W^M \in \mathbb{R}^{d_w \times d_w}$, $W_i^Q \in \mathbb{R}^{d_w/r \times d_w}$, $W_i^K \in \mathbb{R}^{d_w/r \times d_w}$, $W_i^V \in \mathbb{R}^{d_w/r \times d_w}$ is the learn-able parameter of the multi-headed attention module, W^M is the output of the scaled dot-product attention when calculated and connected in series, W_i^Q denote query, W_i^K denote key, and W_i^V denote value of i^{th} head [15].

The inputs **Q**, **K**, **V** in the multi-head attention are all equivalent to the word embedding vector $\{x_1, x_2, \ldots, x_n\}$. The output vectors of the multi-head attention layer is feature vectors that contain contextual information about the input.

C. Bi-GRU Network

The Bi-GRU network layer is utilized to obtain contextual information about the output sequence of the multi-head attention layer which unit has fewer parameters and converges faster than the LSTM unit. For simplicity, the processing of m_i by the GRU unit is denoted in this paper as $GRU\left(m_i\right)$. The calculation process is shown as follows:

$$\overrightarrow{h_t} = \overrightarrow{GRU}\left(m_t\right) \quad (4)$$

$$\overleftarrow{h_t} = \overleftarrow{GRU}\left(m_t\right) \quad (5)$$

$$h_t = \left[\overrightarrow{h_t}; \overleftarrow{h_t}\right] \quad (6)$$

The input m_t of Bi-GRU is the output of the multi-head attention layer. To make effective use of both past and future features at a given time, the paper concatenates the hidden state vectors of the forward GRU network $\overrightarrow{h_t} \in \mathbb{R}^{d_h}$ at each time step with the hidden state vectors of the backward GRU network $\overleftarrow{h_t} \in \mathbb{R}^{d_h}$, the d_h is the dimension of the hidden state vector of the GRU network unit, and we use $\{h_1, h_2, \ldots, h_n\}$ to denote the hidden state vector of every word and use arrows to indicate the direction.

D. Keyword Attention

The keyword attention mechanism proposed in this paper aims to perform a soft selection of hidden layer vectors, and the attention weights are also a linear combination of a set of scalars. The weights are utilized to indicate the degree of attentions that the model should focus on a word of the sentence and it takes values between 0 and 1 in keyword attention mechanism. The proposed model defines a state variable *z*, in which it means that the corresponding word is irrelevant to the relational classification when *z* is equal to 0, or the word required for the relational expression in the sentence if *z* is equal to 1.Thus each sentence has its corresponding sequence of binary state variables *z*. According to this definition, the expected value of a hidden state **N**, will be selected and is calculated as follows:

$$\mathbf{N} = \sum_i p\left(z_i = 1 | \mathbf{H}\right)\mathbf{h}_i \quad (7)$$

To derive $p\left(z_i = 1 | \mathbf{H}\right)$, the CRF is introduced here to calculate the sequence of weights for the hidden sequence $\mathbf{H} = \{h_1, h_2, \ldots, h_n\}$, in which $\mathbf{H}$ represents the input sequence and h_i represents the hidden output of GRU for the i^{th} word. In particular, CRF provides a probabilistic framework for the computation of conditional probabilities from a sequence to another sequence. The linear-chain CRF defines a family of conditional probability $p\left(z_i = 1 | \mathbf{H}\right)$ given $\mathbf{H}$ with the following equation:

$$p(\mathbf{z}|\mathbf{H}) = \frac{1}{Z(\mathbf{H})}\prod_{c \in C}\psi\left(\mathbf{z}_c, \mathbf{H}\right) \quad (8)$$

$$Z(\mathbf{H}) = \sum_{\mathbf{z}' \in \mathcal{Z}}\prod_{c \in C}\psi\left(\mathbf{z}_c', \mathbf{H}\right) \quad (9)$$

Where $\mathcal{Z}$ denotes the set of state sequences *z*, $Z(\mathbf{H})$ is the normalization constant and $\mathbf{z_c}$ denotes the subset of *z* given by individual clique *c*, $\psi\left(\mathbf{z_c}, \mathbf{H}\right)$ is the potential function of this clique shown as the following equation:

$$\prod_{c \in C}\psi\left(\mathbf{z_c}, \mathbf{H}\right) = \prod_{i=1}^{n}\psi_1\left(z_i, \mathbf{h}_i\right)\prod_{i=1}^{n-1}\psi_2\left(z_i, z_{i+1}\right) \quad (10)$$

This paper utilized two types of feature function for calculation, the vertex feature function $\psi_1\left(z_i, \mathbf{H}\right)$ and the edge feature function $\psi_2\left(z_i, z_{i+1}\right)$. ψ_1 represents the mapping of the output *h* of GRU to the state variable *z*, and ψ_2 simulates the transition

of two state variables at adjacent time steps. The equations for their definitions are shown as follows respectively:

$$\psi_1(z_i, \mathbf{H}) = \exp\left(\mathbf{W}^H F_1 + \mathbf{W}^E F_2 + b\right) \quad (11)$$

$$F_1 = [h_i; p_i^{e_1}; h_{e_2}; t_2];\ F_2 = [h_{e_1}; t_1; h_{e_2}; t_2] \quad (12)$$

$$\psi_2(z_i, z_{i+1}) = \exp\left(\mathbf{W}^t_{z_i, z_{i+1}}\right) \quad (13)$$

Where $\mathbf{W}^H$ and $\mathbf{W}^E$ are learnable parameters of a linear transformation, and b is a bias term. They map the contextual information in the sentence into feature scores for each state variable, which makes use of the relative entity position features $p_i^{e_1}$ $p_i^{e_2}$ in the sentence and keyword features (entity pair features h_{e_1}, h_{e_2} and entity pair hidden similarity features t_1 and t_2).

*1) Entity position feature:*Relative location features $p_i^{e_1}$, $p_i^{e_2}$ are utilized to jointly represent contextual information as well as entity location relationships by concatenating them with the hidden layer outputs h_i, as shown by F_1 in Equation 12. There is a definition such as $p_i^{e_1}, p_i^{e_2} \in \mathbb{R}^{d_p}$, $e_j \in \{1, 2\}$.

Positional embedding is similar to word embedding vectors in which it transforms a relative positional scalar into a vector by traversing through the embedding matrix $W_{pos} \in \mathbb{R}^{d_p \times (2L-1)}$, where L is the length of each piece of data in the dataset, and d_p is the dimension of the position vectors.

2) Entity hidden similarity features: Since entity words in sentences are inherently strong cues for relational classification, most of NLP tools were utilized in Zeng *et al*. [7] to obtain linguistic features of entity words, however, this approach is not an end-to-end model anymore. Therefore, this paper proposes a method to extract entity features that avoid the use of traditional NLP tools, and such features are named entity hidden similarity features in this paper, the calculation procedure is shown in Equation 14, 15.

$$a_i^j = \frac{\exp\left(\left(h_{e_j}\right)^\top c_i\right)}{\sum_{k=1}^{K} \exp\left(\left(h_{e_j}\right)^\top c_k\right)} \quad (14)$$

$$t_{j \in \{1,2\}} = \sum_{i=1}^{K} a_i^j c_i \quad (15)$$

In this paper, entity words are categorized based on the similarity between the embedding vector and the hidden vector of the entity words. Where $c \in \mathbb{R}^{2d_h \times K}$ is a potential vector constructed in the potential vector space to represent the classes of similar entities, where K is the number of classes in which entities are classified by their hidden similarities.

The hidden similarity feature t_j of the j^{th} entity is calculated by weighting the similarity of c with the output h_{e_j} of the hidden layer based on the j^{th} entity. Entity features are constructed by cascading the hidden states corresponding to the entity location and the potential type representation of the entity pair, as shown by F_2 in Equation 12.

E. Classification Layer

To calculate the probability p of the output distribution of the state variable (the conditional probability of all relations), a *softmax* layer has been added after the keyword attention layer, which is calculated as shown in the Equation 16.

$$p(y|\mathbf{N}) = softmax\left(\mathbf{W}_y \mathbf{N} + b_y\right) \quad (16)$$

Where $b_y \in \mathbb{R}^{|R|}$ is a bias term, $|R|$ is a number of relationship categories, $\mathbf{W}_y$ maps the expected value of a hidden state **N** to the feature score for relation labels.

F. Training

With the introduction of the keyword attention mechanism, the model in this paper is shown in Fig. 1. This attention is calculated about to with concerning the cross-entropy loss of the RE. This loss function is defined as shown in Equation 17.

$$\mathcal{L}' = -\sum_{i=1}^{|D|} \log p\left(y^{(i)} \middle| S^{(i)}, \theta\right) \quad (17)$$

Where $|D|$ is the size of the training data set, $(S^{(i)}, y^{(i)})$ is the i^{th} sample in the data set. In this paper, the AdaDelta optimizer is used to minimize the loss calculation parameter θ.

L2 regularization is added to the loss function to prevent overfitting, and λ_1, λ_2 are hyper-parameters of regularizes. The second regularizer tries to force the model to process the few words that matter and returns a sparse weight distribution. The final objective function $\mathcal{L}$ is shown in Equation 18.

$$\mathcal{L} = \mathcal{L}' + \lambda_1 \|\theta\|_2^2 + \lambda_2 \sum_{i}^{n} p\left(z_i = 1 \middle| \mathbf{H}\right) \quad (18)$$

IV. EXPERIMENTS

A. Dataset and Metric

Our experiments were evaluated on the SemEval-2010 Task 8 dataset. The dataset has 19 relationship types 10717 sentences, including 8000 samples for training and 2717 samples for testing. The evaluation metrics used in this paper are based on the macro-averaged F1.

B. Implementation Details

In this paper, the word embeddings used as input in the REKA model are trained using the publicly available pre-trained EMLo model, and all other parameters in the model are randomly initialized by the zero-mean Gaussian distribution, the hyperparameters are shown in Tab. I.

C. Comparison Models

The proposed REKA model is compared with the following benchmark model such as SVM [6], MV-RNN [10], CNN [7], BLSTM [18], DepNN [19], FCM [20], SDP-LSTM [21].

TABLE I. HYPERPARAMETERS SETTING.

Hyper-parameter	Description	Value
dropout rate	Keyword attention layer	0.5
	Bi-GRU layer	0.6
	Word embedding layer	0.8
	Multi-head attention layer	0.8
λ_1	Regularization coefficient[a]	[0, 0.2]
λ_2		
r	Number of Heads	4
batch size	Size of mini-batch	50
η	Initial learning rate	4
d_r	The decay rate of leaning	0.5
d_a	Size of attention layer	50
d_h	Size of hidden layer	512
K	Number of the similar entities' classes	4
d_p	Size of position embeddings	50

a. (The regularization coefficient values of λ_1 and λ_2 are selected from 0 to 0.2 using grid search.)

D. Experimental Results

To further evaluate the proposed model, we selected the RNN-based model from the above models for comparison. The average precisions (AP) of REKA compared with RNN methods are shown in Tab. II. The results of REKA model compared to other models are shown in Tab. III, From the experimental results, the proposed REKA model outperforms the existing model using the smaller number of features, with a relative improvement of 1.1%, indicating that the keyword attention mechanism can improve the performance of the model.

TABLE II. AVERAGE PRECISION SCOREFOR OUR MODEL AND COMPARED METHODS (MICRO-AVERAGED OVER ALL CLASSES)

[a]	BLSTM	SDP-LSTM	REKA
1%	0.26	0.47	**0.55**
20%	0.60	0.68	**0.76**
100%	0.73	0.70	**0.81**

a. (The first columns show how much of testing data has been used. Performance is on the SemEval-2010 task dataset)

TABLE III. COMPARATIVE RESULTS ON SEMEVAL-2010 TASK 8 TEST DATASET.

Model	Additional Features[a]	F1
SVM[6]	POS, WN, etc.	82.3
MV-RNN[10]	POS, NER, WN	82.4
CNN[7]	PE, WN	82.7
BLSTM[18]	None,	82.7
	+ PF, POS, etc.	84.3
DepNN[19]	DEP	83.6
FCM[20]	SDP, NER	83.0
SDP-LSTM[21]	SDP	83.7
REKA Model	**PE**	**84.8**

a. (Where WN, DEP, SDP, PE are WordNet, dependency features, shortest dependency path, position embeddings, respectively)

V. CONCLUSION

In this paper, we propose an end-to-end Bi-GRU network model based on a keyword attention mechanism. The model fully extracts the features available in the dataset using the keyword attention mechanism and achieves an F1 of 84.8 without the use of other tools for natural language processing. In the keyword attention mechanism, we use the relative position vectors of entity pairs and the similarity between entity pairs and their hidden vectors for computing the marginal distribution of each word, which is chosen as the attention weight. In the future, we will further investigate attention mechanisms that can better extract key information from sentences and plan to use them for the recognition of relationships between multiple entities.

REFERENCES

[1] Zhang, Y.; Zhong, V.; Chen, D.; Angeli, G.; Manning, C.D. Position-aware attention and supervised data improve slot filling. In Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, Copenhagen, Denmark, 7–11 September 2017; pp. 35–45.

[2] Liu T, Zhang X, Zhou W, et al. Neural RE via inner-sentence noise reduction and transfer learning[J]. arXiv preprint arXiv:1808.06738, 2018.

[3] Lee J, Seo S, Choi Y S. Semantic relation classification via bidirectional lstm networks with entity-aware attention using latent entity typing[J]. Symmetry, 2019, 11(6): 785.

[4] Lee J, Seo S, Choi Y S. Semantic relation classification via bidirectional lstm networks with entity-aware attention using latent entity typing[J]. Symmetry, 2019, 11(6): 785.

[5] Lafferty J, McCallum A, Pereira F C N. Conditional random fields: Probabilistic models for segmenting and labeling sequence data[J]. 2001.

[6] Rink B, Harabagiu S. Utd: Classifying semantic relations by combining lexical and semantic resources[C]//Proceedings of the 5th International Workshop on Semantic Evaluation. 2010: 256-259.

[7] Zeng D, Liu K, Lai S, et al. Relation classification via convolutional deep neural network[C]//Proceedings of COLING 2014, the 25th international conference on computational linguistics: technical papers. 2014: 2335-2344.

[8] Santos C N, Xiang B, Zhou B. Classifying relations by ranking with convolutional neural networks[J]. arXiv preprint arXiv:1504.06580, 2015.

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

[10] Socher R, Huval B, Manning C D, et al. Semantic compositionality through recursive matrix-vector spaces[C]//Proceedings of the 2012 joint conference on empirical methods in natural language processing and computational natural language learning. 2012: 1201-1211.

[11] Hashimoto K, Miwa M, Tsuruoka Y, et al. Simple customization of recursive neural networks for semantic relation classification[C]//Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing. 2013: 1372-1376.

[12] Mikolov T, Chen K, Corrado G, et al. Efficient estimation of word representations in vector space[J]. arXiv preprint arXiv:1301.3781, 2013.

[13] Pennington J, Socher R, Manning C D. Glove: Global vectors for word representation[C]//Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP). 2014: 1532-1543.

[14] Peters M E, Neumann M, Iyyer M, et al. Deep contextualized word representations[J]. arXiv preprint arXiv:1802.05365, 2018.

[15] Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need[J]. arXiv preprint arXiv:1706.03762, 2017.

[16] Shen T, Zhou T, Long G, et al. Disan: Directional self-attention network for rnn/cnn-free language understanding[C]//Proceedings of the AAAI Conference on Artificial Intelligence. 2018, 32(1).

[17] Tan Z, Wang M, Xie J, et al. Deep semantic role labeling with self-attention[C]//Proceedings of the AAAI Conference on Artificial Intelligence. 2018, 32(1).

[18] Zhang S, Zheng D, Hu X, et al. Bidirectional long short-term memory networks for relation classification[C]//Proceedings of the 29th Pacific Asia conference on language, information and computation. 2015: 73-78.

[19] Liu Y, Wei F, Li S, et al. A dependency-based neural network for relation classification[J]. arXiv preprint arXiv:1507.04646, 2015.

[20] Yu M, Gormley M, Dredze M. Factor-based compositional embedding models[C]//NIPS Workshop on Learning Semantics. 2014: 95-101.

[21] Xu Y, Mou L, Li G, et al. Classifying relations via long short term memory networks along shortest dependency paths[C]//Proceedings of the 2015 conference on empirical methods in natural language processing. 2015: 1785-1794.