Bidirectional Semantic Matching with Deep Contextualized Word Embedding for Chinese Sentence Matching

Kunxun Qi, Jianfeng Du[*], Qiqi Ou, Linxi Jin and Jinglan Zhong

School of Computer Science and Technology,
Guangdong University of Foreign Studies, Guangzhou 510006, China
jfdu@gdufs.edu.cn

Abstract. In this paper, a bidirectional matching model is proposed to identify whether two Chinese sentences are paraphrases of each other. The model is adapted from the well-known BiMPM model on two main aspects. On the one hand, it exploits a deep contextualized model named ELMo to generate the input word embedding. On the other hand, three out of four bidirectional matching mechanisms in BiMPM are carefully selected to model interaction between two sentences. The proposed model is evaluated on a dataset about Chinese sentence pairs from CCKS 2018. Experimental results show that the model achieves 86.2% F1-score on the validation set and 84.6% F1-score on the test set.

Keywords: Sentence Matching, Chinese Sentence Pairs, Deep Neural Network.

1 Introduction

Modeling two natural language sentences is a fundamental task in many natural language processing (NLP) tasks, such as paraphrase identification (PI) [3], textual entailment(TE) [3] and etc. In paraphrase identification task, we identify whether two sentences are paraphrase or not. In text entailment task, we estimate whether a sentence can be inferred from another sentence.

In recent years, neural network models have been widely used in modeling sentence pairs. Two advanced frameworks have been proposed in previous work. The first framework usually implements two weight sharing sentence encoders, such as Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN), to represent a sentence pair as two low-dimensional real-value vectors u1, u2 and then makes a prediction based on the two vectors. This framework usually constructs a feature vector, such as (u1, u2, $|u1 - u2|$, $u1 * u2$), feeding it into a fully-connected network followed by a softmax layer to make final prediction. Some typical methods in this framework include BCNN [3], InferSent [4] and SWEMs [5]. This framework pays more attention on constructing sentence encoder, but ignores the relevance between

[*] Corresponding author.

two sentences. Existing empirical studies reveal that this framework cannot achieve the state-of-the-art performance. This limitation may be caused by the losing of some interactive information between the two sentences. To further improve the performance, the second framework studies how to learn interaction between two sentences. This framework usually calculates the relevance between the two sentences by using a variety of attention mechanisms. The prominent methods in this framework include ABCNN [3], ESIM [6] and BiMPM [2]. In this paper, we implement three out of four bidirectional matching mechanisms in BiMPM to calculate the interaction between two sentences, including full-matching, attentive-matching and max-attentive-matching. We do not use the maxpooling-matching mechanism because it is time consuming and hard to be evaluated in our experiments.

All the above approaches use word embedding as input. Word embedding aims to represent the tokens from textual documents as low-dimensional real-value vectors. As known that, word embedding has been widely used in a broad range of NLP tasks, such as named entity extraction (NER), part-of-speech tagging (POS Tagging), question answering (QA), textual entailment (TE), machine comprehension (MC), etc. The most famous word embedding models are Word2vec [7] and GloVe [8], which have demonstrated advanced performance in a variety of NLP tasks. However, most of these word embedding models generate pre-trained word vectors for each natural language token in training corpus, which means that the out of vocabulary (OOV) words have no representation. One common solution is to initialize the word embedding randomly and update the word vectors during training. It is easy to incur overfitting. Another solution is using the N-gram features in training the word embedding. For example, FastText [9] trains word embedding by predicting the labels of documents. It is applicable to the document classification task but is not suitable for sentence modeling tasks. Recently, a new type of deep contextualized word representation, ELMo [1], has been proposed to address wrongly written or mispronounced characters, wrongly Chinese word segmentation and OOV words. It has been demonstrated to improve the performance in six challenging NLP tasks [1]. ELMo generates word vectors based on the input of character sequences and the representations of the contextualized words in a sentence. In this paper, we train an ELMo model on Chinese Wikipedia corpus and use it to generate word vectors.

In this study, our model is evaluated on the dataset about Chinese sentence pairs from CCKS 2018. Experimental results show that the model achieves 86.2% F1-score on the validation set as well as 84.6% F1-score on the test set.

2 Related work

There are lots of studies for modeling sentence pairs. In this section, we only make a review on previous deep learning methods. We refer the interested reader to [3] for other methods. There are two major deep learning frameworks for modeling sentence pairs, namely the classical encoding framework and attention-based encoding framework.

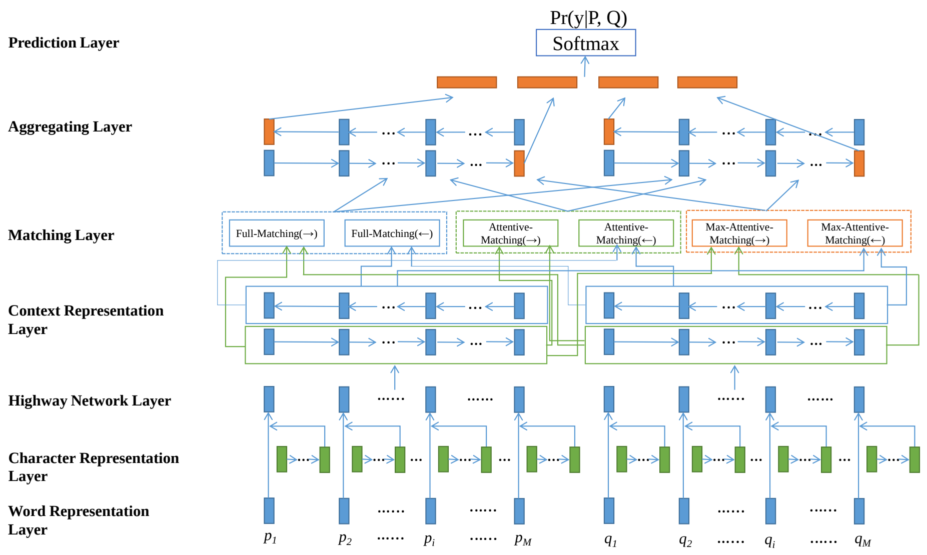

Fig. 1. The overview architecture of our model for Chinese sentence pair matching.

2.1 Classical Encoding Framework

Methods in this framework employ two weight-sharing classical encoders, suach as CNN or RNN, to generate two vector representations for the two input sentences. BCNN [3] used two weight sharing CNNs to generate two sentence representations and constructed a feature vector by connecting the two vectors. [4] implemented two bidirectional LSTM (BiLSTM) networks as sentence encoders. SWEMs [5] employed two hierarchical pooling encoders instead of using any CNNs or RNNs. [11] modeled sentence pairs by using Transformer [12] encoder, which is a recent network architecture that makes use of self-attention [12] mechanism.

2.2 Attention-based Encoding Framework

On the basis of the first framework, methods in this framework employ various attention mechanisms that are based on the similarity between two sentences to adjust the two representations. ABCNN [3] enhanced the BCNN [3] by employing an attention feature matrix to learn interactive information. ESIM [6] employed Tree-LSTM (Long Short-Term Memory) as sentence encoder. It calculated the relevance between two sentences by applying a local inference modeling layer. BiMPM [2] proposed four effective bidirectional matching mechanisms to learn the interactive information.

3 Adaptation of BiMPM with ELMo

Our proposed model is shown in Figure 1. The input of our model has two parts for each sentence. The first part is the word embedding generated by ELMo. The second part is the character embedding created by a bidirectional LSTM (BiLSTM) network on randomly instantiated character embedding. The concatenated vector from these two parts are fed into a Highway network [13] to generate two sequences of word vectors. The two sequences of word vectors are fed into the contextual representation layer to learn the contextual representations. Three bidirectional matching mechanisms in BiMPM are employed in the matching layer to calculate the interaction between two sentences. The two matching vectors are fed into the aggregating layer to generate the feature vectors, which are used to make prediction in the prediction layer.

3.1 Word Representation Layer

This layer generates a d-dimensional vector for each word within the experimental sentences. There are two parts in this layer. The first part is ELMo generated word embedding. We train an ELMo model on Chinese Wikipedia articles corpus[1] and use it to generate word vectors. The second part is the character embedding. We initialize fixed dimensional vectors randomly for each character within a word. They are fed into a BiLSTM network to compose word vectors. We pick the last hidden state of the BiLSTM network as the representation of each word. We feed the concatenated vectors from these two parts into a Highway network to generate the final word vectors.

3.2 Contextual Representation Layer

This layer generates the context representation of two sentences by using two BiLSTM networks. The weights in these two networks are shared during training.

3.3 Matching Layer

This layer calculates the interactive information between two sentences. We apply three out of four bidirectional matching mechanisms in BiMPM, including the full-matching mechanism, the attentive-matching mechanism and the max-attentive-matching mechanism. For details, we use function f_m to calculate the relevance between two contextual representations.

$$\boldsymbol{m} = f_m(v_1,\ v_2;\ \boldsymbol{W}) \tag{1}$$

In eq(1), v_1 and v_2 are the hidden states of the two BiLSTM networks in contextual representation layer. Both v_1 and v_2 are d-dimensional vectors. $W \in \mathrm{R}^{l \times d}$ is a trainable parameter and l is a hyperparameter that means the perspective of the interactive features. For each element $m_k \in \boldsymbol{m}$, k means the k-th dimension of the interactive vector. They are calculated by a cosine similarity function

$$m_k = cosine(W_k \circ v_1,\ W_k \circ v_2) \tag{2}$$

where $\circ$ is the element-wise multiplication and W_k is the k-th row in W.

Further, we apply three bidirectional matching mechanisms to calculate the interac-

[1] https://zh.wikipedia.org/wiki/

tive features of each time-step of sentence against all time-steps of the other sentence:

Full-Matching. This matching mechanism calculates the interactive features between each contextual representation $\overrightarrow{h}_i^p$ (or $\overleftarrow{h}_i^p$) and the last time-step of the contextual representation of the other sentence $\overrightarrow{h}_N^q$ (or $\overleftarrow{h}_1^q$).

$$\overrightarrow{m}_i^{full} = f_m\left(\overrightarrow{h}_i^p,\ \overrightarrow{h}_N^q;\ \boldsymbol{W^1}\right)$$
$$\overleftarrow{m}_i^{full} = f_m\left(\overleftarrow{h}_i^p,\ \overleftarrow{h}_1^q;\ \boldsymbol{W^2}\right) \tag{3}$$

Attentive-Matching. This matching mechanism calculates interactive feature between each contextual representation and the weighted summing contextual representation of the other sentence. Firstly, we calculate the similarity between two contextual representations.

$$\overrightarrow{\alpha}_{i,j} = cosine\left(\overrightarrow{h}_i^p,\ \overrightarrow{h}_j^q\right)$$
$$\overleftarrow{\alpha}_{i,j} = cosine\left(\overleftarrow{h}_i^p,\ \overleftarrow{h}_j^q\right) \tag{4}$$

Then, we use $\overrightarrow{\alpha}_{i,j}$ (or $\overleftarrow{\alpha}_{i,j}$) as the weight of $\overrightarrow{h}_j^q$ (or $\overleftarrow{h}_j^q$). We generate an attentive contextual representation by weighted summing all time-steps of the contextual representations of the other sentence.

$$\overrightarrow{h}_i^{mean} = \frac{\sum_{j=1}^{N} \overrightarrow{\alpha}_{i,j} \cdot \overrightarrow{h}_j^q}{\sum_{j=1}^{N} \overrightarrow{\alpha}_{i,j}}$$
$$\overleftarrow{h}_i^{mean} = \frac{\sum_{j=1}^{N} \overleftarrow{\alpha}_{i,j} \cdot \overleftarrow{h}_j^q}{\sum_{j=1}^{N} \overleftarrow{\alpha}_{i,j}} \tag{5}$$

Finally, we calculate the interactive features between each contextual representation $\overrightarrow{h}_i^p$ (or $\overleftarrow{h}_i^p$) and the attentive contextual representation $\overrightarrow{h}_i^{mean}$ (or $\overleftarrow{h}_i^{mean}$)..

$$\overrightarrow{m}_i^{att} = f_m\left(\overrightarrow{h}_i^p,\ \overrightarrow{h}_i^{mean};\ \boldsymbol{W^5}\right)$$
$$\overleftarrow{m}_i^{att} = f_m\left(\overleftarrow{h}_i^p,\ \overleftarrow{h}_i^{mean};\ \boldsymbol{W^6}\right) \tag{6}$$

Max-Attentive-Matching. This matching mechanism uses the most similar contextual representation as the attentive representation with max cosine similarity.

$$\overrightarrow{h}_i^{max} = g_{max}(cosine\left(\overrightarrow{h}_i^p,\ \overrightarrow{h}_j^q\right))$$
$$\overleftarrow{h}_i^{max} = g_{max}(cosine\left(\overleftarrow{h}_i^p,\ \overleftarrow{h}_j^q\right)) \tag{7}$$

Function g_{max} generates attentive representation $\overrightarrow{h}_i^{max}$ (or $\overleftarrow{h}_i^{max}$) by picking the highest cosine similarity between the two contextual representations.

$$\overrightarrow{m}_i^{max} = f_m\left(\overrightarrow{h}_i^p,\ \overrightarrow{h}_i^{max};\ \boldsymbol{W^7}\right)$$

$$\overleftarrow{m}_i^{max} = f_m\left(\overleftarrow{h}_i^p,\ \overleftarrow{h}_i^{max};\ \boldsymbol{W}^{\boldsymbol{8}}\right) \tag{8}$$

We calculate the interactive feature between each contextual representation $\overrightarrow{h}_i^p$ (or $\overleftarrow{h}_i^p$) and the max-attentive contextual representation $\overrightarrow{h}_i^{max}$ (or $\overleftarrow{h}_i^{max}$).

3.4 Aggregating Layer

This layer employs two BiLSTM networks to generate the feature vector individually. The four last hidden states of the BiLSTM networks are used to compose the feature vector.

3.5 Prediction Layer

This layer employs a two-layer feed-forward network and a softmax transformation function to calculate the probability distribution *Pr(y|P, Q).*

4 Experiments

4.1 Dataset and Evaluation

In the CCKS 2018 challenge, the organizers provided 100, 000 labeled Chinese sentence pairs for the training set, 10, 000 unlabeled sentence pairs for the validation set and 110, 000 unlabeled sentence pairs for the test set.

All the evaluation results are calculated by an official evaluation system for the CCKS 2018 challenge. The evaluation system computes four metrices including micro-average precision (Prec.), recall (Rec.), F1-score (F1) and accuracy (Acc.) on the validation set and the test set.

4.2 Experiments Settings and results

We train a ELMo model on 3.3GB Chinese Wikipedia corpus. Both the corpus and the dataset are processed by Jieba2 tool for Chinese word segmentation. We use the ELMo generated word vectors to initialize the word embedding layer and do not update them during training. We initialize the 20-dimensional character vectors randomly. We utilize a 1-layer Highway network to generate the final word representation. We set the hidden size as 100 for all BiLSTM networks. We employ a dropout for each layer in Figure 1 and set the dropout ratio as 0.5. We set the learning rate as 0.0005 for Adam optimizer and 3 for Adadelta optimizer. We generate three results by applying Adadelta optimizer twice and Adam optimizer once. We apply a vote mechanism on the three result to generate the final prediction.

Table 1 shows the performances of some prior methods on validation set. We evaluate seven state-of-the-art models as baseline. We can see that models in the sentence pair matching framework have a better performance than those in the sentence encod-

[2] https://github.com/fxsjy/jieba

Table 1. Performances of various prior methods on validation set

	Methods	Prec.	Rec.	F1	Acc.
Classical encoding framework	BCNN	78.6%	79.7%	79.1%	78.9%
	SWEMs	82.5%	80.9%	81.7%	81.9%
	Transformer-based encoders	81.8%	84.1%	82.9%	82.7%
	BiLSTM-Attention encoders	78.5%	87.1%	82.6%	81.6%
Attention based encoding framework	ABCNN-2	79.4%	84.9%	82.0%	81.4%
	ABCNN-2 (Multi-Perspective)	80.6%	84.9%	82.9%	81.5%
	ESIM	83.7%	82.8%	83.2%	83.3%
	BiMPM	84.1%	83.2%	83.5%	83.6%
	Our model	**86.3%**	83.7%	85.0%	85.2%
	Our model (Vote)	85.0%	**87.4%**	**86.2%**	**86.1%**
	Our model on test set	83.2%	86.0%	84.6%	84.3%

ing-based framework. All the baseline models use word2vec embedding as input. In classical encoding framework, we apply four sentence encoders, including CNN, Hierarchical Pooling, Transformer and BiLSTM. Transformer and BiLSTM have a better performance, achieving 82.9% and 82.6% F1-scores. In the attention based encoding framework, we evaluate four baseline models, including ABCNN-2, ABCNN-2 (Multi-Perspective), ESIM and BiMPM. ABCNN-2 (Multi-Perspective) is the implement of ABCNN-2 model with different kernel sizes. We can see that ESIM and BiMPM achieve better performances than ABCNN. Our model achieves highest performance in single model with 85.0% F1-score. Finally, we employ a vote mechanism to merge different results of our model. It achieves the best performance in the validation set with 86.2% F1-score and achieve 84.6% F1-score in the test set.

5 Conclusion

In this study, we have proposed a model adapted from BiMPM. The model implements three out of four bidirectional matching mechanisms in BiMPM and exploits ELMo to generate word embedding. The final prediction of our adapted model is given by voting three results of our model with different hyperparameters. We evaluate our model on the dataset about Chinese sentence pairs from CCKS 2018. Experimental results reveal that the model achieves 86.2% F1-score on the validation set and 84.6% F1-score on the test set, ranking the fifth in this challenge.

Acknowledgements

This work was partly supported by National Natural Science Foundation of China (61375056), Science and Technology Program of Guangzhou (201804010496), and

Scientific Research Innovation Team in Department of Education of Guangdong Province (2017KCXTD013).

References

1. Peters, M., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., Zettlemoyer, L.: Deep Contextualized Word Representations. In: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT). pp. 2227–2237 (2018)
2. Wang, Z., Hamza, W., Florian, R.: Bilateral Multi-Perspective Matching for Natural Language Sentences. In: Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence (IJCAI). pp. 4144-4150 (2017)
3. Yin, W., Schütze, H., Xiang, B., Zhou, B.: ABCNN: Attention-Based Convolutional Neural Network for Modeling Sentence Pairs. Transactions of the Association for Computational Linguistics vol.4, 259-272 (2016)
4. Conneau, A., Kiela, D., Schwenk, H., Barrault, L., Bordes, A.: Supervised Learning of Universal Sentence Representations from Natural Language Inference Data. In: Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP). pp. 670-680 (2017)
5. Shen, D., Wang, G., Wang, W., Min, M. R., Su, Q., Zhang, Y., Li, C., Henao, R., Carin, L.: Baseline Needs More Love: On Simple Word-Embedding-Based Models and Associated Pooling Mechanisms. In: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL). pp. 440–450 (2018)
6. Chen, Q., Zhu, X., Ling, Z., Wei, S., Jiang, H., Inkpen, D.: Enhanced LSTM for Natural Language Inference. In: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL). pp. 1657-1668 (2017)
7. Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., Dean, J.: Distributed Representations of Words and Phrases and their Compositionality. In: Proceedings of the 27th Annual Conference on Neural Information Processing Systems (NIPS). pp. 3111-3119 (2013)
8. Pennington, J., Socher, R., Manning, C. D.: Glove: Global vectors for word representation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). pp. 1532-1543 (2014)
9. Bojanowski, P., Grave, E., Joulin, A., Mikolov, T.: Enriching Word Vectors with Subword Information. Transactions of the Association for Computational Linguistics vol.4, 135-146 (2017)
10. Bowman, S. R., Angeli, G., Potts, C., Manning, C. D.: A large annotated corpus for learning natural language inference. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP). pp. 632-642 (2015)
11. Yang, Y., Yuan, S., Cer, D., Kong, S.Y., Constant, N., Pilar, P., Ge, H., Sung, Y.h., Strope, B., Kurzweil, R.: Learning Semantic Textual Similarity from Conversations. In: Proceedings of The Third Association for Computational Linguistics Workshop on Representation Learning for NLP. pp. 164-174 (2018)
12. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., Polosukhin, I.: Attention Is All You Need. In: Proceedings of the 30th Annual Conference on Neural Information Processing Systems (NIPS). pp. 6000-6010 (2017)
13. Zilly, J. G., Srivastava, R. K., Koutník, J., Schmidhuber, J.: Recurrent Highway Networks. In: Proceedings of the 34th International Conference on Machine Learning (ICML). pp. 4189-4198 (2017)