An Ensemble Model Based on Siamese Neural Networks for the Question Pairs Matching Task

Shiyao Xu, Shijia E, and Yang Xiang

Tongji University, Shanghai 201804, P.R. China,
{xushiyao,436_eshijia,shxiangyang}@tongji.edu.cn

Abstract. The problem of question pairs matching aims to seek whether the underlying semantics of two questions are equivalent. For WeBank Chinese question pairs which are collected from real-world intelligent customer service questions, the goal is to identify question pairs that have the same intent. In this paper, we propose an ensemble model which based on both word and character level neural networks such as the convolutional neural network (CNN), and the long short-term memory network (LSTM) for modeling semantic similarity. And we adopt an enhanced deep semantic model (R-ESIM) which is proved to be more effective for sentence modeling. Our model takes 10-fold cross-validation into account to improve the generalization ability. In the evaluation of the CCKS 2018 shared task three, our model achieves the F1 score of 0.85085 for the opening test data which ranks the second.

Keywords: semantic similarity, siamese neural network, word embedding, char embedding.

1 Introduction

Semantic textual similarity plays an important role in natural language processing (NLP). It is the basis of many NLP tasks such as question answering and information retrieval. In recent years, there are more and more English semantic similarity tasks such as Quora Question Pairs in Kaggle and Semantic Textual Similarity (STS) in SemEval. The CCKS 2018 WeBank intelligent customer service question pairs matching task provides a Chinese dataset similar to the Quora question pairs. We need to assess the degree of underlying semantic similarity between two questions and identify whether the two questions have the same intent. Similar question pairs are labeled as 1 and 0 otherwise. The question matching task is challenging not only due to the short text with less semantic information but also due to many typos from real users.

There have been numerous methods to solve the problem of similarity. LSI [3] and LDA [1] are typical traditional methods. Latent Semantic Indexing(LSI) maps words and documents to latent semantic space by SVD to solve the problem of polysemy and synonymy. Latent Dirichlet Allocation (LDA) is a topic model that represents documents by the probability distribution of topics. A series of neural network models for sentence matching have emerged with the

development of deep learning. Most of them use word embedding as input, convert word embedding to sentence representation by a siamese base network (CNN or LSTM), and compute the similarity between two sentence representations. Hu et al. [5] propose a classification model which makes use of CNN to get sentence representations and computes the sentence matching score by MLP. Mueller et al. [9] devise a regression model based on LSTM and Manhattan metric. Chen et al. [2] present a very effective attention-based enhanced LSTM model.

In this paper, we propose an ensemble method combining the k-fold cross-validation results of various word-level and character-level deep learning models to catch more semantic information for the WeBank task. Our approach achieves the F1 score of 0.85085 which ranks the second in the final evaluation.

2 Model Description

In this section, we describe the proposed deep neural networks to solve the WeBank question matching task. We first model the questions by a certain siamese base network to translate natural language into mathematical representations. Then we treat the task as a binary classification problem and compute the matching score of semantic representations for question pairs with a multi-layer perceptron (MLP). Finally, we find optimal ensemble strategy based on the results of validation set and average the output of different single models.

2.1 Data preprocessing

Word embedding and character embedding are pre-trained by word2vec [8] based on the training data provided by WeBank. They will be used as the initial weight of the embedding layer. We count all the questions in the train set. The max character length and word length (segmented by Jieba[1]) are 123 and 80. We pad or truncate the questions to fixed length based on the max length.

2.2 Semantic Matching Networks

We use siamese architectures due to the input of question pairs. Namely, two networks with the same structure and the same weight each process one question in a pair. We have tried multiple deep siamese models for sentence modeling and classification in the task. The input of them is a sequence of words or characters defined as $x = (x_1, x_2, , x_n)$ (n is the fixed length). The following is a detailed description of these networks.

CNN Siamese Kim [6] applies CNN to natural language in 2014. $\mathbf{V}_{x_i}$ is the k-dimensional vector corresponding to i-th word or character in questions obtained from the embedding layer. Filters with size $h \times k$ are used to produce feature maps in the convolutional layer defined as $\mathbf{z} = \sigma(\mathbf{W} * \mathbf{V}_x + \mathbf{b})$. Here $\mathbf{W}$ is the

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

convolutional weight, $\mathbf{b}$ is the bias, and σ is the activation function (we use $ReLU$ in this task). Then we apply 1-max pooling which means taking the highest value for each feature map to capture the most important feature. All the features generated by filters can be concatenated to represent the input sentences. Then we concatenate two sentence vectors and feed it into two fully-connected layers to obtain the matching score.

BiLSTM Siamese LSTM [4] is an improvement of the recurrent neural network (RNN). LSTM is capable of learning long-term dependencies due to the cell units and gates that can store or forget information. Bidirectional LSTM combines the results of the forward and backward LSTM (we use concatenation). h_i is the hidden state at i-th time step which can represent i-th word and its context. Then we use the last hidden states to represent the input sentences and compute the similarity.

CNN-BiLSTM Siamese The CNN-BiLSTM model contains both CNN and BiLSTM network. CNN can extract n-gram features and LSTM is able to operate over sequence input. Both of these two mainstream models have their own advantages. So we combine the strengths of them. We take advantage of CNN to extract features and use LSTM to encode them to a semantic representation.

R-ESIM Siamese ESIM is a natural language inference (NLI) model, but the main idea is still the sentence modeling and classification. Thus, we can modify and refine the ESIM network to fit the sentence matching task. There are mainly six steps: 1) Input question pairs are encoded to hidden vectors ($a = \{a_1, ..., a_n\}$ and $b = \{b_1, ..., b_n\}$) by BiLSTM; 2) We compute the similarity of each two hidden vectors from given pairs as attention weight ($e_{ij} = {a_i}^T b_j$); 3) New vectors that can reflect local relevance between question pairs are obtained by weighted summation of hidden vectors; 4) The concatenation of the original hidden vectors and new vectors, and the difference and dot product of them are put into another BiLSTM; 5) We use both max and average pooling and concatenate all these vectors to represent the input sentences; 6) We feed the concatenation of sentence semantic vectors into the MLP classifier with sigmoid function in the output layer. The following experiments show that our refined-ESIM achieves great performance in the question matching task.

2.3 Additional Features

In addition to the neural network, there are many features that can reflect the similarity between question pairs in a way. We roughly summarize these useful features into three categories: character level features such as character overlap, Levenshtein distance, and longest common subsequence (LCS), word level features such as word overlap (TF-IDF weighted), word mover's distance (WMD), and sentence level features such as topic models, average of word embedding. We

concatenate all these features including the output of deep models mentioned above and try putting them into a random forest which is a typical ensemble learning method to combine semantic sentence models with additional features.

2.4 Ensemble of Deep Semantic Matching Models

Figure 1 shows the overall framework of our final model. 10-fold cross-validation is an effective approach to improve performance. The original train set is randomly split into 10 subsamples. For each of ten folds, we take one subsample as validation data and the remaining nine subsamples as training data. So we can get ten trained models. The ten outputs can be averaged to produce the final results. This method can avoid the impact of the random data partition.

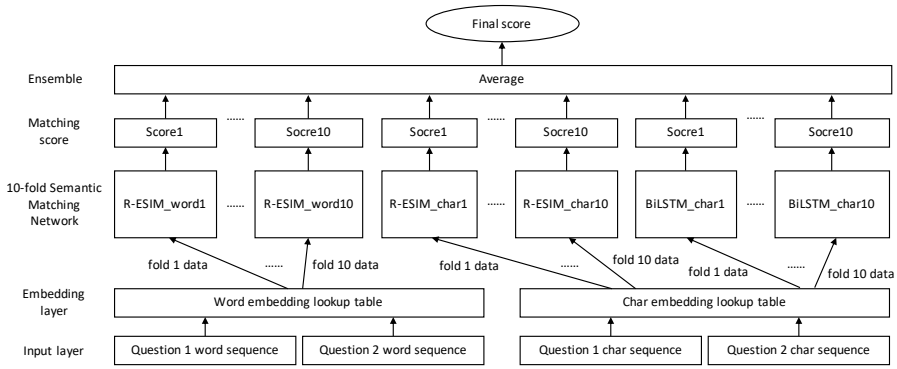

Fig. 1. The overall architecture of our proposed model.

As shown in Figure 1, we respectively train a character-level R-ESIM model, a word-level R-ESIM model, and a character-level BiLSTM model based on 10-fold cross-validation. Thus there are 30 models in total. Word-level and character-level models have the same structure, but the input of them are word and character embeddings. We try to combine these methods to get a more comprehensive model. Voting and averaging are the two simple but effective ensemble methods. In this paper, we average the output of 30 models as the final prediction.

3 Experiments

3.1 Experimental Setups

Our proposed models was implemented in Keras[2] with TensorFlow backend. We used a TITAN X GPU device to train the model. We pre-trained 100-dimensional word embedding and character embedding by gensim as the initial weight of the

[2] https://keras.io

embedding layer which will be updated during the training process. We employed binary cross-entropy as the loss function. The optimizer we utilized was Adam [7] and the batch size was 128.

3.2 The Results of validation data

We evaluated the performance of our models described in Section 2. Table 1 shows the experimental results of these models on the validation set in detail.

Table 1. The results of validation data.

Model	Acc	P	R	F1
CNN word	0.7615	0.7788	0.7304	0.7538
CNN char	0.7682	0.7923	0.7270	0.7580
BiLSTM char	0.8449	0.8622	0.8210	0.8411
CNN-BiLSTM char	0.7886	0.8110	0.7526	0.7807
R-ESIM word	0.8220	0.8517	0.7798	0.8142
R-ESIM char	0.8319	0.8361	0.8256	0.8308
R-ESIM word + char	0.8449	0.8622	0.8210	0.8411
R-ESIM word + char + additional features	0.8517	0.8909	0.8016	0.8439
R-ESIM char (cv)	0.8412	0.8424	0.8394	0.8409
R-ESIM word + char (cv)	0.8533	0.8675	0.8340	0.8504
R-ESIM word + char (cv) + BiLSTM char (cv)	**0.8544**	**0.8696**	**0.8338**	**0.8513**

We can find that character-level models are better than word-level models and 10-fold cross-validation can effectively improve the performance. The ensemble model of character and word level R-ESIM models and character level BiLSTM models based on cross-validation achieves the best result. In other words, our best model is the average of the output of 22 models. We only use the first two folds of the BiLSTM model due to the time limit. Moreover, we incorporate additional features with R-ESIM models, and it can slightly improve the performance. The method is time-consuming and not better than the ensemble method of simple averaging, so we give up them in the final evaluation.

3.3 Final Results

In the final evaluation of test data, we submitted the result of our best model which contains character and word level R-ESIM models (10 folds) and character level BiLSTM models (first 2 folds). Table 2 shows the final results of our submission and other competitors'.

We achieve the F1 score of 0.85085 and rank the second. We do not make any post-processing for the output of our model. The distribution of positive and negative samples is different between the validation set and the test set, but our model is stable and achieves corresponding performance on both datasets which can prove that our model has strong generalization ability.

Table 2. The final results of test data.

Model	Acc	P	R	F1
R-ESIM word + char (cv) + BiLSTM char (cv)	0.8524	0.8599	0.8420	0.8508
ThunderUp	-	-	-	0.8513
DST	-	-	-	0.8483
Laiye-Suda	-	-	-	0.8459
GDUFSER	-	-	-	0.8455

4 Conclusion

In this paper, we describe an ensemble method of various deep neural networks based on 10-fold cross-validation to solve the WeBank question matching task. And the model can be migrated to other similar tasks easily. Furthermore, there are many models that we haven't tried because of time constraint. Further work will focus on combining the 10-fold cross-validation of other deep models and finding a more appropriate method to add additional features.

Acknowledgments This work was supported by the National Basic Research Program of China (2014CB340404), the National Natural Science Foundation of China (71571136), and the Project of Science and Technology Commission of Shanghai Municipality (16JC1403000).

References

1. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent dirichlet allocation. Journal of Machine Learning Research **3**, 993–1022 (2003)
2. 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. pp. 1657–1668 (2017)
3. Deerwester, S.C., Dumais, S.T., Landauer, T.K., Furnas, G.W., Harshman, R.A.: Indexing by latent semantic analysis. Journal of the American Society of Information Science **41**(6), 391–407 (1990)
4. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation **9**(8), 1735–1780 (1997)
5. Hu, B., Lu, Z., Li, H., Chen, Q.: Convolutional neural network architectures for matching natural language sentences. In: Advances in Neural Information Processing Systems. pp. 2042–2050 (2014)
6. Kim, Y.: Convolutional neural networks for sentence classification. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing. pp. 1746–1751 (2014)
7. Kingma, D., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
8. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781 (2013)
9. Mueller, J., Thyagarajan, A.: Siamese recurrent architectures for learning sentence similarity. In: AAAI. pp. 2786–2792 (2016)