FuRongWang at SemEval-2017 Task 3: Deep Neural Networks for Selecting Relevant Answers in Community Question Answering

Sheng Zhang, Jiajun Cheng, Hui Wang,
Xin Zhang, Pei Li and Zhaoyun Ding
College of Information Systems and Management,
National University of Defense Technology,
Changsha, P. R. China
{zhangsheng,jiajun.cheng,huiwang,
zhangxin,peili,zyding}@nudt.edu.cn

Abstract

We describe deep neural networks frameworks in this paper to address the community question answering (cQA) ranking task (SemEval-2017 task 3). Convolutional neural networks and bi-directional long-short term memory networks are applied in our methods to extract semantic information from questions and answers (comments). In addition, in order to take the full advantage of question-comment semantic relevance, we deploy interaction layer and augmented features before calculating the similarity. The results show that our methods have the excellent effectiveness for both subtask A and subtask C.

1 Introduction

Answer selection is regarded as a key step in question answering tasks, especially for community question answering (cQA), which is greatly valuable for user to retrieve information. Some cQA forums, such as Quora, Qatar Living and Stack Overflow, are quite open to users, providing a convenient platform for asking and answering questions. Hence, the development of cQA forums makes it urgent to answer questions automatically.

SemEval-2017 (Bethard et al., 2017) task 3 (Nakov et al., 2017) is the task for selecting relevant answers (or comments) for questions in community question answering (cQA). The data is collected from an online cQA forum, Qatar Living, which is close to some real application needs. The task consists of several subtasks, which are described briefly as follows:

- **Question-Comment Similarity**: Given one question and ten candidates comments $\{c_1, c_2, \cdots, c_n\}$, the goal is to rank these comments according to their relevance to the question.
- **Question-External Comment Similarity**: Given one question and ten candidate questions, which also have ten candidate comments. The questions can be relevant or completely irrelevant. As for the original question, the target is to rank the top ten relevant comments from these 100 candidate comments.

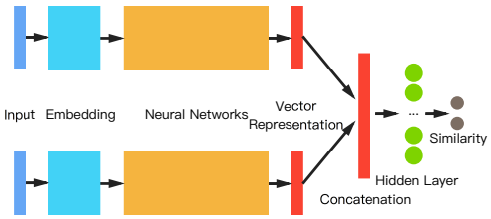

Figure 1: One Basic Deep Learning cQA Framework. The question and the comment are mapped into fixed-length word vectors. After that, they are fed into the neural network framework to extract features. Through a fully connected neural network, the framework outputs the similarity of the question and the comment.

The relevance between the question and the comment can be divided as "Good", "PotentiallyUseful" and "Bad". "PotentiallyUseful" and "Bad" do not make a clear distinction. "Good" comments are considered useful and should be ranked before "PotentiallyUseful" and "Bad" comments. From the above description, these tasks can be regarded as a binary classification tasks: the relation between the question and the comment is divided into "Relevant" or "Irrelevant".

Deep neural networks techniques accelerate the development of automatic question answering,

which due to the great capability of capturing the semantic meaning of texts. Our work is mainly motivated by the previous work (Tan et al., 2016; Feng et al., 2015; Cui et al., 2016; Hu et al., 2014), which utilizes neural networks to extract features from texts. We deploy a deep neural networks framework to measure the relevance of questions and comments in the cQA task, and then rank the candidates according to their similarity to the original question. In this study, methods based on deep neural networks extract semantic features from the question and the comment respectively, which involve little manual operation and show an advantage on the huge amount of data.

In addition, to increase the connection between the question and the comment, we apply an interaction layer before calculating the similarity. Then, we add augmented features to improve the performance of deep neural networks. Finally, the system we proposed are used to address subtask A and subtask C, and the results significantly surpass the baselines provided by the organizers.

2 Methodology

Neural networks are increasingly applied in various of natural language processing tasks, due to the capability of capturing semantic meaning of texts. Our models are fundamentally motivated by previous work in Question Answering.

One basic deep learning architecture for cQA is shown as Figure 1, which is used in a great many papers (Tan et al., 2015; dos Santos et al., 2016; Feng et al., 2015; Cui et al., 2016; Hu et al., 2014). The question and the comment are mapped into fixed-length word vectors. After that, they are fed into the neural network framework to extract features. The frameworks output semantic vectors of the question and the comment, which are then concatenated into a vector. Through a fully connected neural network, the architecture outputs the similarity of the question and the comment. Finally, the similarity value can be the criteria to rank the candidate comments according to the original question.

Currently, convolutional neural networks (CNNs) have proved superiority in a variety of tasks due to the ability of learning the representation of short texts or sentences. Meanwhile, recurrent neural networks (RNNs), especially the variant: long short term memory networks (LSTMs), successfully model the long and short term information of the sequence.

2.1 CNN-based Architecture

The question and the comment are represented by word embedding sequences with a fixed length: $\{w_1, \cdots, w_l\}$, each element is real-valued and $w \in \mathbb{R}^d$. Each sentence is normalized to a fixed length sequence by adding paddings if the sentence is short or truncating the excess otherwise. After embeddings, each sentence can be presented by a matrix $S \in \mathbb{R}^{l \times d}$.

In order to capture higher semantic information of sentences, convolutional layer is applied after embeddings, which consists of several convolutional feature maps. Suppose we have k feature maps $z_i \in \mathbb{R}^{s \times d}$, after convolutional operation, the outputs of CNN is $C \in \mathbb{R}^{(l-s+1) \times k}$.

A pooling layer is added after the convolutional layer. Max pooling and average pooling are commonly used in the model, which choose the max or average value of the features extracted from the former layer to reduce the presentation. In this study, we use 1-max pooling as our methods to select the max value of each filter, and the question and the comment vectors generated by neural networks are $q, c \in \mathbb{R}^k$ respectively.

In order to extract features from different scales, we use different types of feature maps altogether. These feature maps have different width to capture information from different contexts, which contribute to the feature extraction of CNNs.

2.2 LSTM-based Architecture

The long short term memory (LSTM) network is a variant of recurrent neural network (RNN) which has recently received great results on various of sequence modeling tasks. LSTM overcomes the shortcoming of RNN in handling "long-term dependencies". Memory cells and forget gates are the key points of the LSTM: they make it capable of handling both long and short sequences through controlling the information flow of a sequence. A LSTM network is made up of several cells with the following formula:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f), \quad (1)$$
$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i), \quad (2)$$
$$\tilde{C}_t = tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \quad (3)$$
$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t, \quad (4)$$
$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o), \quad (5)$$
$$h_t = o_t \odot tanh(C_t), \quad (6)$$

where W and b are parameters which are trained and shared by all cells, and $\odot$ indicates element-wise production. C_t is the memory cell, which stores previous values. The forget gate f_t and the input gate i_t control the percentage of information from the previous memory and new inputs respectively, while the output gate o_t controls the output of hidden states. $\sigma(\cdot)$ indicates the sigmoid function.

Single directional LSTM only utilizes the former information of a sequence, while bi-directional LSTMs utilize the information both forward and afterward. Usually, in order to capture more information of the sequence, bi-directional LSTMs are adopted, with the one processes information from the front to the end of a sequence while the other processes information in the reverse direction. The output of bi-directional LSTMs can be the concatenation of two output vectors from both direction, i.e. $h_t = \overrightarrow{h_t} || \overleftarrow{h_t}$.

2.3 Augmented Features

In general, neural networks are able to extract features automatically. However, (Fu et al., 2016) and (Yu et al., 2014) have shown that augmented features contribute to the behavior of neural networks. Some commonly used augmented features such as word overlap indices, part-of-speech tags, position indices, etc.

In this work, we use word overlap indices as augmented features. Given a question $q_i = (x_1^q, x_2^q, \cdots, x_m^q)$ and a comment $c_i = (x_1^c, x_2^c, \cdots, x_m^c)$, the overlap features q_{feat} and c_{feat} are calculated as follows:

$$q_{feat}^{(i)} = \begin{cases} 1 & x_i^q \in c_i, \\ 0 & \text{otherwise}, \end{cases} \quad (7)$$

$$c_{feat}^{(i)} = \begin{cases} 1 & x_i^c \in q_i, \\ 0 & \text{otherwise}, \end{cases} \quad (8)$$

where $q_{feat}^{(i)}$ is the i^{th} element of q_{feat}, so is $c_{feat}^{(i)}$.

As shown in Figure 2, q_{feat} and c_{feat} are added at the tail of sequence embeddings. Also, their concatenation x_{feat} is regarded as augmented features before feeding into the fully connected networks.

2.4 Interaction Layer

In order to make full use of the connection of the question and the comment, we design an interaction layer to capture the relevance of them, which is shown in Figure 2.

Given a question vector $q \in \mathbb{R}^k$ and a comment vector $c \in \mathbb{R}^k$ produced by neural networks, the interaction layer calculates the matrix multiplication as follows:

$$z_{int} = f(q^T W c), \quad (9)$$

where $z_{int} \in \mathbb{R}$ is the output of interaction layer, and $M \in \mathbb{R}^{k \times k}$ is the parameter of the layer and updated when training, while $f(\cdot)$ is the activation function.

2.5 Objective Function and Optimizer

Features extracted by deep neural networks are concatenated with extra features, which are then fed into a fully connected neural network altogether.

$$o_i^h = f(W_o[q, c, x_{feat}, z_{int}] + b_o), \quad (10)$$

where o_i^h is the output of hidden layer node i, and x_{feat} is the augmented features. W_o and b_o are the weight and the bias of the hidden layer respectively.

After that, the softmax function is applied to obtain the similarity of the question and the comment:

$$o_i = Softmax(W_s \cdot o_i^h + b_s), \quad (11)$$

where $o_i \in [0, 1]$ is the output of the network, which satisfies $\sum_i o_i = 1$. W_s and b_s are the weight and the bias of the output layer respectively.

The objective function in this study is cross entropy, which is illustrated as follows:

$$\mathcal{L} = -\frac{1}{N} \sum_{i \in N} \{y_i \log o_i + (1 - y_i) \log (1 - o_i)\}, \quad (12)$$

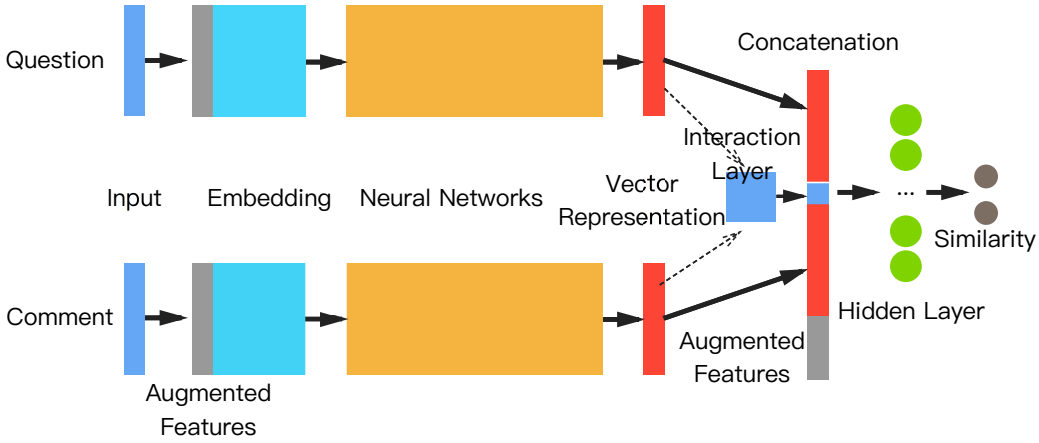

Figure 2: The deep neural network frameworks for cQA. The question and the answer are represented by word vectors, with augmented features added at the end. Then the vectors are fed into neural networks, such as CNN or LSTM, to extract vectors. Interaction layer compute the similarity of vectors, which are then concatenated with extracted vectors together. Through a fully connected neural network, the framework outputs the similarity of the question and the comment.

where $y_i \in \{0, 1\}$ is the ground truth label of the question-comment relation. N is the number of samples.

The relation of the question and the comment is divided to two classes, "Good" and "Bad" ("PotentiallyUseful" is regraded as "Bad"). To train the parameters in networks, Adagrad optimizer (Duchi et al., 2011) is applied. Adagrad is an algorithm designed for gradient-based optimization, which adapt the learning rate for better convergence when training the parameters. (Dean et al., 2012) point out that Adagrad increase the robustness of stochastic gradient descent when training large-scale neural networks.

3 Experimental Results

In this section, we describe the detail of training deep neural networks, including the datasets, metrics, baseline, parameters settings and so on. Then, we present the results and give a brief analysis of these results.

3.1 Data Description

Datasets provided by the organizers are collected from Qatar Living forum, which is an online cQA website. The components of the datasets are shown as Table 3.1:

In our experiment, datasets "train_1", "train_2" and "dev" are used to train neural networks. In addition, dataset "test_2016" is used for development and parameters searching, and dataset "test_2017" is submitted for evaluation.

Dataset	train_1	train_2	dev
Questions	1999	670	500
Dataset	test_2016	test_2017	
Questions	700	880	

Table 1: The components of the datasets.

3.2 Metrics and Baselines

The official evaluation metrics in this task is Mean Average Precision (MAP), which is used for ranking submissions from different teams. MAP is often used to measure the quality of ranking in information retrieval. In addition, Average Recall (AvgRec), Mean Reciprocal Rank (MRR), Accuracy (Acc), etc. are also reported by the official scorer.

Baselines are given by the organizer, which consists of Information Retrieval (IR) baseline and random baseline. IR baseline is the rank of candidates given by a search engine, such as Google or Bing. Random baseline is the results of given a random number (from 0 to 1) to rank each candidate.

3.3 Experimental Setup

Our models in this work are mainly implemented with tensorflow v1.0 (Abadi et al., 2016) from a scratch. The code runs on a GTX 1080 GPU and

Datasets	test_2016			test_2017		
Methods	MAP	AvgRec	MRR	MAP	AvgRec	MRR
Random (baseline)	52.80	66.52	58.71	62.30	70.56	68.74
IR (baseline)	59.53	72.60	67.83	72.61	79.32	82.37
CNN	73.13	84.25	79.74	82.93	89.94	88.28
multi-CNN	74.38	85.24	**81.58**	83.42	90.13	88.96
+Augmented Features	73.86	84.99	80.81	83.52	90.48	89.60
+Interaction Layer (Pri)	**74.94**	**85.85**	81.10	**84.26**	**90.79**	89.40
biLSTM	74.02	84.97	80.79	83.57	90.28	89.42
+Augmented Features	73.57	84.66	79.94	84.06	90.56	**89.74**
+Interaction Layer	74.72	85.59	81.22	83.96	90.70	89.49

Table 2: Results on Subtask A

Datasets	test_2016			test_2017		
Methods	MAP	AvgRec	MRR	MAP	AvgRec	MRR
Random (baseline)	15.01	11.44	15.19	5.77	7.69	5.70
IR (baseline)	40.36	45.97	45.83	9.18	21.72	10.11
CNN	47.62	50.25	52.64	12.57	29.00	14.06
multi-CNN	48.85	51.56	55.11	12.55	**29.65**	**14.64**
+Augmented Features	49.81	52.04	55.68	**13.55**	29.45	**14.63**
+Interaction Layer (Pri)	**50.15**	**53.58**	54.56	13.23	29.51	14.27
biLSTM	46.31	51.64	55.39	12.78	26.48	14.25
+Augmented Features	47.29	51.72	**55.91**	12.73	26.48	14.24
+Interaction Layer	48.06	52.72	53.70	13.23	26.13	14.52

Table 3: Results on Subtask C

it is about 100 epochs that the model become to converge.

All texts from questions and comments are used at first to train Word2Vec vectors (Mikolov et al., 2013a,b) by a Python package gensim [1], whose length is fixed to 100. The max sequence length of the question and the comment are fixed to 200. We add paddings if the sequence is short or truncate the excess otherwise.

The single-type CNN networks have the filter size of 3, and 800 feature maps, while the multi-type CNN networks have the filter sizes of 1,2,3 and 5 with 800 feature maps each. The bi-LSTMs have the output length of 400 of each direction, and the hidden states are outputted directly for the higher layer. The number of nodes in hidden layer is 256 and the activation function used in fully connected neural networks is ReLu. The optimizer is set to AdagradOptimizer and the learning rate is set to 0.01 initially.

[1] https://radimrehurek.com/gensim/

3.4 Results on subtask A

Table 3.1 summarizes the results on subtask A: Question-Comment Similarity. The first two rows illustrate the random baseline and the IR baseline, follow by 4 rows of CNN results. The last three rows are the results of LSTMs. As shown in the table, our results significantly outperform the baselines. Neural network based methods perform quite stable, the differences between their results are less than 1%.

The multiCNN along with augmented features and interaction layer achieves the best MAP scores among these methods, although it does not rank the first as for MRR scores. It is also clear that interaction layer and augmented features contribute to the behavior of neural networks.

3.5 Results on subtask C

Table 3.1 illustrates the results on subtask C: Question-External Comment Similarity. It should be noted that we use the reciprocal rank of questions to improve the rank of the comments.

From the table, our methods surpass the baselines and the best method obtains the MAP of

13.55. The circumstance is quite similar to subtask A, which has proved that our deep neural based architectures are of great stability. However, LSTM-based architectures is far behind CNN-based architectures in terms of MAP and AvgRec scores, but have the advantage in terms of MRR scores.

4 Conclusion

In this paper, we present deep neural networks frameworks to address community question answering tasks. CNNs and biLSTMs are used to extract the semantic features of questions and comments. We add an interaction layer and augmented features to improve the performance of the framework. The results illustrate that our methods are greatly superior to the baselines provided by organizers both in subtask A and subtask C.

Acknowledgments

The research is supported by National Natural Science Foundation of China (No.71331008) and (No.61105124).

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. 2016. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467* .

Steven Bethard, Marine Carpuat, Marianna Apidianaki, Saif M. Mohammad, Daniel Cer, and David Jurgens, editors. 2017. *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*. Association for Computational Linguistics, Vancouver, Canada. http://www.aclweb.org/anthology/S17-2.

Yiming Cui, Zhipeng Chen, Si Wei, Shijin Wang, Ting Liu, and Guoping Hu. 2016. Attention-over-attention neural networks for reading comprehension. *arXiv preprint arXiv:1607.04423* .

Jeffrey Dean, Greg Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Mark Mao, Andrew Senior, Paul Tucker, Ke Yang, Quoc V Le, et al. 2012. Large scale distributed deep networks. In *Advances in neural information processing systems*. pages 1223–1231.

Cıcero Nogueira dos Santos, Ming Tan, Bing Xiang, and Bowen Zhou. 2016. Attentive pooling networks. *CoRR, abs/1602.03609* .

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research* 12(Jul):2121–2159.

Minwei Feng, Bing Xiang, Michael R Glass, Lidan Wang, and Bowen Zhou. 2015. Applying deep learning to answer selection: A study and an open task. In *Automatic Speech Recognition and Understanding (ASRU), 2015 IEEE Workshop on*. IEEE, pages 813–820.

Jian Fu, Xipeng Qiu, and Xuanjing Huang. 2016. Convolutional deep neural networks for document-based question answering. In *International Conference on Computer Processing of Oriental Languages*. Springer, pages 790–797.

Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional neural network architectures for matching natural language sentences. In *Advances in neural information processing systems*. pages 2042–2050.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* .

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. pages 3111–3119.

Preslav Nakov, Doris Hoogeveen, Lluís Màrquez, Alessandro Moschitti, Hamdy Mubarak, Timothy Baldwin, and Karin Verspoor. 2017. Semeval-2017 task 3: Community question answering. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*. Association for Computational Linguistics, Vancouver, Canada, pages 27–48. http://www.aclweb.org/anthology/S17-2003.

Ming Tan, Cicero dos Santos, Bing Xiang, and Bowen Zhou. 2015. Lstm-based deep learning models for non-factoid answer selection. *arXiv preprint arXiv:1511.04108* .

Ming Tan, Cicero Dos Santos, Bing Xiang, and Bowen Zhou. 2016. Improved representation learning for question answer matching. In *Meeting of the Association for Computational Linguistics*. pages 464–473.

Lei Yu, Karl Moritz Hermann, Phil Blunsom, and Stephen Pulman. 2014. Deep learning for answer sentence selection. *arXiv preprint arXiv:1412.1632* .