Attentive Interactive Neural Networks for Answer Selection in Community Question Answering

Xiaodong Zhang[1], **Sujian Li**[1,2], **Lei Sha**[1], **Houfeng Wang**[1,2]
[1] MOE Key Lab of Computational Linguistics, Peking University, Beijing, 100871, China
[2] Collaborative Innovation Center for Language Ability, Xuzhou, Jiangsu, 221009, China
{zxdcs, lisujian, shalei, wanghf}@pku.edu.cn

Abstract

Answer selection plays a key role in community question answering (CQA). Previous research on answer selection usually ignores the problems of redundancy and noise prevalent in CQA. In this paper, we propose to treat different text segments differently and design a novel attentive interactive neural network (AI-NN) to focus on those text segments useful to answer selection. The representations of question and answer are first learned by convolutional neural networks (CNNs) or other neural network architectures. Then AI-NN learns interactions of each paired segments of two texts. Row-wise and column-wise pooling are used afterwards to collect the interactions. We adopt attention mechanism to measure the importance of each segment and combine the interactions to obtain fixed-length representations for question and answer. Experimental results on CQA dataset in SemEval-2016 demonstrate that AI-NN outperforms state-of-the-art method.

Introduction

With the rapid development of Internet, community question answering (CQA) forums, e.g. Quora[1] and Stack Overflow[2], have accumulated a large quantity of questions and corresponding answers. These forums are quite open, and thus they typically have little restrictions, if any, on who can post and who can answer a question. As a consequence, it takes effort to go through all possible answers and to make sense of them because of the uneven quality of answers (Nakov et al. 2016). Given a question, answer selection aims to pick out good answers from a set of candidates. Answer selection in CQA can contribute to various aspects. For example, it is beneficial to list good answers ahead of bad ones in CQA websites, which can save users' time. Besides, it is a crucial step for automatic question answering based on similar question matching.

Table 1 lists an example question and its two possible answers. A question usually includes two parts, that is a title, which gives a brief summary of the question, and a body, which describes the question in detail. Answer 1 is a good answer, because it provides helpful information, such as "lady siam massage". Although Answer 2 is a reply to the question, it does not contain any useful information so that it is regarded as a bad answer.

Question title	Best place for massage
Question body	Tell me; where is the best place to go for a massage? Mind you; I don't want to spend 1000QR for it... (Guys; please don't come up with answers that you would gladly do it yourself; plz...)
Answer 1	"have you try lady siam massage next to toy ""r"" us al sadd? for one hour = 120riyal. not bad i must say Everybody is right Everybody is wrong; it depend where we stand."
Answer 2	I have never been for a massage! Can you believe it? Maybe I should go and try too! :)

Table 1: An example question and answers in CQA.

Unlike answer selection in other areas, e.g. WIKIQA (Yang, Yih, and Meek 2015), an obvious characteristic can be seen from Table 1, that is redundant and noisy. Questions and answers in CQA generally contains many sentences. Some sentences are auxiliary and do not provide meaningful information, like the bracketed sentences in the question body. Furthermore, the text is non-standard. There are quantities of informal language usages, such as abbreviations, typos, emoticons and grammatical mistakes.

In recent years, many researchers have proposed various deep learning methods that automatically select answers. These methods usually learn representations of two texts using neural networks, e.g. convolutional neural networks (CNNs) (Qiu and Huang 2015) or recurrent neural networks (RNNs) (Wang and Nyberg 2015). Based on the representations, a function is utilized to give the matching score of two texts. Instead of learning global representation of whole text, some researchers have proposed models that learn the interaction information of the representations and achieved better results (Hu et al. 2014). Although interaction is employed, they are treated equally, thus this method is not suitable for redundant and noisy text in CQA.

Two main characteristics of our model, as implied by the

[1] https://www.quora.com/

[2] http://www.stackoverflow.com/

name, is attentive and interactive. With interaction, it can model the relation between each segments of question and answer, thus digging out more information for answer selection. With attention, it can focus on useful segments of text and neglect meaningless segments, thus suitable for redundant and noisy text in CQA. In AI-NN, representations of question and answer are first learned by some neural networks such as CNNs and RNNs. Then it calculates the interaction (or called matching patterns) between each pair of representations. Inspired by (Santos et al. 2016), row-wise and column-wise max-pooling is used to summarizes the interactions. Different from their work, we use a 3D tensor to depict interactions. This way has two advantages. Firstly, the summarized interaction contains influence of a segment to the other text, which is key information to guide attention. Secondly, the summarized interaction is a vector rather than a real number as in (Santos et al. 2016), alleviating loss of information. We adopt a network to calculate attention of each segment and a variety of information is considered, including representation of segment, interaction with the other text, question topic, and question type. Information used by us is more diverse than prior work (Santos et al. 2016; Yin et al. 2015), which can bring more accurate attentions. Combining the interactions with the attention weights, global representations of two texts are obtained, which are finally used for calculating matching score or making classification. By attention calculation, the redundant and noisy segments are endowed with less importance, and the representation puts emphasis on useful segments, thus our model is suitable for processing text in CQA. Although we focus on answer selection task in this paper, our model is rather generic and can be used for other tasks that matching two texts or predicting their relations.

Model

An overview of our model is illustrated in Figure 1. The inputs are two texts, i.e. a question $Q = \{w_t^q | t \in [1, T^q]\}$ and an answer $C = \{w_t^c | t \in [1, T^c]\}$, where w_t^q and w_t^c are the t-th words in question and answer respectively, and T^q and T^c are their lengths. The left and top parts of Figure 1 are text representations based on neural networks. The middle part is interactions. The bottom and right parts are attentions. The lower right is output, which is a predicted matching score or class distribution. In the following subsections, our model is depicted in detail.

Text Representation

Different from (Hu et al. 2014), who use convolution to directly learn interaction, we split the process of representation learning and interaction learning. It brings additional flexibility, because any kind of neural networks can be utilized to learn the representation, e.g. CNNs and RNNs. Here we give an example of learning representations of question through one-layer convolution. Specifically, AI-NN with representations learned by CNNs is referred to as AI-CNN. It is also easy to define AI-RNN, AI-LSTM, and so on.

For CNNs, a segment of text can be regarded as words in a sliding window. For question $Q = \{w_t^q | t \in [1, T^q]\}$,

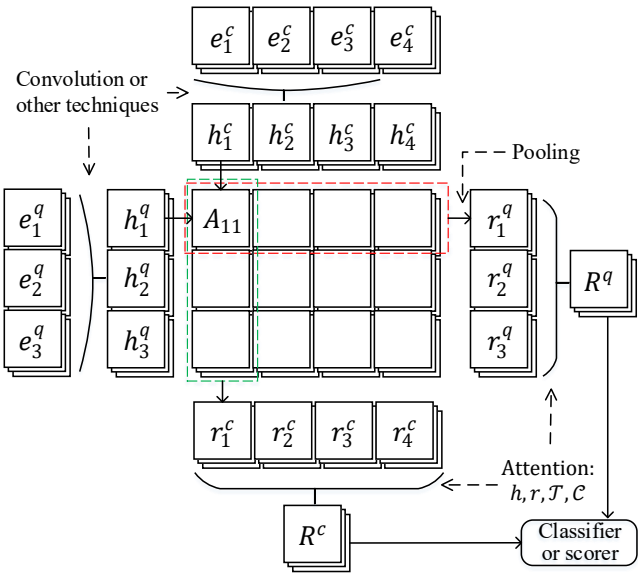

Figure 1: The architecture of AI-NN. A square represents for a real number, a set of stacked squares for a vector, a row or column of stacked squares for a matrix, and the middle part of the figure is a 3D tensor.

the word embeddings are $E^q = \{e_t^q | t \in [1, T^q]\}$, where e_t^q is the embedding of word w_t^q. The t-th input of convolution layer x_t is the concatenation of the t-th word embedding and its surrounding word embeddings in a window. Formally,

$$x_t = \left[e^q_{t-\lfloor d/2 \rfloor}, ..., e^q_t, ..., e^q_{t+\lceil d/2 \rceil - 1} \right] \quad (1)$$

where d is the size of window, $\lfloor n \rfloor$ and $\lceil n \rceil$ represent the floor and ceiling of n respectively. The representations of question, i.e. the hidden states of convolution layer, are obtained by convolution of the input window, adding a bias term and then applying a non-linear function. Formally,

$$h_i^q = \sigma(W^h * x_i + b^h) \quad (2)$$

where representation h_i^q of i-th segment is a k-dimensional vector, k is the number of feature map, σ is an activation function, e.g. Tanh or Relu, W^h is transformation matrix, and b^h is bias vector.

Interaction

After obtaining the representation $H^q = \{h_i^q | i \in [1, T^q]\}$ of question Q and the representation $H^c = \{h_j^c | j \in [1, T^c]\}$ of answer C, we compute the interactions of each paired segments in Q and C. Specifically, for the i-th hidden state h_i^q in H^q and the j-th hidden state h_j^c in H^c, their interaction A_{ij} is calculated by

$$A_{ij} = \sigma(W^a * [h_i^q, h_j^c] + b^a) \quad (3)$$

where $[u, v]$ represents the concatenation of u and v, W^a and b^a are transformation matrix and bias vector respectively.

The interaction A is a 3D tensor. Following (Santos et al. 2016), we use row-wise pooling to obtain a matrix that

summarizes the interaction of each segment in question with all segments in answer.

$$r_i^q = \max_{m \in [1, T^c]} A_{im} \tag{4}$$

Similarly, column-wise pooling is used to summarize the interaction of each segment in answer.

$$r_j^c = \max_{n \in [1, T^q]} A_{nj} \tag{5}$$

Through row-wise and column-wise pooling, the summarized interaction contains influence of a segment to the other text. We extent 2D matrix used in (Santos et al. 2016) to 3D tensor, in which each interaction is a vector, thereby summarizing more abundant information. The summarized interaction has two uses. It forms the final representation and serves as a factor to compute attention.

Attention Calculation

In the redundant and noisy text, not all segments are helpful. It is necessary to discard the dross and select the essence. We borrow the attention mechanism and design a neural network to compute the importance of each segment and is trained jointly with the whole model. Four kinds of information are used for calculating the attention, including segment representation, interaction with other text, question topic, and question type. Formally, the attention α_i^q of each segment in a question is calculated by

$$\alpha_i^q = \frac{\exp(u_i^q)}{\sum_{k=1}^{T^q} \exp(u_k^q)} \tag{6}$$

$$u_i^q = a(h_i^q, r_i^q, \mathcal{T}, \mathcal{C}) \tag{7}$$

where u_i^q is unnormalized attention, a is a feedforward neural network, $\mathcal{T}$ is embedding of question topic, and $\mathcal{C}$ is vector of question type. In a CQA forum, generally each question belongs to a certain topic category chosen by a user, like "cars and driving", "computers and Internet" and so on. We associate each topic with an embedding, which is initialized randomly and tuned in training process. In addition to question topic, question type is also considered. All questions are classified into six types according to interrogatives, namely what, where, when, why, who (whose, whom) and how. Interrogatives have influence on the key points in an answer. For example, if a question asks "where", segments that involve place and location should be focused. A question in CQA usually contains several sentences. The occurrence number of the interrogatives in all sentences is counted to get a six dimensional vector, in which each dimension represents a kind of interrogative. Then the vector is normalized and is used as the vector of question type.

Finally, a fix-length representation of question is computed as a weighted sum of the variable-length interactions of all segments.

$$R^q = \sum_{l=1}^{T^q} \alpha_l^q r_l^q \tag{8}$$

The attention α^c and fix-length representation R^c of answer are computed in the same way as question.

These four kinds of information depict different aspects and guide attention calculation together. Consequently, the redundant and noisy segments are endowed with less importance, and the representation puts emphasis on useful segments. This is the key reason that AI-NN can effectively model text in CQA.

Additional features and Scorer

Generally, questions in a CQA forum contain some meta information (e.g., date, user, etc.), which provides additional features for answer selection. By analyzing the CQA dataset in SemEval-2016, we design four additional features from the meta information:

a) Whether answer and question are from the same user. Some questioner may communicate with answerer to provide more information or ask more questions. Consequently, the questioners can answer their own question and these answers are probably meaningless.

b) Whether the answerer is anonymous. Compared with user with username, anonymous user may provide answers of worse quality.

c) The order of an answer. We find that the quality of front answers is better than the quality of rear answers. This feature is set to $o/10$, where o is the order of an answer.

d) The length of an answer. Long answers are likely to have higher quality than short answers. If the length l of an answer is less than 150, this feature is set to $l/150$, otherwise it is set to 1.

Consequently, the additional features are represented by a four dimensional vector. The first two dimensions are 0 or 1, and the latter two are real numbers. A quantitative analysis of these features is given in experimental section, which can explain why these features are chosen. It should be noted that although additional features are used in our model, they are extracted from meta information. We do not use any text matching feature as used in previous work, like BM25, word overlap, and so on.

A non-linear transformation is applied to the representation learned by the neural networks and then it was concatenated with the additional features to get final representation.

$$R = [R^T, R^F] \tag{9}$$

$$R^T = \sigma(W^T * [R^q, R^c] + b^T) \tag{10}$$

where W^T and b^T are transformation matrix and bias vector respectively, and R^F is the vector of additional features.

The last part of AI-NN is a classifier or a scorer. The simplest way to train the model is to use a Softmax classifier with cross entropy as the loss function.

$$p(Q, C) = softmax(W * R + b) \tag{11}$$

An alternative way is a pair-wise learning strategy with a large margin loss. Triples (Q, C^+, C^-) are constructed from training data, with Q matched with C^+ better than with C^-. A ranking-based loss is defined as

$$\mathcal{L} = \max(0, m + s(Q, C^-) - s(Q, C^+)) \tag{12}$$

$$s(Q, C) = \sigma(W * R) \tag{13}$$

where $s(Q, C)$ is predicted matching score for question Q and answer C, and m is the margin.

Experiments

Dataset

We evaluate our proposed method on SemEval-2016 Task 3: *Community Question Answering* (Nakov et al. 2016). This dataset contains real data from the community-created Qatar Living Forums[3]. There are three subtasks and we focus on Subtask A: *Question-Comment Similarity*. Here a comment is equal to an answer. Given a question from a question-comment thread, the participant systems rank the comments according to their relevance (similarity) with respect to the question. Each comment is labeled with one of three labels, namely "Good", "PotentiallyUseful", and "Bad". The latter two are not distinguished and are considered "Bad" in terms of evaluation. Table 2 demonstrates statistics of the dataset. The major difference between the dataset and other answer selection dataset is the length of question. In WIKIQA (Yang, Yih, and Meek 2015) and TREC-QA (Wang, Smith, and Mitamura 2007), a question is a short sentence, while in the CQA dataset, the question body contains quite a few sentences.

	Train	**Dev**	**Test**
# of ques.	4879	244	327
# of ans.	36198	2440	3270
Avg. len. of ques. (title / body)	6.40 / 43.29	6.04 / 46.88	6.17 / 49.77
Avg. len. of ans.	37.76	36.18	37.27

Table 2: Statistics of the CQA dataset.

Baselines

We compare our model with five baseline methods. Kelp and ConvKN were participant teams in SemEval-2016 CQA task and got the first and second place respectively.

- ARC-I (Hu et al. 2014): It first finds the representation of each sentence, and then compares the representation for the two sentences with a multi-layer perceptron (MLP).
- ARC-II (Hu et al. 2014): It models all the possible combinations of sliding windows on both sentences, which consider interactions between two sentences.
- AP (Santos et al. 2016): It jointly learns a similarity measure over projected segments of the paired text, and derives the corresponding attention vector for each input to guide the pooling.
- Kelp (Filice et al. 2016): It uses SVM learning algorithm that operates on a linear combination of kernel functions. Three kinds of feature vectors are used: linguistic similarities between texts, shallow syntactic trees, and task-specific information.
- ConvKN (Barrón-Cedeño et al. 2016): It combines convolutional tree kernels with convolutional neural networks and manually designed features including text similarity and thread specific features.

Experimental Settings

The questions and answers are tokenized and lemmatized using NLTK[4]. We do not remove any stop word. The title and body of a question are concatenated into a single text string. We do not set a max length for text as we find it has a negtive impact. GloVe (Pennington, Socher, and Manning 2014) is utilized to train word embeddings, which is used as initializations of embeddings in neural networks. The training corpus is the unannotated data in the SemEval-2016 CQA dataset. The dimension of word embedding is 200. Tokens that did not appear in the pre-trained word embeddings are replaced with a special token, of which the embedding is initialized randomly. The window size and number of feature map of CNN is 3 and 200. The dimension of embeddings of question topic is 20. We find that results of the two training strategies are close, so Softmax classifier is adopted in the following experiments for convenience. Adagrad (Duchi, Hazan, and Singer 2011) is used to update parameters. Initial learning rate is 0.002, and batch size is 1.

Results and Analysis

Three metrics are used for evaluation, including a ranking metric, Mean Average Precision (MAP), and two classification metrics, Accuracy and F1. The scores are calculated using the official evaluation script[5]. Table 3 presents the results of our model and the baseline methods.

Method	**MAP**	**Acc**	**F1**
ARC-I	77.05	74.07	69.50
ARC-II	77.98	75.26	71.64
AP	77.12	75.47	71.72
Kelp	79.19	75.11	64.36
ConvKN	77.66	75.54	66.16
AI-CNN (w/o features)	79.17	76.30	72.75
AI-CNN	80.14	76.87	73.03

Table 3: Comparison of our model with baselines.

We can see that ARC-II outperforms ARC-I, demonstrating that interactions between questions and answers are helpful. Although using a two-way attention mechanism, AP does not outperform ARC-II. This may be because AP uses dot product with a transformation to calculate attentions. In redundant and noisy texts, if both texts contain some same or similar but meaningless segments, they are inevitable to be brought into the final representation, consequently restricting the improvement of performance. Kelp used various kinds of features and got the best result in SemEval-2016, which demonstrates that feature engineering is effec-

[3] http://www.qatarliving.com/forum

[4] http://www.nltk.org/

[5] http://alt.qcri.org/semeval2016/task3/index.php?id=data-and-tools

tive in answer selection task. ConvKN used tree kernel and CNNs to measure the similarity of question and answer. The two scores are combined to make a final prediction. The CNNs used in ConvKN is similar to ARC-I. ConvKN outperforms ARC-I, showing that the syntactic information is useful. The first three baselines do not use manually designed features. To make a fair comparison, we report the result of AI-CNN without additional features. It outperforms ARC-II and AP, showing that our model can effectively select useful segments and measure the relation of question and answer accurately. More surprisingly, without any manual features, it even outperforms Kelp, which has heavy feature engineering. With four additional features, the result of AI-CNN improves further and outperforms other baselines on all metrics.

Analysis of Attention

The results of using each kind of information to calculate attentions are listed in Table 4. In these experiments, additional features are not used.

Information	MAP	Acc	F1
w/o attention	78.05	75.19	71.50
+ representation	78.83	75.95	72.16
+ interaction	78.75	75.92	72.43
+ question topic	77.91	75.25	71.93
+ question type	78.22	75.22	72.10
+ all	79.17	76.30	72.75

Table 4: Influence of different information to attention calculation.

In baseline method, we do not use attention but pooling all interactions to form the final representation, i.e. pick out a maximum value from the first and second dimensions (can be viewed as a matrix) of the 3D tensor. When using a single information to calculate attention, the representation of segment and the interaction with the other text outperform the baseline by a large margin. The question topic information does not perform well, even decreasing the MAP. The question type information outperforms the baseline a little. The way that the question type information vector is calculated is simple. With more sophisticated method, we think the performance can improve further, which will be explored in the future. Nevertheless, with all the information, the attentive model outperforms the baseline on all three metrics, demonstrating the effectiveness of the attention.

To give a visual display, two heat maps are demonstrated in Figure 2, in which (a) and (b) depict attentions in AP and AI-CNN respectively. Although AP gives different level of attention to different segments, the differentiation is not significant. In AI-CNN, useful segments are endowed with higher level of attention and useless segments with lower level. The variance is more significant. With various kinds of information to calculate attention, our model is effective at selecting helpful segments in the redundant and noisy text.

We use two separate parameters to calculate attention for question and answer. This is because question and answer may have different segments to attend to. An interesting thing can be found in Figure 2. The question mark in the question is endowed with low level of attention, while it is endowed with high level of attention in the answer. This is because it is common for a question mark to appear in a question, which means it is meaningless. However, a question mark in an answer indicate the answerer ask in reply for more information or give some advices, which is a clue for good answer.

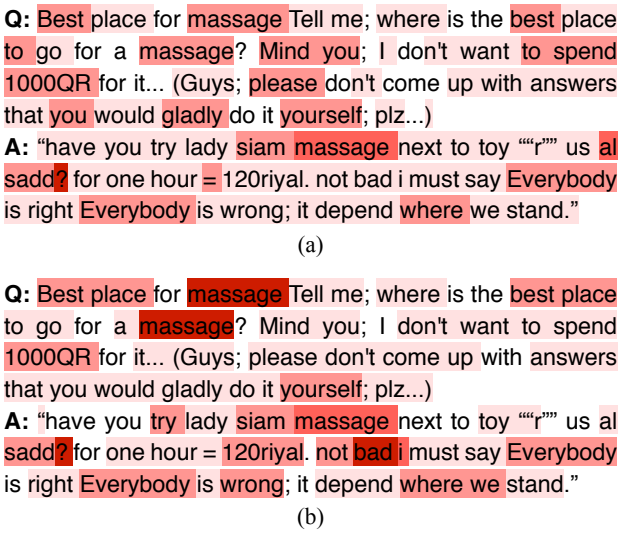

Figure 2: Comparison of attentions in AP and AI-CNN.

Analysis of Additional Features

For each of the additional features, we collect statistics from training set of the CQA dataset. As the official evaluation, “PotentiallyUseful” is regarded as “Bad”, so there are only “Good” and “Bad” answers in the dataset. For answers posted by the same user of the question, only 11% of them are good answers. The proportion of good answers in all answers posted by anonymous users is 35.8%, while the proportion is 44.3% in non-anonymous users. Figure 3(a) demonstrates the influence of the order of answers. The later an answer is posted, the worse the quality is. Figure 3(b) demonstrates the influence of the length of answers. With the increase of the length, the percent of good answers increases, and becomes steady when length is larger than 130.

The results of using each feature and all features are listed in Table 5. Basically, Feature a, c and d can improve at least two metrics, demonstrating the effectiveness of these features. Feature b, however, does not perform well, only improving the accuracy and decreasing MAP and F1. With all four features, the result beats not using any feature and using one feature.

Related Work

From the emergence of answer selection task until now, researchers have been racking their brains to design all kinds of features to solve the problem. Surdeanu, Ciaramita, and Zaragoza (2011) investigated a wide range of feature types

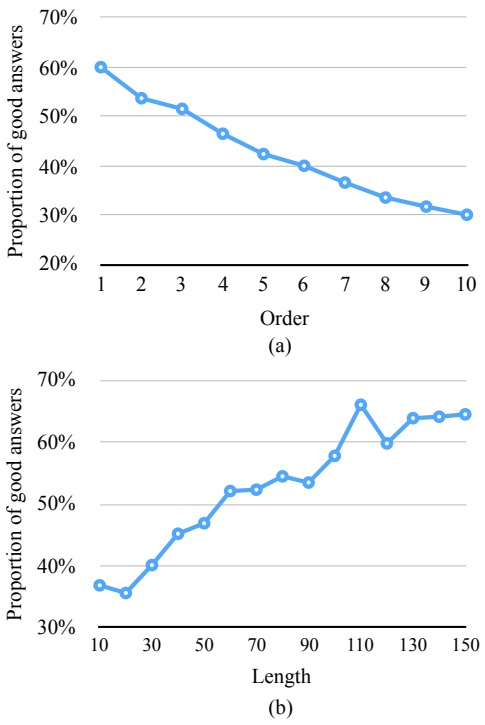

Figure 3: The influence of order and length of answer.

Feature	MAP	Acc	F1
w/o feature	79.17	76.30	72.75
+ a) same user	79.62	76.50	72.46
+ b) anonymous	78.87	76.41	72.11
+ c) order	79.54	76.62	72.91
+ d) length	79.32	76.22	72.83
+ all	80.14	76.87	73.03

Table 5: Influence of different features.

such as similarity features, translation features, density / frequency features for ranking answers to non-factoid questions in Yahoo! Answers. Xue, Jeon, and Croft (2008) proposed a retrieval model that combined a translation-based language model for the question part with a query likelihood approach for the answer part. Yih et al. (2013) formulated answer sentence selection as a semantic matching problem with a latent word-alignment structure and conducted a series of experimental studies on leveraging proposed lexical semantic models. Tymoshenko and Moschitti (2015) studied the use of syntactic and semantic structures obtained with shallow and deeper syntactic parsers in the answer passage re-ranking task. Barrón-Cedeño et al. (2015) designed specific features looking globally at a QA thread and applied structure prediction models. Filice et al. (2016), who got best results in SemEval-2016 CQA task, used various types of features including 8 similarity features, 44 heuristic features and 16 thread-based features. Although achieving good performance, these methods rely heavily on feature engineering, which require a large amount of manual work and domain expertise.

To avoid feature engineering, many deep learning models have been proposed for text matching and ranking. DSSM and C-DSSM perform a non-linear projection to map the query and documents to a common semantic space (Huang et al. 2013; Shen et al. 2014). The relevance score between a query and each document is measured by the cosine similarity between their semantic vectors. DeepMatch uses a topic model to construct the interactions between two texts and then makes different levels of abstractions with a deep architecture to model the relationships between topics (Lu and Li 2013). For syntactic information, Iyyer et al. (2014) introduced a recursive neural network model that can reason over text that contained very few individual words by modeling textual compositionality. Wang and Nyberg (2015) proposed a method which uses a stacked bidirectional Long Short-Term Memory (LSTM) network to sequentially read words from question and answer sentences, and then output their relevance scores. Hu et al. (2014) investigated two different architectures ARC-I and ARC-II for matching natural language sentences and demonstrated the effectiveness of interaction. ARC-I directly learns the representation of each sentence, while ARC-II learns interactions between two sentences and then obtain representations based on the interactions. However, they did not consider the different importance of interactions, which makes their method unsuitable for modeling text in CQA.

The recent proposed attention mechanism has demonstrated good performance on machine translation (Bahdanau, Cho, and Bengio 2015), and other tasks. The idea is to adjust the machine's attentions to different regions of the text sequence that has been processed in order to make new decisions. Yin et al. (2015) proposed ABCNN for modeling a pair of sentences. Attention is used to integrate mutual influence between sentences into CNN. Thus, the representation of each sentence takes into consideration its counterpart. Santos et al. (2016) presented attentive pooling, a two-way attention mechanism for discriminative model training. It enables the pooling layer to be aware of the current input pair, in a way that information from the two input items can directly influence the computation of each other's representations. They merely use interaction to calculate attention, while in this work, to adapt to redundant and noisy text in CQA, not only interaction preserves more information than their method, but also other diverse information is utilized.

Conclusion and Future Work

In this paper, we present a novel attentive interactive neural networks for answer selection in CQA. The AI-NN learns interactions between each paired representations of question and answer. For the redundant and noisy text in CQA, we use attention mechanism to measure the importance of each segment. Four kinds of information are utilized to guide the attention. Consequently, irrelevant segments are ignored and useful segments are combined to form a representation. Four additional features are extracted from meta information. The representation and features are combined to predict a score. We conducted experiments on SemEval-2016 CQA dataset.

Our model outperforms several baselines and achieve state-of-the-art results. In future works, we will try text standardization methods to process the noisy text. Furthermore, unlabeled data is much easier to obtain than labeled data. We will explore unsupervised methods for answer selection.

Acknowledgments

Our work is supported by Major National Social Science Fund of China (No. 12&ZD227) and National Natural Science Foundation of China (No.61370117 & No.61433015). The corresponding author of this paper is Houfeng Wang.

References

Bahdanau, D.; Cho, K.; and Bengio, Y. 2015. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations (ICLR)*.

Barrón-Cedeño, A.; Filice, S.; Da, G.; Martino, S.; Joty, S.; Arquez, L. M.; Nakov, P.; and Moschitti, A. 2015. Thread-level information for comment classification in community question answering. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL-IJCNLP)*, 687–693.

Barrón-Cedeño, A.; Da San Martino, G.; Joty, S.; Moschitti, A.; Al-Obaidli, F.; Romeo, S.; Tymoshenko, K.; and Uva, A. 2016. Convkn at semeval-2016 task 3: Answer and question selection for question answering on arabic and english fora. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, 896–903.

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

Filice, S.; Croce, D.; Moschitti, A.; and Basili, R. 2016. Kelp at semeval-2016 task 3: Learning semantic relations between questions and answers. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, 1116–1123.

Hu, B.; Lu, Z.; Li, H.; and Chen, Q. 2014. Convolutional neural network architectures for matching natural language sentences. In *Advances in Neural Information Processing Systems (NIPS)*, 2042–2050.

Huang, P.-S.; He, X.; Gao, J.; Deng, L.; Acero, A.; and Heck, L. 2013. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM international conference on Conference on information and knowledge management (CIKM)*, 2333–2338.

Iyyer, M.; Boyd-Graber, J. L.; Claudino, L. M. B.; Socher, R.; and Daumé III, H. 2014. A neural network for factoid question answering over paragraphs. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 633–644.

Lu, Z., and Li, H. 2013. A deep architecture for matching short texts. In *Advances in Neural Information Processing Systems (NIPS)*, 1367–1375.

Nakov, P.; Màrquez, L.; Moschitti, A.; Magdy, W.; Mubarak, H.; Freihat, a. A.; Glass, J.; and Randeree, B. 2016. Semeval-2016 task 3: Community question answering. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, 525–545.

Pennington, J.; Socher, R.; and Manning, C. D. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 1532–43.

Qiu, X., and Huang, X. 2015. Convolutional neural tensor network architecture for community-based question answering. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI)*, 1305–1311.

Santos, C. d.; Tan, M.; Xiang, B.; and Zhou, B. 2016. Attentive pooling networks. *arXiv preprint arXiv:1602.03609*.

Shen, Y.; He, X.; Gao, J.; Deng, L.; and Mesnil, G. 2014. Learning semantic representations using convolutional neural networks for web search. In *Proceedings of the 23rd International Conference on World Wide Web (WWW)*, 373–374.

Surdeanu, M.; Ciaramita, M.; and Zaragoza, H. 2011. Learning to rank answers to non-factoid questions from web collections. *Computational Linguistics* 37(2):351–383.

Tymoshenko, K., and Moschitti, A. 2015. Assessing the impact of syntactic and semantic structures for answer passages reranking. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management (CIKM)*, 1451–1460.

Wang, D., and Nyberg, E. 2015. A long short-term memory model for answer sentence selection in question answering. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL-IJCNLP)*, 707–712.

Wang, M.; Smith, N. A.; and Mitamura, T. 2007. What is the jeopardy model? a quasi-synchronous grammar for qa. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, 22–32.

Xue, X.; Jeon, J.; and Croft, W. B. 2008. Retrieval models for question and answer archives. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, 475–482.

Yang, Y.; Yih, W.-t.; and Meek, C. 2015. Wikiqa: A challenge dataset for open-domain question answering. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2013–2018.

Yih, W.-t.; Chang, M.-W.; Meek, C.; and Pastusiak, A. 2013. Question answering using enhanced lexical semantic models. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL)*, 1744–1753.

Yin, W.; Schütze, H.; Xiang, B.; and Zhou, B. 2015. Abcnn: Attention-based convolutional neural network for modeling sentence pairs. *arXiv preprint arXiv:1512.05193*.