Answer Selection in Non-Factoid Question Answering using Convolutional Neural Network

Md Mustafizur Rahman
School of Information
University of Texas at Austin
nahid@utexas.edu

ABSTRACT

Answer selection in question answering has recently received a considerable amount of attention because the syntactic and semantic facts involved in this task cannot be captured by standard information retrieval models. However, the success of word embedding techniques along the deep learning algorithms in reducing the semantic gap in many natural language processing task indicates that question answering should be another beneficiary of these techniques. As a result, in this project, we want to investigate the approach proposed in [4], which has applied deep learning technique to answer selection task for non-factoid question answering and shown promising results.

1. INTRODUCTION

In the question answering task, one of the core tasks is to select answer from a set of candidate answers. There are several direct applications of this task. For example, this task can improve question recommendation which is the process of recommending a set of similar questions for a given new questions. In addition to that, community question answering is the direct beneficiary of this task. Typical question answer selection approaches consist of modeling some form of similarity measure between question and answer by considering different kind of textual features [15]. Some studies [19] exploit syntactic tree structures to calculate the semantic overlapping between the question/answer pair. However, the necessity of high-quality labeled data and the requirement of hand-crafted features with various external resources hinder the applicability of these models in commercial search engines.

Deep learning models have the capability of generating the feature set without any human supervision and have already obtained significant success on various natural language processing tasks, such as semantic analysis [14], machine translation [1] and text summarization [9]. Consequently, there is an ongoing research trend of applying deep learning techniques in the question answer selection domain as well. For example, deep learning based approaches like [16, 5] propose to directly learn the distributed representations of question-answer pairs.

Mathematically, the answer selection problem can be formulated as follows: Given a question q and a set of answers $\{a_1, a_2, ..., a_n\}$ for this question, the job is to search for the best answer candidate a_i where $1 \leq i \leq n$. An answer is a sequence of string with an arbitrary length, and for a given question, we can have multiple ground-truth answers.

Based on the nature of the question, the question answering task can be classified into two classes: i) factoid and ii) non-factoid question answering (QA). In factoid QA, for a given question, the task is to identify a fact which answers the question. Here the answer is usually a word or two or a phrase. In contrast, for the non-factoid QA, the answer is typically more than one sentences and the degree of word overlapping between the questions and the answers is relatively much less than the factoid QA task.

Consequently, the inherent syntactic and semantic gap between the correct answer and the question imposes a major challenge for the non-factoid QA task. Moreover, in community question answering task (another form of non-factoid QA task), most of the answers are not only noisy but also topically diverse. As a result, the main purpose of this project is to understand one deep learning technique, namely Convolutional Neural Network (CNN), via its application to the answer selection task in the non-factoid QA domain, and to gain insight about possible future directions in this research area.

The rest of the project report is organized as follows: In Section 2 we discuss related work in question answer selection domain. We discuss the method and experimental setup in Section 3 and in Section 4, respectively. Section 5 contains the results and performance analysis of the proposed model. We conclude and provide possible future directions in Section 6.

2. RELATED WORK

Traditionally feature engineering, linguistic tools, or external resources are used for question answer selection. Semantic features constructed using the WordNet are used in [22]. Syntactic matching on the parse tree based representation between the question/answer pair is proposed by [20] to solve the answer selection problem. [11, 21] extends the work of [20] with the help of edit distance similarity measure between the dependency parse tree representation of the question/answer pair. However, the inherent constraints associated in these methods are the availability of additional resources (e.g. WordNet), and the effort of manual feature engineering. These methods also suffer from systematic complexity introduced by various linguistic tools, such as parse and dependency trees.

The burgeoning need of developing the linguistic features automatically from the text corpus places the deep learning techniques in this QA domain. However, most of the deep learning approaches focus on solving the factoid QA task. The authors in [10] propose to learn the vector representation of the questions and the answers using a deep

convolutional neural network. Although the model [10] originally proposed for matching the short text pairs, it also performs well in factoid QA task. The proposed model consists of multiple convolutional layers to process the questions and the answers separately. Interestingly, they compute the amount of overlapping between the CNN representation of these questions and answers using some form of similarity measure and use this value as a feature for the next dense layer of the deep network along with the representation of the questions and the answers from the previous layer.

Factoid QA task is also explored using the recurrent neural networks (RNNs) in [6]. However, in their data set, multiple questions can have the same answers and the questions usually contain multiple tokens whereas answers mostly contain single token. The basic model builds on the RNN with a rank specific loss function to capture the compositionality of the text.

The authors in [18, 17] explore the applicability of the bidirectional long short term memory (BLSTM) on the TREC factoid question answering task. Two separate BLSTMs are used to read the question and answer sequentially and then the hidden memory vectors produced by these BLSTMs are combined. The authors in [18] use the pre-trained word embedding. In order to outperform the other non-deep learning approaches, they use BM25 with the help of Gradient boosted regression tree (GBRT).

From the above discussion, it is implied that most of the deep learning approaches try to solve the answer selection task for the factoid question QA. These approaches can be largely classified into two groups: the first group of solutions [4, 23, 3, 10] learn the representation of the questions and the answers separately using some deep networks and use a similarity metric between these representations to perform the answer selection; and the second group [18] learns the joint representation of the questions and the answers and perform a learning-to-rank approach to provide a list of answers. Besides, all these discussed deep approaches process the questions and the answers using the pre-trained word embedding proposed by [8].

Even though deep learning approaches are quite successful for the factoid QA task, the same is not true for the non-factoid QA task. Specially the nature of the answers for the non-factoid QA typically imposes a huge challenge as we discussed in Section 1. In fact, it is shown in [2] that the method proposed in [10] to solve the factoid QA task provides extremely inferior performance for the non-factoid QA task. To the best of our knowledge only a small handful of deep neural network approaches [12, 2, 4] are proposed to solve the non-factoid QA task until now.

Cohen et al. [2] solve the problem of non-factoid QA task by employing BLSTM with an embedding layer. While the previous deep learning approaches learn the distributed representation of the words independent of the network using a pre-trained word embedding vector, this approach [2] back-propagates the error to the embedding layer to best fit the proposed rank specific loss function. The end to end nature of the proposed BLSTM based approach performs quite well on the Yahoo's Web scope L4 data set[1] [13].

A deep feature fusion network (DFFN) is proposed in [12] to leverage both the manually crafted features and the deep neural network features for the community question answering task. DFFN learns the representation of the questions and the answers jointly using a CNN, then pass these features along with the manually crafted features to the fully connected network in the second layer. DFFN outperforms many other approaches submitted in SemEval 2015 with the help of the manually crafted features. According to the authors, some of those manually crafted features are deemed necessary for the performance gain as these features are extremely hard to learn by the machine.

[1]https://webscope.sandbox.yahoo.com/catalog.php?datatype=l

3. METHOD

In this section, at first we describe the proposed deep learning architecture for the question answering task. Next the training algorithm is discussed in details with the specific loss function. Finally, we discuss about the set of similarity measures used in [4].

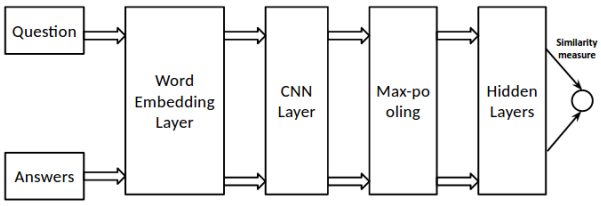

Figure 1: Proposed QA-CNN architecture [4] (Architecture I)

3.1 Proposed QA-CNN architecture

The architectural model [4] is based on the fact that we have to learn a higher level semantic representation of a given question and its answer candidate. Once we have that form of representation, we can use a similarity metric to measure matching overlap between the question and the answers. The basic architecture consists of one word embedding layer and one CNN layer for question and answer separately. The CNN layer is followed by the max-pooling layer and the fully connected hidden layer which is defined as $z = tanh(Wx + B)$, where W is the weight matrix, B is the bias, x is the input and z is the output of the activation function $tanh$. The final step is to compute the similarity between each question/answer pair using one of the similarity measures discussed in Table 1. The answer with the highest similarity value is returned as the positive answer. Fig. 1 represents the graphical representation of the proposed architecture. We refer this model as Question Answering Convolutional Neural Network (QA-CNN) model.

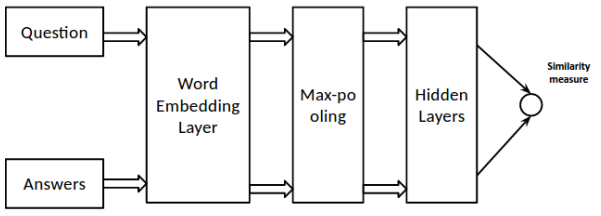

Figure 2: Architecture II

Several architectural variants are possible if we take a close look at the Fig. 1. For example, the architecture II (Fig. 2) does not have any convolutional layer. There are separate embedding layers for each question and its answer candidate which is followed by the max-pooling layer applied on these embedding layers to generate the vector representation of the questions and the answers. Fig. 3 presents the architecture III where there is no fully connected hidden layer after the max-pooling layer, whereas in the architecture IV (Fig. 4) there can be multiple hidden layers after the max-pooling layer.

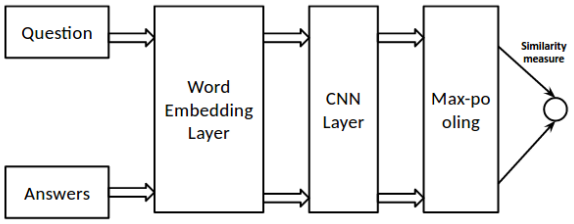

Figure 3: Architecture III

3.2 Training and Loss Function

Before training the model we need to generate the training data set. For each training question Q, there is one correct answer, which we refer as positive answer A^+ (the ground truth). A training instance is formed by pairing a positive answer A^+ with a negative answer A^- (a wrong answer). It is to be noted that A^- is sampled from the whole answer space. The job of this proposed deep learning architecture is to generate the semantic representation for the question (V_Q) and the two candidates (V_A^+ and V_A^-). Next, a specific similarity measure (see Table 1) is used to calculated the similarities $sim(V_Q, V_{A^+})$ and $sim(V_Q, V_{A^-})$. Finally, the margin: $sim(V_Q, V_{A^+}) - sim(V_Q, V_{A^-}) \leq m$ where m is the margin, is used to compare the distance between these two. This condition of the margin will be satisfied when the proposed architecture ranks the positive answer below the negative answer. As a result, an update of the parameters is done. However, if $sim(V_Q, V_{A^+}) - sim(V_Q, V_{A^-}) \geq m$, then there is no update of parameters and a new negative sample is drawn from the answer set. Therefore, the hinge loss function is defined as follows:

$$L = \max\{0, m - sim(V_Q, V_{A^+}) + sim(V_Q, V_{A^-})\} \quad (1)$$

For testing, the similarity $sim(V_Q, V_{candidate})$ is calculated between the question Q and each answer candidate $V_{candidate}$ from the pool of size 500. The candidate answer with the largest similarity score with the question is selected as a positive answer.

3.3 Similarity Measure

The authors in [4] proposed to use nine different similarity measures. Table 1 provides a brief description of those similarity measures. γ, c and d are three user defined parameters for these similarity measures. Among these similarity measures, two new similarity measures are introduced in [4]: i) Geometric mean of Euclidean and Sigmoid Dot product (GESD) and ii) Arithmetic mean of Euclidean and Sigmoid Dot product (AESD). According to [4], both of these similarity measures performed well for the insurance QA data set.

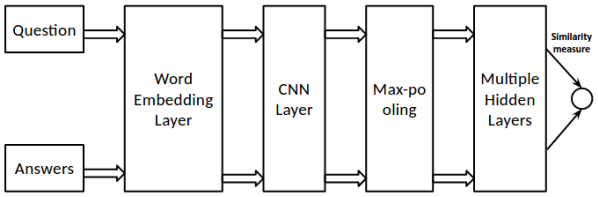

Figure 4: Architecture IV

Table 1: Several similarity measures are used in [4]. sim(x,y) denotes the similarity between vector x and y. $||x||$ is the L_2 norm and $||x||_1$ is the L_1 norm. xy^T represents the inner product of x and y. γ, c and d are the user defined parameters.

Similarity Measure	Expression
cosine	$sim(x,y) = \frac{xy^T}{\|\|x\|\|\|\|y\|\|}$
polynomial	$sim(x,y) = (\gamma xy^T + c)^d$
Sigmoid	$sim(x,y) = tanh(\gamma xy^T + c)$
RBF	$sim(x,y) = \exp(-\gamma\|\|x-y\|\|^2)$
euclidean	$sim(x,y) = \frac{1}{1+\|\|x-y\|\|}$
exponential	$sim(x,y) = \exp(-\gamma\|\|x-y\|\|_1)$
Manhattan	$sim(x,y) = \frac{1}{1+\|\|x-y\|\|_1}$
GESD	$sim(x,y) = \frac{1}{1+\|\|x-y\|\|} * \frac{1}{1+\exp(-\gamma(xy^T+c))}$
AESD	$sim(x,y) = \frac{0.5}{1+\|\|x-y\|\|} * \frac{1}{1+\exp(-\gamma(xy^T+c))}$

4. EXPERIMENTAL SETTINGS

In this section, we discuss the experimental setup which includes the value of the different parameters, the data set and the set of evaluation metrics.

4.1 Parameter Settings

The word embedding is trained by word2vec [8] on the whole insurance QA data set and used for the initialization of the embedding layer. Training of the word embedding is done using 100, 300 and 500 dimensions. The batch-size of Stochastic Gradient Descent (SGD) is set to 100. The initial learning rate for the Adam optimizer [7] is set to 0.001 with hyper-parameter value $\beta_1 = 0.9$, $\beta_2 = 0.999$ and error rate $\epsilon = 1 * e^{-08}$. The margin (m) for the hinge loss function is set to 0.05. The number of filters in the convolutional layer is set to 500 and the filter length of these filters are 2, 3, 5 and 7.

4.2 Data set

For experimental evaluation, we use the data set (insurance QA) provided by [4]. The data set is divided into three parts: development, test1 and test2. The task is to identify an answer for each question in each part of the corpus. There are in total 24,981 unique answers in the insurance QA data set [4]. The authors in [4] argue that they should use the whole answer set to generate the training instance. But considering the computational cost, they prefer to use only 500 randomly chosen answers as a whole answer set. Table 2 presents the basic statistics of this insurance QA data set.

Table 2: Insurance QA corpus statistics collected from [4]. The number of questions and the answers are represented by the first two columns. Some questions can have multiple answers which is the reason the number mismatch between the first two columns. The total number of words in the question corpus is presented in column three

	Questions	Answers	Word Count
Train	12887	18540	92095
Dev	1000	1454	7158
Test 1	1800	2616	12893
Test 2	1800	2593	12905

4.3 Evaluation Metrics

Because of its similarity with ad-hoc retrieval task, the answer selection task in question answering can be viewed as a ranking problem. Using the similarity score, the proposed model generates a ranked list of the answers for each question. The basic objective of the ranker algorithm is to place the positive answer at a higher rank position than the negative answer. Since most of questions have only one correct answer, we prefer to use Precision at 1 (P@1) and Mean Reciprocal Rank (MRR) as evaluation metrics. The formal definition of P@1 and MRR are described as follows:

$$P@1 = (\frac{1}{N}\sum_{i=1}^{N}\delta(rank(A^+) = 1)) \quad (2)$$

$$MRR = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{rank(A^+)} \quad (3)$$

where N is the number of questions, A^+ is the ground truth, $rank(.)$ represents the ranking position of the answer in the list and δ is the indicator function to indicate the whether the rank position is 1 or not.

5. RESULTS AND DISCUSSIONS

In this section, we analyze the performance of the proposed architecture for the answer selection task in question answering domain. Unless otherwise specified, for all the following experiments, we use the insurance QA data set. In addition to that, the number of epochs is set to 100.

5.1 Effect of the number of dimensions of word embedding

The authors in [4] report all the experimental analysis using 100 dimensions of word embedding. Mikolov et al. [8] argue that increasing the number of dimensions of the word embedding does not necessarily help. As a result, we want to know that how the number of dimensions of word embedding affects the performance of the answer selection task in the non-factoid QA. We use the proposed QA-CNN model (Fig. 1) with GESD as a similarity measure. We vary the number of dimensions from 100 to 500 with the step size of 200. The result of these three different dimensions of word embedding are presented in Table 3.

From Table 3, we can find that the performance of the QA-CNN model does not improve with the increased number of dimensions. This finding is true for all the three different corpus settings.

Table 3: Performance comparison of QA-CNN (Fig. 1) model among different number of dimensions of word embedding. P@1 and MRR are reported using GESD with $\gamma = 1$

Number of dimensions	Test set	P@1	MRR
100	**Dev**	0.284	0.284
	Test 1	0.280	0.280
	Test 2	0.250	0.250
300	**Dev**	0.225	0.225
	Test 1	0.215	0.216
	Test 2	0.206	0.205
500	**Dev**	0.220	0.220
	Test 1	0.201	0.202
	Test 2	0.179	0.178

5.2 Effect of similarity measure

The experimental evidences in [4] suggest that GESD and AESD provide superior performance over the cosine similarity. In order to verify the authors' claim, we design an experimental setup using five different similarity measures from Table 1. In this experimental setting, the QA-CNN based model (Fig. 1) is used and the number of epochs for the training of this architecture is set to 100. Table 4 depicts the result for this setting.

Table 4: Performance comparison of QA-CNN model using different similarity measures. P@1 and MRR are reported using 100 dimensions of word embedding. The best result across a column is highlighted in bold faced font

Similarity	Dev		Test 1		Test 2	
metrics	P@1	MRR	P@1	MRR	P@1	MRR
cosine	0.004	0.005	0.002	0.002	0.001	0.002
euclidean	**0.337**	**0.449**	**0.323**	**0.447**	**0.295**	**0.410**
GESD	0.284	0.284	0.280	0.280	0.250	0.250
exponential	0.257	0.363	0.260	0.378	0.235	0.342
RBF	0.060	0.111	0.056	0.103	0.049	0.096

From Table 4, it is evident that both in terms of P@1 and MRR, euclidean similarity measure outperforms all the others similarity measures. We can also find that GESD outperforms cosine similarity measure by a significant margin in all three testing sets but it is second to the euclidean similarity measure. This experimental finding contradicts the evidence of the superior performance of GESD as reported in [4].

To investigate this issue further, we develop another experimental setup with the same setting as discussed above except this time we use the architecture II (Fig. 2). The experimental result for this setting is reported in Table 5.

From Table 5, we can find that GESD outperforms euclidean in dev and test1 set in terms of MRR but in terms of P@1 euclidean outperforms GESD. However, for the test2 set, the exponential measure outperforms all the other similarity measures. These experimental evidences suggest us that performance variation for different similarity measures largely depends on the architecture of the model.

Table 5: Performance comparison of the second architecture (Fig. 2) using different similarity measures. P@1 and MRR are reported using 100 dimensions of word embedding. The best result across a column is highlighted in bold faced font

Similarity	Dev		Test 1		Test 2	
metrics	P@1	MRR	P@1	MRR	P@1	MRR
cosine	0.004	0.008	0.002	0.008	0.002	0.006
euclidean	0.333	**0.429**	0.307	**0.415**	0.297	**0.395**
GESD	**0.344**	0.344	**0.320**	0.321	0.297	0.298
exponential	0.339	0.423	0.318	0.414	**0.307**	0.391
RBF	0.161	0.235	0.150	0.224	0.135	0.208

5.3 Effect of the convolutional layer

As we discuss in Section 3 that several architectural variations are possible for the architecture I (Fig. 1). One such variation is to simply remove the convolutional layer which generates the architecture II as shown in Fig. 2. In this experimental setting, we are interested to understand the effect of the convolutional layer for the answer selection in the non-factoid QA task. We already have the experimental result for the architecture I using different dimensions of word embedding in Table 3. Now, we report the same experimental result using the architecture II in Table 6.

Table 6: Performance comparison of the architecture II (Fig. 2) among different number of dimensions of word embedding

Number of dimensions	Test set	P@1	MRR
100	**Dev**	0.344	0.344
	Test 1	0.320	0.321
	Test 2	0.297	0.298
300	**Dev**	0.321	0.321
	Test 1	0.317	0.317
	Test 2	0.303	0.304
500	**Dev**	0.297	0.297
	Test 1	0.299	0.299
	Test 2	0.285	0.285

From Table 3 and 6, it is evident that architecture II is providing superior performance over the architecture I in all three testing sets for all three types of word embedding dimensions. This experimental evidence again contradicts with the finding reported in [4]. To verify this issue further, we can further take a look at the result reported in Table 4 and 5. Table 4 and 5 report the experimental finding for the architecture I and II respectively using different similarity measures. Taking a closer look at these Table 4 and 5 again reveals the fact that the architecture II is a better option than the architecture I in answer selection task.

Several issues can contribute to the better performance of the architecture II. For example, the authors in [4] set the number of epochs is 1000 with 1000 filters for the convolutional layer of architecture I. However, in our experimental setting, considering the computational cost, we set the number of filters to 500 with only 100 number of epochs for the architecture I. As a result, we hypothesize that, with the reduced number of filters and epochs the architecture I cannot outperform the architecture II.

Table 7: Performance comparison among different architectures due to the hidden layer. The best result across a column is highlighted in bold faced font

Similarity	Dev		Test 1		Test 2	
metrics	P@1	MRR	P@1	MRR	P@1	MRR
Architecture I	**0.284**	**0.284**	**0.280**	**0.280**	**0.250**	**0.250**
Architecture III	0.266	0.266	0.280	0.280	0.247	0.247
Architecture IV	0.283	0.282	0.278	0.279	0.248	0.249

5.4 Effect of the hidden layer

Based on the presence or absence of the hidden layer after the max-pooling layer, we have two variants of the architecture I which are shown in Fig. 3 and 4. In the architecture III, we have no hidden layer after the max-pooling layer, but in the architecture IV, we have one two hidden layers after the max-pooling layer. The first and second hidden layer of the architecture IV contain 400 and 100 nodes respectively. We employ the same *tanh* activation function for all these hidden layer nodes. The other parameters of these three architectures remain same as discussed earlier. The experimental result for these three different architectures are presented in Table 7.

It is imperative from Table 7 that adding one hidden layer (architecture I) after the max-polling layer provides some performance gain against no hidden layer (architecture III). However, we observe the performance gain only for dev and test2 set but for test1 set there is no significant difference in performance between these two architectures. Additionally, we find that adding more than one hidden layers after the max-pooling layers provides inferior performance than one hidden layer. The contributing factors for this result includes the number of hidden layer nodes, the activation function etc. As a result, we envision that trying different number of nodes with different activation functions (e.g. sigmoid) can improve the performance of the architecture IV.

6. CONCLUSION AND FUTURE WORKS

Non-factoid question answering is becoming an integral component of modern e-commerce (e.g. Amazon, Walmart) and question-answering (e.g. Yahoo answers, Quora, Stackoverflow) systems. Most of these systems largely depend on the human judged voting mechanism (e.g. up vote in Quora, rating in Amazon) given the answer is already there. These human judged voting mechanism has many inherent shortcomings which includes the lack of domain knowledge, the absence of full topical coverage of the questions etc. As a result, there is a growing need for an automated question answering system for non-factoid QA. Deep learning techniques developed for factoid QA cannot be directly applied on the non-factoid QA [2] and thus we need more sophisticated and fine-grained deep learning models for the non-factoid QA.

In this project, we develop a convolutional neural network based model to perform the answer selection task for the non-factoid QA. We also implement three variants of the proposed QA-CNN model. Experimental analysis based on the insuranceQA data set [4] suggests that word embedding layer with max-pooling performs better than convolutional layer based architectures. We also find out that adding one hidden layer after the max-pooling layer is better than adding more than one hidden layers which saves a

lot of computational cost.

In the future work, we want to investigate other deep architectures including recurrent neural networks (RNNs) or Long Short-Term Memory networks (LSTMs) and their effect on the question answer selection task. Additionally, we are interested to apply our QA-CNN model on the stackoverflow question answering data set[2] because the presence of both the text and the image in the question/answer pairs makes the non-factoid QA for this data set a challenging one.

References

[1] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[2] D. Cohen and W. B. Croft. End to end long short term memory networks for non-factoid question answering. In *Proceedings of the 2016 ACM on International Conference on the Theory of Information Retrieval*, pages 143–146. ACM, 2016.

[3] C. dos Santos, L. Barbosa, D. Bogdanova, and B. Zadrozny. Learning hybrid representations to retrieve semantically equivalent questions. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, volume 2, pages 694–699, 2015.

[4] M. Feng, B. Xiang, M. R. Glass, L. Wang, and B. Zhou. Applying deep learning to answer selection: A study and an open task. In *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pages 813–820. IEEE, 2015.

[5] H. Hu, B. Liu, B. Wang, M. Liu, and X. Wang. Multimodal dbn for predicting high-quality answers in cqa portals. In *ACL (2)*, pages 843–847, 2013.

[6] M. Iyyer, J. L. Boyd-Graber, L. M. B. Claudino, R. Socher, and H. Daumé III. A neural network for factoid question answering over paragraphs. In *EMNLP*, pages 633–644, 2014.

[7] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[8] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[9] A. M. Rush, S. Chopra, and J. Weston. A neural attention model for abstractive sentence summarization. *arXiv preprint arXiv:1509.00685*, 2015.

[10] A. Severyn and A. Moschitti. Learning to rank short text pairs with convolutional deep neural networks. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 373–382. ACM, 2015.

[2]https://archive.org/details/stackexchange

[11] M. H. N. A. Smith. Tree edit models for recognizing textual entailments, paraphrases, and answers to questions.

[12] S. P. Suggu, K. N. Goutham, M. K. Chinnakotla, and M. Shrivastava. Deep feature fusion network for answer quality prediction in community question answering. *arXiv preprint arXiv:1606.07103*, 2016.

[13] M. Surdeanu, M. Ciaramita, and H. Zaragoza. Learning to rank answers on large online qa collections. In *ACL*, volume 8, pages 719–727, 2008.

[14] D. Tang, B. Qin, and T. Liu. Document modeling with gated recurrent neural network for sentiment classification. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1422–1432, 2015.

[15] B. Wang, B. Liu, C. Sun, X. Wang, and L. Sun. Extracting chinese question-answer pairs from online forums. In *Systems, Man and Cybernetics, 2009. SMC 2009. IEEE International Conference on*, pages 1159–1164. IEEE, 2009.

[16] B. Wang, X. Wang, C. Sun, B. Liu, and L. Sun. Modeling semantic relevance for question-answer pairs in web social communities. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1230–1238. Association for Computational Linguistics, 2010.

[17] D. Wang and E. Nyberg. A recurrent neural network based answer ranking model for web question answering. In *WebQA Workshop, SIGIR*, volume 15.

[18] D. Wang and E. Nyberg. A long short-term memory model for answer sentence selection in question answering. *ACL, July*, 2015.

[19] K. Wang, Z. Ming, and T.-S. Chua. A syntactic tree matching approach to finding similar questions in community-based qa services. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 187–194. ACM, 2009.

[20] M. Wang and C. D. Manning. Probabilistic tree-edit models with structured latent variables for textual entailment and question answering. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 1164–1172. Association for Computational Linguistics, 2010.

[21] X. Yao, B. Van Durme, C. Callison-Burch, and P. Clark. Answer extraction as sequence tagging with tree edit distance. In *HLT-NAACL*, pages 858–867. Citeseer, 2013.

[22] W.-t. Yih, M.-W. Chang, C. Meek, and A. Pastusiak. Question answering using enhanced lexical semantic models. 2013.

[23] L. Yu, K. M. Hermann, P. Blunsom, and S. Pulman. Deep learning for answer sentence selection. *arXiv preprint arXiv:1412.1632*, 2014.