LSTM-BASED DEEP LEARNING MODELS FOR NON-FACTOID ANSWER SELECTION

Ming Tan, Cicero dos Santos, Bing Xiang & Bowen Zhou
IBM Watson Core Technologies
Yorktown Heights, NY, USA
{mingtan,cicerons,bingxia,zhou}@us.ibm.com

ABSTRACT

In this paper, we apply a general deep learning framework for the answer selection task, which does not depend on manually defined features or linguistic tools. The basic framework is to build the embeddings of questions and answers based on bidirectional long short-term memory (biLSTM) models, and measure their closeness by cosine similarity. We further extend this basic model in two directions. One direction is to define a more composite representation for questions and answers by combining convolutional neural network with the basic framework. The other direction is to utilize a simple but efficient attention mechanism in order to generate the answer representation according to the question context. Several variations of models are provided. The models are examined by two datasets, including TREC-QA and InsuranceQA. Experimental results demonstrate that the proposed models substantially outperform several strong baselines.

1 INTRODUCTION

The answer selection problem can be formulated as follows: Given a question q and an answer candidate pool $\{a_1, a_2, \cdots, a_s\}$ for this question, we aim to search for the best answer candidate a_k, where $1 \leq k \leq s$. An answer is a token sequence with an arbitrary length, and a question can correspond to multiple ground-truth answers. In testing, the candidate answers for a question may not be observed in the training phase. Answer selection is one of the essential components in typical question answering (QA) systems. It is also a stand-alone task with applications in knowledge base construction and information extraction.

The major challenge of this task is that the correct answer might not directly share lexical units with the question. Instead, they may only be semantically related. Moreover, the answers are sometimes noisy and contain a large amount of unrelated information.

Recently, deep learning models have obtained a significant success on various natural language processing tasks, such as semantic analysis (Tang et al., 2015), machine translation (Bahdanau et al., 2015) and text summarization (Rush et al., 2015).

In this paper, we propose a deep learning framework for answer selection which does not require any feature engineering, linguistic tools, or external resources. This framework is based on building bidirectional long short term memory (biLSTM) models on both questions and answers respectively, connecting with a pooling layer and utilizing a similarity metric to measure the matching degree. We improve this basic model from two perspectives. Firstly, a simple pooling layer may suffer from the incapability of keeping the local linguistic information. In order to obtain better embeddings for the questions and answers, we build a convolutional neural network (CNN) structure on top of biLSTM. Secondly, in order to better distinguish candidate answers according to the question, we introduce a simple but efficient attention model to this framework for the answer embedding generation according to the question context.

We report experimental results for two answer selection datasets: (1) InsuranceQA (Feng et al., 2015) [1], a recently released large-scale non-factoid QA dataset from the insurance domain. The

[1] git clone https://github.com/shuzi/insuranceQA.git

proposed models demonstrate a significant out-performance compared to two non-DL baselines and a strong DL baseline based on CNN. (2) TREC-QA [2], which was created by Wang et al. (2007) based on Text REtrieval Conference (TREC) QA track data. The proposed models outperform various strong baselines.

The rest of the paper is organized as follows: Section 2 describes the related work for answer selection; Section 3 provides the details of the proposed models; Experimental settings and results of InsuranceQA and TREC-QA datasets are discussed in section 4 and 5 respectively; Finally, we draw conclusions in section 6.

2 RELATED WORK

Previous work on answer selection normally used feature engineering, linguistic tools, or external resources. For example, semantic features were constructed based on WordNet in (Yih et al., 2013). This model pairs semantically related words based on word semantic relations. In (Wang & Manning, 2010; Wang et al., 2007), the answer selection problem is transformed to a syntactical matching between the question/answer parse trees. Some work tried to fulfill the matching using minimal edit sequences between dependency parse trees (Heilman & Smith, 2010; Yao et al., 2013). Recently, discriminative tree-edit features extraction and engineering over parsing trees were automated in (Severyn & Moschitti, 2013).

While these methods show effectiveness, they might suffer from the availability of additional resources, the effort of feature engineering and the systematic complexity by introducing linguistic tools, such as parse trees and dependency trees.

There were prior methods using deep learning technologies for the answer selection task. The approaches for non-factoid question answering generally pursue the solution on the following directions: Firstly, the question and answer representations are learned and matched by certain similarity metrics (Feng et al., 2015; Yu et al., 2014; dos Santos et al., 2015; Shen et al., 2014; Huang et al., 2013). Secondly, a joint feature vector is constructed based on both the question and the answer, and then the task can be converted into a classification or learning-to-rank problem (Wang & Nyberg, 2015). Finally, recently proposed models for textual generation can intrinsically be used for answer selection and generation (Bahdanau et al., 2015; Vinyals & Le, 2015).

The framework proposed in this work belongs to the first category. There are two major differences between our approaches and the work in (Feng et al., 2015): (1) The architectures developed in (Feng et al., 2015) are only based on CNN, whereas our models are based on bidirectional LSTMs, which are more capable of exploiting long-range sequential context information. Moreover, we also integrate the CNN structures on the top of biLSTM for better performance. (2) Feng et al. (2015) tackle the question and answer independently, while the proposed structures develop an efficient attentive models to generate answer embeddings according to the question.

3 APPROACH

In this section, we describe the proposed framework and its variations. We first introduce the general framework, which is to build bi-directional LSTM on both questions and their answer candidates, and then use the similarity metric to measure the distance of question answer pairs. In the following two subsections, we extend the basic model in two independent directions.

3.1 BASIC MODEL: QA-LSTM

Long Short-Term Memory (LSTM): Recurrent Neural Networks (RNN) have been widely exploited to deal with variable-length sequence input. The long-distance history is stored in a recurrent hidden vector which is dependent on the immediate previous hidden vector. LSTM (Hochreiter & Schmidhuber, 1997) is one of the popular variations of RNN to mitigate the gradient vanish problem of RNN. Our LSTM implementation is similar to the one in (Graves et al., 2013) with minor

[2]The data is obtained from (Yao et al., 2013) `http://cs.jhu.edu/~xuchen/packages/jacana-qa-naacl2013-data-results.tar.bz2`

modification. Given an input sequence $\mathbf{x} = \{\mathbf{x}(1), \mathbf{x}(2), \cdots, \mathbf{x}(n)\}$, where $\mathbf{x}(t)$ is an E-dimension word vector in this paper. The hidden vector $\mathbf{h}(t)$ (the size is H) at the time step t is updated as follows.

$$
\begin{aligned}
i_t &= \sigma(\mathbf{W}_i \mathbf{x}(t) + \mathbf{U}_i \mathbf{h}(t-1) + \mathbf{b}_i) && (1)\\
f_t &= \sigma(\mathbf{W}_f \mathbf{x}(t) + \mathbf{U}_f \mathbf{h}(t-1) + \mathbf{b}_f) && (2)\\
o_t &= \sigma(\mathbf{W}_o \mathbf{x}(t) + \mathbf{U}_o \mathbf{h}(t-1) + \mathbf{b}_o) && (3)\\
\tilde{C}_t &= \tanh(\mathbf{W}_c \mathbf{x}(t) + \mathbf{U}_c \mathbf{h}(t-1) + \mathbf{b}_c) && (4)\\
C_t &= i_t * \tilde{C}_t + f_t * C_{t-1} && (5)\\
\mathbf{h}_t &= o_t * \tanh(C_t) && (6)
\end{aligned}
$$

In the LSTM architecture, there are three gates (input i, forget f and output o), and a cell memory vector c. σ is the $sigmoid$ function. The input gate can determine how incoming vectors x_t alter the state of the memory cell. The output gate can allow the memory cell to have an effect on the outputs. Finally, the forget gate allows the cell to remember or forget its previous state. $\mathbf{W} \in R^{H \times E}$, $\mathbf{U} \in R^{H \times H}$ and $\mathbf{b} \in R^{H \times 1}$ are the network parameters.

Bidirectional Long Short-Term Memory (biLSTM): Single direction LSTMs suffer a weakness of not utilizing the contextual information from the future tokens. Bidirectional LSTM utilizes both the previous and future context by processing the sequence on two directions, and generate two independent sequences of LSTM output vectors. One processes the input sequence in the forward direction, while the other processes the input in the reverse direction. The output at each time step is the concatenation of the two output vectors from both directions, ie. $h_t = \overrightarrow{h_t} \parallel \overleftarrow{h_t}$.

QA-LSTM: The basic model in this work is shown in Figure 1. BiLSTM generates distributed representations for both the question and answer independently, and then utilize cosine similarity to measure their distance. Following the same ranking loss in (Feng et al., 2015; Weston et al., 2014; Hu et al., 2014), we define the training objective as a hinge loss.

$$L = \max\{0, M - cosine(q, a_+) + cosine(q, a_-)\} \quad (7)$$

where a_+ is a ground truth answer, a_- is an incorrect answer randomly chosen from the entire answer space, and M is constant margin. We treat any question with more than one ground truth as multiple training examples, each for one ground truth.

There are three simple ways to generate representations for questions and answers based on the word-level biLSTM outputs: (1) Average pooling; (2) max pooling; (3) the concatenation of the last vectors on both directions. The three strategies are compared with the experimental performance in Section 5. Dropout operation is performed on the QA representations before cosine similarity matching.

Finally, from preliminary experiments, we observe that the architectures, in which both question and answer sides share the same network parameters, is significantly better than the one that the question and answer sides own their own parameters separately, and converges much faster. As discussed in (Feng et al., 2015), this is reasonable, because for a shared layer network, the corresponding elements in question and answer vectors represent the same biLSTM outputs. While for the network with separate question and answer parameters, there is no such constraint and the model has double-sized parameters, making it difficult to learn for the optimizer.

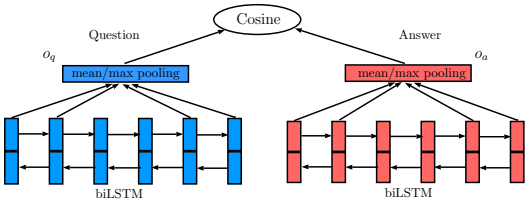


Figure 1: Basic Model: QA-LSTM

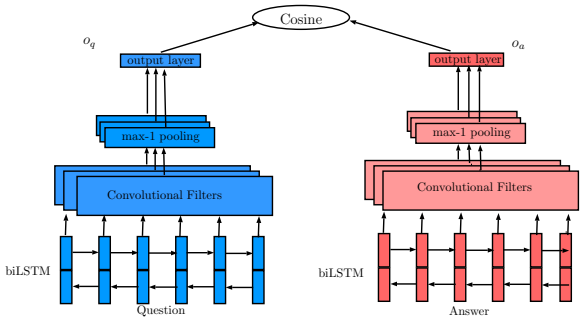


Figure 2: QA-LSTM/CNN

3.2 QA-LSTM/CNN

In the previous subsection, we generate the question and answer representations only by simple operations, such as max or mean pooling. In this subsection, we resort to a CNN structure built on the outputs of biLSTM, in order to give a more composite representation of questions and answers.

The structure of CNN in this work is similar to the one in (Feng et al., 2015), as shown in Figure 2. Unlike the traditional forward neural network, where each output is interactive with each input, the convolutional structure only imposes local interactions between the inputs within a filter size m.

In this work, for every window with the size of m in biLSTM output vectors, ie. $\mathbf{H}_m(t) = [\mathbf{h}(t), \mathbf{h}(t+1), \cdots, \mathbf{h}(t+m-1)]$, where t is a certain time step, the convolutional filter $\mathbf{F} = [\mathbf{F}(0) \cdots \mathbf{F}(m-1)]$ will generate one value as follows.

$$o_F(t) \quad = \quad \tanh\left[\left(\sum_{i=0}^{m-1} \mathbf{h}(t+i)^T \mathbf{F}(i)\right) + b\right] \tag{8}$$

where b is a bias, and $\mathbf{F}$ and b are the parameters of this single filter.

Same as typical CNNs, a max-k pooling layer is built on the top of the convolutional layer. Intuitively, we want to emphasize the top-k values from each convolutional filter. By k-MaxPooling, the maximum k values will be kept for one filter, which indicate the highest degree that a filter matches the input sequence.

Finally, there are N parallel filters, with different parameter initialization, and the convolutional layer gets N-dimension output vectors. We get two output vectors with dimension of kN for the questions and answers respectively. In this work, $k = 1$. $k > 1$ did not show any obvious improvement in our early experiments. The intuition of this structure is, instead of evenly considering the lexical information of each token as the previous subsection, we emphasize on certain parts of the answer, such that QA-LSTM/CNN can more effectively differentiate the ground truths and incorrect answers.

3.3 Attention-based QA-LSTM

In the previous subsection, we described one extension from the basic model, which targets at providing more composite embeddings for questions and answers respectively. In this subsection, we investigate an extension from another perspective. Instead of generating QA representation independently, we leverage a simple attention model for the answer vector generation based on questions.

The fixed width of hidden vectors becomes a bottleneck, when the bidirectional LSTM models must propagate dependencies over long distances over the questions and answers. An attention mechanism are used to alleviate this weakness by dynamically aligning the more informative parts of answers to the questions. This strategy has been used in many other natural language processing tasks, such as machine translation (Bahdanau et al., 2015; Sutskever et al., 2014), sentence summarization (Rush et al., 2015) and factoid question answering (Hermann et al., 2015; Sukhbaatar et al., 2015).

	Train	Validation	Test1	Test2
# of Qs	12887	1000	1800	1800
# of As	18540	1454	2616	2593

Table 1: Numbers of questions and answers of InsuranceQA.

Inspired by the work in (Hermann et al., 2015), we develop a very simple but efficient word-level attention on the basic model. Figure 3 shows the structure. Prior to the average or mean pooling, each biLSTM output vector will be multiplied by a softmax weight, which is determined by the question embedding from biLSTM.

Specifically, given the output vector of biLSTM on the answer side at time step t, $\mathbf{h}_a(t)$, and the question embedding, $\mathbf{o}_q$, the updated vector $\widetilde{\mathbf{h}}_a(t)$ for each answer token are formulated below.

$$\mathbf{m}_{a,q}(t) = \tanh(\mathbf{W}_{am}\mathbf{h}_a(t) + \mathbf{W}_{qm}\mathbf{o}_q) \quad (9)$$

$$s_{a,q}(t) \propto \exp(\mathbf{w}_{ms}^T \mathbf{m}_{a,q}(t)) \quad (10)$$

$$\widetilde{\mathbf{h}}_a(t) = \mathbf{h}_a(t) s_{a,q}(t) \quad (11)$$

where $\mathbf{W}_{am}$, $\mathbf{W}_{qm}$ and $\mathbf{w}_{ms}$ are attention parameters. Conceptually, the attention mechanism give more weights on certain words, just like tf-idf for each word. However, the former computes the weights according to question information.

The major difference between this approach and the one in (Hermann et al., 2015) is that Hermann et al. (2015)'s attentive reader emphasizes the informative part of supporting facts, and then uses a combined embedding of the query and the supporting facts to predict the factoid answers. In this work, we directly use the attention-based representations to measure the question/answer distances. Experiments indicate the attention mechanism can more efficiently distinguish correct answers from incorrect ones according to the question text.

3.4 QA-LSTM/CNN WITH ATTENTION

The two extensions introduced previously are combined in a simple manner. First, the biLSTM hidden vectors of answers $\mathbf{h}_a(t)$ are multiplied by $s_{a,q}(t)$, which is computed from the question average pooling vectors o_q, and updated to $\widetilde{\mathbf{h}}_a(t)$, illustrated in Eq. 9-11. Then, the original question and updated answer hidden vectors serve as inputs of CNN structure respectively, such that the question context can be used to evaluate the softmax weights of the input of CNN. From the experiments, we observe that the two extensions vary on their contributions on the performance improvement according to different datasets. However, QA-LSTM/CNN with attention can outperform the baselines on both datasets.

4 INSURANCEQA EXPERIMENTS

Having described a number of models in the previous section, we evaluate the proposed approaches on the insurance domain dataset, InsuranceQA, provided by Feng et al. (2015). The InsuranceQA dataset provides a training set, a validation set, and two test sets. We do not see obvious categorical differentiation between two tests' questions. One can see the details of InsuranceQA data in (Feng

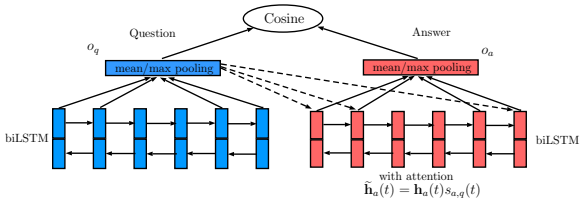


Figure 3: QA-LSTM with attention

	Validation	**Test1**	**Test2**
A. Bag-of-word	31.9	32.1	32.2
B. Metzler-Bendersky IR model	52.7	55.1	50.8
C. Architecture-II in (Feng et al., 2015)	61.8	62.8	59.2
D. Architecture-II with GESD	**65.4**	**65.3**	**61.0**

Table 2: Baseline results of InsuranceQA

et al., 2015). We list the numbers of questions and answers of the dataset in Table 1. A question may correspond to multiple answers. The questions are much shorter than answers. The average length of questions is 7, and the average length of answers is 94. The long answers comparing to the questions post challenges for answer selection task. This corpus contains 24981 unique answers in total. For the development and test sets, the dataset also includes an answer pool of 500 candidate answers for each question. These answer pools were constructed by including the correct answer(s) and randomly selecting candidate from the complete set of unique answers. The top-1 accuracy of the answer pool is reported.

4.1 SETUP

The models in this work are implemented with Theano (Bastien et al., 2012) from scratch, and all experiments are processed in a GPU cluster. Each model is trained for 140 epochs. All models are converged after 140 epochs. We use the accuracy on validation set to locate the best epoch and best hyper-parameter settings for testing.

The word embedding is trained by word2vec (Mikolov et al., 2013), and the word vector size is 100. Word embeddings are also parameters and are optimized as well during the training. Stochastic Gradient Descent (SGD) is the optimization strategy. The learning rate in our experiments is 0.1. We tried different margin values, such as 0.05, 0.1 and 0.15, and finally fixed the margin as 0.1. We also tried to include l_2 norm in the training objective. However, preliminary experiments show that regularization factors do not show any improvements. Also, the dimension of LSTM output vectors is 141 for one direction, such that biLSTM has a comparable number of parameters with a single-direction LSTM with 200 dimension.

We train our models in mini-batches (the batch size B is 100), and the maximum length L of questions and answers is 200. Any tokens out of this range will be discarded. Because the questions or answers within a mini-batch may have different lengths, we resort to a mask matrix $M \in R^{B \times L}$ to indicate the real length of each token sequence.

4.2 BASELINES

For comparison, we report the performances of four baselines in Table 2: two state-of-the-art non-DL approaches and two variations of a strong DL approach based on CNN as follows.

Bag-of-word: The idf-weighted sum of word vectors for the question and for all of its answer candidates is used as a feature vector. Similar to this work, the candidates are re-ranked according the cosine similarity to a question.

Metzler-Bendersky IR model: A state-of-the-art weighted dependency (WD) model, which employs a weighted combination of term-based and term proximity-based ranking features to score each candidate answer.

Architecture-II in (Feng et al., 2015): Instead of using LSTM, a CNN model is employed to learn a distributed vector representation of a given question and its answer candidates, and the answers are scored by cosine similarity with the question. No attention model is used in this baseline.

Architecture-II with Geometricmean of Euclidean and Sigmoid Dot product (GESD): GESD is used to measure the distance between the question and answers. This is the model which achieved the best performance in (Feng et al., 2015).

	Model	Validation	Test1	Test2
A	QA-LSTM basic-model(head/tail)	54.0	53.1	51.2
B	QA-LSTM basic-model(avg pooling)	58.5	58.2	54.0
C	QA-LSTM basic-model(max pooling)	64.3	63.1	58.0
D	QA-LSTM/CNN(fcount=1000)	65.5	65.9	62.3
E	QA-LSTM/CNN(fcount=2000)	64.8	66.8	62.6
F	QA-LSTM/CNN(fcount=4000)	66.2	64.6	62.2
G	QA-LSTM with attention (max pooling)	66.5	63.7	60.3
H	QA-LSTM with attention (avg pooling)	**68.4**	**68.1**	62.2
I	QA-LSTM/CNN (fcount=4000) with attention	67.2	65.7	**63.3**

Table 3: The experimental results of InsuranceQA for QA-LSTM, QA-LSTM/CNN and QA-LSTM with attentions

4.3 Results and discussions

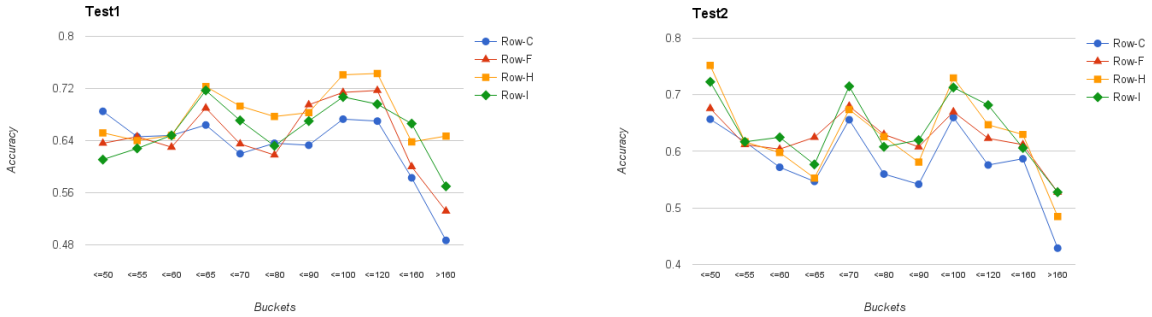
In this section, detailed analysis on experimental results are given. Table 3 summarizes the results of our models on InsuranceQA. From Row (A) to (C), we list QA-LSTM without either CNN structure or attention model. They vary on how to utilize the biLSTM output vectors to form sentential embeddings for questions and answers in shown in section 3.1. We can observe that just concatenating of the last vectors from both direction (A) performs the worst. It is surprised to see using max-pooling (C) is much better than average pooling (B). The potential reason is that the max-pooling extracts more local values for each dimension, so that more local information can be reflected on the output embeddings.

From Row (D) to (F), CNN layers are built on the top of the biLSTM with different filter numbers. We set the filter width $m = 2$, and we did not see better performance if we increase m to 3 or 4. Row (F) with 4000 filters gets the best validation accuracy, obtained a comparable performance with the best baseline (Row (D) in Table 2). Row F shared a highly analogous CNN structure with Architecture II in (Feng et al., 2015), except that the later used a shallow hidden layer to transform the word embeddings into the input of CNN structure, while Row F take the output of biLSTM as CNN input.

Row (G) and (H) corresponds to QA-LSTM with the attention model. (G) connects the output vectors of answers after attention with a max pooling layer, and (H) with an average pooling. In comparison to Model (C), Model (G) shows over 2% improvement on both validation and Test2 sets. With respect to the model with mean pooling layers (B), the improvement from attention is more remarkable. Model (H) is over 8% higher on all datasets compared to (B), and gets improvements from the best baseline by 3%, 2.8% and 1.2% on the validation, Test1 and Test2 sets, respectively. Compared to Architecture II in (Feng et al., 2015), which involved a large number of CNN filters, (H) model also has fewer parameters.

Row (I) corresponds to section 3.4, where CNN and attention mechanism are combined. Although compared to (F), it shows 1% improvement on all sets, we fail to see obvious improvements compared to Model (H). Although Model (I) achieves better number on Test2, but does not on validation and Test1. We assume that the effective attention might have vanished during the CNN operations. However, both (H) and (I) outperform all baselines.

We also investigate the proposed models on how they perform with respect to long answers. We divide the questions of Test1 and Test2 sets into eleven buckets, according to the average length of their ground truths. In the table of Figure 4, we list the bucket levels and the number of questions which belong to each bucket, for example, Test1 has 165 questions, whose average ground truth lengths are $55 < L \leq 60$. We select models of (C), (F), (H) and (I) in Table 3 for comparison. Model (C) is without attention and sentential embeddings are formed only by max pooling. Model (F) utilizes CNN, while model (H) and (I) integrate attention. As shown in the left figure in Figure 4, (C) gets better or close performance compared to other models on buckets with shorter answers (≤ 50, ≤ 55, ≤ 60). However, as the ground lengths increase, the gap between (C) and other models becomes more obvious. The similar phenomenon is also observed in the right figure for Test2. This suggests the effectiveness of the two extensions from the basic model of QA-LSTM, especially for long-answer questions.



Buckets	**≤50**	**≤55**	**≤60**	**≤65**	**≤70**	**≤80**	**≤90**	**≤100**	**≤120**	**≤160**	**>160**
Test1	121	167	165	152	137	223	161	147	191	180	156
Test2	105	191	192	168	169	230	153	115	170	165	142

Figure 4: The accuracy of Test1 and Test2 of InsuranceQA sets for the four models (C, H, F and I in Table 3), on different levels of ground truth answer lengths. The table divided each test set into 11 buckets. The figures above show the accuracy of each bucket.

Models	**MAP**	**MRR**
Wang et al. (2007)	0.6029	0.6852
Heilman & Smith (2010)	0.6091	0.6917
Wang & Manning (2010)	0.6029	0.6852
Yao et al. (2013)	0.6307	0.7477
Severyn & Moschitti (2013)	0.6781	0.7358
Yih et al. (2013)-BDT	0.6940	0.7894
Yih et al. (2013)-LCLR	0.7092	0.7700
Severyn & Moschitti (2015)	0.6709	0.7280
Wang & Nyberg (2015)	**0.7134**	0.7913
Architecture-II (Feng et al., 2015)	0.7106	**0.7998**

Table 4: Test results of baselines on TREC-QA

Feng et al. (2015) report that GESD outperforms cosine similarity in their models. However, the proposed models with GESD as similarity scores do not provide any improvement on accuracy.

Finally, we replace the cosine similarity with a MLP structure, whose input (282x2-dimension) is the concatenation of question and answer embeddings, and the output is a single similarity score and test the modified models by a variety of hidden layer size (100,500,1000). We observe that the modified models not only get >10% accuracy decrease, but also converge much slower. One possible explanation is the involvement of more network parameters by MLP makes it more difficult for training, although we believed that MLP might partially avoid the conceptual challenge of projecting questions and answers in the same high-dimensional space, introduced by cosine similarity.

5 TREC-QA EXPERIMENTS

In this section we detail our experimental setup and results using the TREC-QA dataset.

	Models	**MAP**	**MRR**
A	QA-LSTM (avg-pool)	68.19	76.52
B	QA-LSTM with attention	68.96	78.49
C	QA-LSTM/CNN	70.61	81.04
D	QA-LSTM/CNN with attention	**71.11**	**83.22**
E	QA-LSTM/CNN with attention (LSTM hiddenvector=500)	**72.79**	**82.40**

Table 5: Test results of the proposed models on TREC-QA

5.1 DATA, METRICS AND BASELINES

In this paper, we adopt TREC-QA, created by Wang et al. (2007) based on Text REtrieval Conference (TREC) QA track (8-13) data. We follow the exact approach of train/dev/test questions selection in (Wang & Nyberg, 2015), in which all questions with only positive or negative answers are removed. Finally, we have 1162 training questions, 65 development questions and 68 test questions.

Following previous work on this task, we use Mean Average Precision (MAP) and Mean Reciprocal Rank (MRR) as evaluation metrics, which are calculated using the official evaluation scripts.

In Table 4, we list the performance of some prior work on this dataset, which can be referred to (Wang & Nyberg, 2015). In order to make a fair comparison, we include the result of (Severyn & Moschitti, 2015) that uses word embeddings only [3]. We implemented the Architecture II in (Feng et al., 2015) from scratch. Wang & Nyberg (2015) and Feng et al. (2015) are the best baselines on MAP and MRR respectively.

5.2 SETUP

We keep the configurations same as those in InsuranceQA in section 4.1, except the following differences: First, since the training data size becomes smaller, we set the minibatch size as 10; Second, we set the maximum length of questions and answers as 40 instead of 200. Third, following (Wang & Nyberg, 2015), We use 300-dimensional vectors that were trained and provided by word2vec [4]. Finally, we use the models from the epoch with the best MAP on the validation set for training. Moreover, although TREC-QA dataset provided negative answer candidates for each training question, we randomly select the negative answers from all the candidate answers in the training set.

5.3 RESULTS

Table 5 shows the performance of the proposed models. Compared to Model (A), which is with average pooling on top of biLSTM but without attention, Model (B) with attention improves MAP by 0.7% and MRR by approximately 2%. The combination of CNN with QA-LSTM (Model-C) gives greater improvement on both MAP and MRR from Model (A). Model (D), which combines the ideas of Model (B) and (C), achieves the performance, competitive to the best baselines on MAP, and 2$\sim$4% improvement on MRR compared to (Wang & Nyberg, 2015) and (Feng et al., 2015). Finally, Model (E), which corresponds to the same model (D) but uses a LSTM hidden vector size of 500, achieves the best results for both metrics and outperforms the baselines.

6 CONCLUSION

In this paper, we study the answer selection task by employing a bidirectional-LSTM based deep learning framework. The proposed framework does not rely on feature engineering, linguistic tools or external resources, and can be applied to any domain. We further extended the basic framework on two directions. Firstly, we combine a convolutional neural network into this framework, in order to give more composite representations for questions and answers. Secondly, we integrate a simple but efficient attention mechanism in the generation of answer embeddings according to the question. Finally, two extensions combined together. We conduct experiments using the TREC-QA dataset and the recently published InsuranceQA dataset. Our experimental results demonstrate that the proposed models outperform a variety of strong baselines. In the future, we would like to further evaluate the proposed approaches for different tasks, such as answer quality prediction in Community QA and recognizing textual entailment. With respect to the structural perspective, we plan to generate the attention mechanism to phrasal or sentential levels.

[3] Severyn & Moschitti (2015) report 0.7460 (MAP) and 0.8080 (MRR) when using handcrafted features in addition to word embeddings.

[4] https://code.google.com/p/word2vec/

REFERENCES

Dzmitry Bahdanau, KyungHyun Cho, and Yoshua. Bengio. Neural machine translation by jointly learning to align and translate. *Proceedings of International conference of learning representations*, 2015.

Frederic Bastien, Pascal Lamblin, Razvan Pascanu, James Bergstra, Ian J. Goodfellow, Arnaud Bergeron, Nicolas Bouchard, and Yoshua Bengio. Theano: new features and speed improvements. Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop, 2012.

Cicero dos Santos, Luciano Barbosa, Dasha Bogdanova, and Bianca Zadrozny. Learning hybrid representations to retrieve semantically equivalent questions. In *Proceedings of ACL*, pp. 694–699, Beijing, China, July 2015.

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

Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. *In IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2013.

Michael Heilman and Noah A. Smith. Tree edit models for recognizing textual entailments, paraphrases, and answers to questions. *Annual Conference of the North American Chapter of the Association for Computational Linguistics. Association for Computational Linguistics (NAACL)*, 2010.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend. *In Advances in Neural Information Processing Systems (NIPS)*, 2015.

Sepp Hochreiter and Jurgen Schmidhuber. Long short-term memory. *Neural Computation*, 1997.

Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. Convolutional neural network architectures for matching natural language sentences. *Advances in Neural Information Processing Systems (NIPS)*, 2014.

Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. Learning deep structured semantic models for web search using clickthrough data. In *n Proceedings of the 22nd ACM international conference on Conference on information and knowledge management*, pp. 2333–2338, 2013.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. *Advances in Neural Information Processing Systems (NIPS)*, 2013.

Alexander Rush, Sumit Chopra, and Jason Weston. A neural attention model for sentence summarization. *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2015.

Aliaksei Severyn and Alessandro Moschitti. Automatic feature engineering for answer selection and extraction. *In Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2013.

Aliaksei Severyn and Alessandro Moschitti. Learning to rank short text pairs with convolutional deep neural networks. In *SIGIR*, 2015.

Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Gregoire Mesnil. Learning semantic representations using convolutional neural networks for web search. In *WWW*, 2014.

Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. End-to-end memory networks. *arXiv preprint arXiv:1503.08895*, 2015.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. *Advances in Neural Information Processing Systems*, 2014.

Duyu Tang, Bing Qin, and Ting Liu. Document modeling with gated recurrent neural network for sentiment classification. *In Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2015.

Oriol Vinyals and Quoc V. Le. A neural conversational model. *Proceedings of the 31st International Conference on Machine Learning*, 2015.

Di Wang and Eric Nyberg. A long short-term memory model for answer sentence selection in question answering. *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, 2015.

Mengqiu Wang and Christopher Manning. Probabilistic tree-edit models with structured latent variables for textual entailment and question answering. *The Proceedings of the 23rd International Conference on Computational Linguistics (COLING)*, 2010.

Mengqiu Wang, Noah Smith, and Mitamura Teruko. What is the jeopardy model? a quasi-synchronous grammar for qa. *The Proceedings of EMNLP-CoNLL*, 2007.

Jason Weston, Sumit Chopra, and Keith Adams. #tagspace: Semantic embeddings from hashtags. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014.

Xuchen Yao, Benjamin Durme, and Peter Clark. Answer extraction as sequence tagging with tree edit distance. *Proceedings of NAACL-HLT*, 2013.

Wen-tau Yih, Ming-Wei Chang, Christopher Meek, and Andrzej Pastusiak. Question answering using enhanced lexical semantic models. *Proceedings of the 51st Annual Meeting of the Association for Computational Linguist (ACL)*, 2013.

Lei Yu, Karl M. Hermann, Phil Blunsom, and Stephen Pulman. Deep learning for answer sentence selection. *NIPS Deep Learning Workshop*, 2014.