Frontiers of Information Technology & Electronic Engineering
www.zju.edu.cn/jzus; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

Attention-based encoder-decoder model for answer selection in question answering*

Yuan-ping NIE†[1], Yi HAN[2], Jiu-ming HUANG[1], Bo JIAO[3], Ai-ping LI[1]

([1]*College of Computer, National University of Defense Technology, Changsha 410073, China*)

([2]*Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China*)

([3]*Luoyang Electronic Equipment Test Center, Luoyang 471003, China*)

†E-mail: yuanpingnie@nudt.edu.cn

Received May 15, 2016; Revision accepted Oct. 15, 2016; Crosschecked Mar. 29, 2017

Abstract: One of the key challenges for question answering is to bridge the lexical gap between questions and answers because there may not be any matching word between them. Machine translation models have been shown to boost the performance of solving the lexical gap problem between question-answer pairs. In this paper, we introduce an attention-based deep learning model to address the answer selection task for question answering. The proposed model employs a bidirectional long short-term memory (LSTM) encoder-decoder, which has been demonstrated to be effective on machine translation tasks to bridge the lexical gap between questions and answers. Our model also uses a step attention mechanism which allows the question to focus on a certain part of the candidate answer. Finally, we evaluate our model using a benchmark dataset and the results show that our approach outperforms the existing approaches. Integrating our model significantly improves the performance of our question answering system in the TREC 2015 LiveQA task.

Key words: Question answering; Answer selection; Attention; Deep learning
http://dx.doi.org/10.1631/FITEE.1601232 **CLC number:** TP391.4

1 Introduction

Open-domain question answering is a classic task drawing upon many aspects of natural language processing (NLP), information retrieval (IR), and information extraction (IE). Recently, with the development of Web 2.0, Internet users are more willing to share their knowledge on the web. Some community question answering (CQA) services, such as Yahoo! Answers (http://answers.yahoo.com), Stack Overflow (http://www.stackoverflow.com), and Baidu Zhidao (http://zhidao.baidu.com), have millions of users and create a huge number of questions and answers. These question-answer (QA) pairs are usually generated explicitly by human beings. Thus, these QA pairs are a good resource when looking for answers for an open-domain question answering task.

The Text Retrieval Conference (TREC) is one of the most well-known on-going series of workshops focusing on a list of different IR tracks. In TREC 2015, a new question answering track called 'LiveQA' was created to solve real user questions. The test questions are all from Yahoo! Answers' non-answered questions of the day. Most participants choose to use Internet data for their answer resources, e.g., finding similar posted questions and the corresponding answers. However, the performances of the participating QA systems in TREC 2015 were far from satisfactory. One of the key reasons is that there is a lexical gap (also called a lexical chasm) between the

* Project supported by the National Basic Research Program (973) of China (Nos. 2013CB329601 and 2013CB329604) and the National Natural Science Foundation of China (Nos. 61372191, 61202362, and 61472433)

ORCID: Yuan-ping NIE, http://orcid.org/0000-0002-8351-4108

queries and the answers in the archives for the question answering retrieval tasks. For example, consider a case where the question is "Why is the word 'who' used only for humans but no other animals or creatures?" The answers may be 'because it implies the agency and therefore sentience' or 'its part of using the correct grammar'. Note that the question and answers do not have any matching key words. Actually, the lexical gap may lead the target answer to be easily regarded as an irrelevant one. This becomes a major barrier to prevent traditional IR models, e.g., BM25 (Robertson *et al.*, 1995), from matching a question with the candidate answers. Otherwise, candidate answers can be long and noisy. These answers contain redundant information, which leads the key information of answers to be surrounded by extraneous details.

To deal with the lexical gap, researchers have proposed to leverage QA pairs to learn translation models, to improve traditional IR models. The basic assumption is that question-answer pairs are 'parallel texts' and the relationships of words (or phrases) can be established through word-to-word (or phrase-to-phrase) translation probabilities (Jeon *et al.*, 2005; Zhou *et al.*, 2016). Berger *et al.* (2000) indicated that this 'lexical chasm' might be partially bridged by re-purposing statistical machine translation (SMT) models for QA. The experimental results showed that statistical translation models obtain state-of-the-art performance for question answering retrieval. However, in practice, question-answer pairs are far from 'parallel'. There are expensive training data requirements, and a large set of aligned question-answer pairs are needed for the training.

Recently, deep learning methods have been shown to be effective for many NLP tasks, such as sentence classification (Kim, 2014), machine translation (Bahdanau *et al.*, 2014; Cho *et al.*, 2014; Sutskever *et al.*, 2014), abstractive summarization (Rush *et al.*, 2015), and paraphrase detection (Iyyer *et al.*, 2014), due to the power of deep neural networks. As a part of deep neural network-based networks, the encoder-decoder model has been recently proposed to solve machine translation problems (Bahdanau *et al.*, 2014; Cho *et al.*, 2014; Sutskever *et al.*, 2014). The encoder-decoder usually contains two major parts: the encoder first encodes a query into a latent representation and then the decoder translates the latent vector into a target language. The machine translation technique can also be used to map questions and answers in QA tasks (Cui *et al.*, 2005; Riezler *et al.*, 2007). In other words, generating answers from a given question can be regarded as generating target texts given source texts.

In this paper, we describe approaches that attempt to address the answer selection task for question answering. There are two aspects to the investigation: (1) we use a deep neural network-based machine translation model to bridge the lexical gap between a question and its answer, and (2) an attention mechanism is employed to focus the system on the corresponding key information in the answers. In this paper, we first use a convolutional filter to obtain the semantic representation of the question and answer. Second, we use a bidirectional long short-term memory (LSTM) as the encoder. The encoder reads and encodes the question's representation into a fixed length vector. Then, we use another bidirectional LSTM as a decoder. The translation objective encourages the model to find sentence representations that capture their semantic features, because question-answer pairs with related semantic correlations are close to each other. To address the noise-bearing problem, we introduce a step attention model that focuses on which part of the candidate passage has the key information for the answer.

2 Related works

To solve the question answering selection problem, some previous works have paid attention to the syntactic matching between questions and answers. Punyakanok *et al.* (2004) proposed tree matching methods using tree-edit distance. Wang *et al.* (2007) employed quasi-synchronous grammar for matching QA pairs based on dependency trees. A tree kernel function with a logistic regression model was proposed by Heilman and Smith (2010). Recently, Severyn and Moschitti (2013) designed explicit feature vector representations, which require a substantial feature engineering effort to handle the question answer selection problem. The method relies on a kernel-based learning framework.

Some other researchers (Echihabi and Marcu, 2003; Soricut and Brill, 2006; Surdeanu *et al.*, 2011; Yao *et al.*, 2013b) have tried to learn various translation models to bridge the lexical gap between

questions and answers. In general, these models require some external knowledge, such as structured training data and language models. Echihabi and Marcu (2003) proposed a noisy-channel model for QA that can accommodate the exploitation of a large amount of resources and QA-specific techniques within a unified framework. Soricut and Brill (2006) described an answer ranking system for non-factoid questions built by a large community-generated QA collection. The authors investigated the feature types by exploiting NLP such as coarse word sense disambiguation, named-entity identification, syntactic parsing, and semantic role labeling.

Jeon *et al.* (2005) proposed word-based methods for question retrieval using the similarity between answers in the archive to estimate probabilities for a translation-based retrieval model. Xue *et al.* (2008) introduced a translation-based language model for the question part with a query likelihood approach for the answer part. Experiments showed that the proposed translation-based language model significantly outperforms traditional methods. Riezler *et al.* (2007) and Zhou *et al.* (2011) proposed phrase-based translation models for question and answer retrieval. The phrase-based translation model can capture more contextual information and its translation accuracy outperforms word-based translation models, which can improve the performance of retrieval models. Zhou *et al.* (2013) proposed an alternative method to address word ambiguity and word mismatch problems by enriching semantic information drawn from other languages.

Recently, deep learning methods have attracted considerable interest in NLP. In some previous investigations the performance of deep learning methods used in question answering has been discussed. Yu *et al.* (2014), Feng *et al.* (2015), dos Santos *et al.* (2015), and Severyn and Moschitti (2015) used certain similarity metrics for learning and matching the representations of questions and answers. Yu *et al.* (2014) proposed a convolutional neural network (CNN) to calculate the semantic similarity in answering single-relation factual questions. Severyn and Moschitti (2015) also employed a CNN architecture for reranking pairs of short texts. In a different manner, the model can learn an optimal representation of text pairs and a similarity function to relate them in a supervised way from the available training data. Yih *et al.* (2014) employed CNNs to calculate semantic similarity for answering single-relation factual questions. Wang and Nyberg (2015) employed a method that uses a stacked bidirectional LSTM (BLSTM) network to score the similarities of questions and answers. Finally, recently proposed models for textual generations can intrinsically be used for answer selections and generations (Bahdanau *et al.*, 2014; Sutskever *et al.*, 2014).

3 Approaches

3.1 Problem definition

In this section, we introduce our approaches for question answering tasks. Given a question Q, we first translate the question Q to a query q. Then we search for the candidate answers of the query through community question answering websites or search engines such as Google Search. The answer candidate pool is defined as $\{\boldsymbol{a}_1, \boldsymbol{a}_2, \cdots, \boldsymbol{a}_s\}$ for question Q ($\boldsymbol{a}_i$ is a candidate answer and s is the pool size). The goal is to find the best answer candidate $\boldsymbol{a}_k$ ($1 \leq k \leq s$). If the selected answer $\boldsymbol{a}_k$ is in the ground truth set of query q, query q is considered to be answered correctly; otherwise, it is answered incorrectly. From the definition, the task can also be regarded as an answer ranking problem.

In this study, we investigate an answer selection model and try to use a deep learning method to bridge the lexical gap between questions and answers. As mentioned above, the statistical machine translation (SMT) method has been demonstrated to be effective in question answering retrieval. Recently, as one of the deep learning methods, the encoder-decoder model has been widely used in machine translation tasks (Kalchbrenner and Blunsom, 2013; Bahdanau *et al.*, 2014; Cho *et al.*, 2014; Sutskever *et al.*, 2014). The structure of the encoder-decoder model is shown in Fig. 1. Let the input be a sequence of N words $(\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_N)$ coming from a fixed vocabulary V and the size is $|V|$. Each word is defined as an indicator vector in the set of possible inputs $\boldsymbol{X}$ ($\boldsymbol{x}_i \in \{0,1\}^V$ for $i \in \{1, 2, \cdots, N\}$), with sentences as a sequence of indicators.

The task of the encoder-decoder can be understood from the perspective of machine learning as learning the conditional distribution $p(a|q)$ of a target sentence (answer) a given a source sentence (question) q.

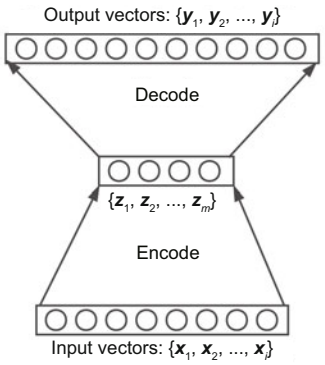

Fig. 1 The structure of an encoder-decoder

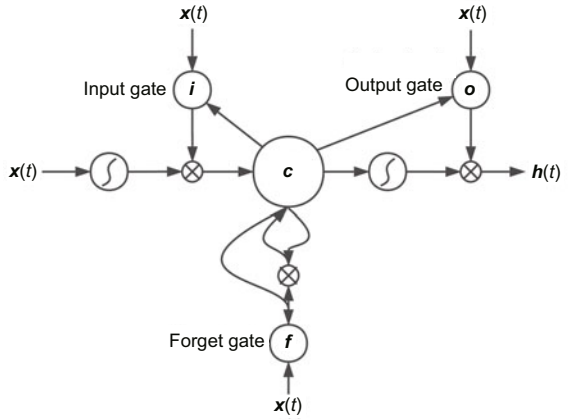

Fig. 2 The long short-term memory architecture

3.2 Recurrent neural networks

Recurrent neural networks (RNNs) are a class of neural networks that consist of a hidden state $\boldsymbol{h}$ and an optional output $\boldsymbol{y}$ which enables the networks to perform temporal processing and to learn a variable-length sequence $\boldsymbol{x}$, at each time step t. Let the input, hidden, and output weight matrices be $\boldsymbol{W}_{\mathrm{IH}}$, $\boldsymbol{W}_{\mathrm{HH}}$, and $\boldsymbol{W}_{\mathrm{OH}}$, respectively, and the hidden and output unit activation functions are f_{H} and f_{O}, respectively. The hidden state $\boldsymbol{h}(t)$ and output state $\boldsymbol{y}(t)$ of the RNN can be described as a dynamical system by the pair of non-linear matrix equations:

$$\begin{cases} \boldsymbol{h}(t) = f_{\mathrm{H}}(\boldsymbol{W}_{\mathrm{IH}}\boldsymbol{x}(t) + \boldsymbol{W}_{\mathrm{HH}}\boldsymbol{h}(t-1) + \boldsymbol{b}_{\mathrm{h}}), \\ \boldsymbol{y}(t) = f_{\mathrm{O}}(\boldsymbol{W}_{\mathrm{OH}}\boldsymbol{h}(t) + \boldsymbol{b}_{y}). \end{cases} \tag{1}$$

An RNN can learn a probability distribution over a sequence after being trained to predict the next symbol in a sequence. In that case, the output at each time step t is the conditional distribution $p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, \boldsymbol{x}_{t-2}, \cdots, \boldsymbol{x}_1)$.

3.3 Long short-term memory

Due to the gradient vanishing problem, conventional RNNs are found to be difficult to train to exploit long-range dependencies (Wang and Nyberg, 2015). However, LSTM (Hochreiter and Schmidhuber, 1997) is known to learn problems with long range temporal dependencies. In the LSTM architecture (Fig. 2), there are three gates: input $\boldsymbol{i}$ (Eq. (2)), forget $\boldsymbol{f}$ (Eq. (3)), output $\boldsymbol{o}$ (Eq. (4)), and a cell memory activation vector $\boldsymbol{c}$. Given an input vector $\boldsymbol{x}_t$ at time step t, the previous output $\boldsymbol{h}_{t-1}$, and cell state $\boldsymbol{c}_{t-1}$, an LSTM with a hidden size k computes the next output $\boldsymbol{h}_t$ and cell state $\boldsymbol{c}_t$ as

$$\boldsymbol{i}(t) = \sigma(\boldsymbol{W}_{\mathrm{i}}\boldsymbol{H} + \boldsymbol{b}_{\mathrm{i}}), \tag{2}$$

$$\boldsymbol{f}(t) = \sigma(\boldsymbol{W}_{\mathrm{f}}\boldsymbol{H} + \boldsymbol{b}_{\mathrm{f}}), \tag{3}$$

$$\boldsymbol{o}(t) = \sigma(\boldsymbol{W}_{\mathrm{o}}\boldsymbol{H} + \boldsymbol{b}_{\mathrm{o}}), \tag{4}$$

$$\boldsymbol{H} = (\boldsymbol{x}(t), \boldsymbol{h}(t-1))^{\mathrm{T}}, \tag{5}$$

$$\boldsymbol{c}_t = \boldsymbol{f}_t\boldsymbol{c}_{t-1} + \boldsymbol{i}_t\tanh(\boldsymbol{W}_{\mathrm{c}}\boldsymbol{H} + \boldsymbol{b}_{\mathrm{c}}), \tag{6}$$

$$\boldsymbol{h}_t = \boldsymbol{o}_t\tanh(\boldsymbol{c}_t), \tag{7}$$

where $\boldsymbol{W}_{\mathrm{i}}$, $\boldsymbol{W}_{\mathrm{f}}$, $\boldsymbol{W}_{\mathrm{o}}$, and $\boldsymbol{W}_{\mathrm{c}}$ are trained weighted matrices and $\boldsymbol{b}_{\mathrm{i}}$, $\boldsymbol{b}_{\mathrm{f}}$, $\boldsymbol{b}_{\mathrm{o}}$, and $\boldsymbol{b}_{\mathrm{c}}$ are biases that parameterize the gates and transformations of the input. A cell memory vector σ may be the sigmoid function. tanh is a kind of non-linear activation function and it may be replaced by other activation functions. An LSTM uses input and output gates to control the flow of information through the cell.

3.4 Bidirectional LSTM

As mentioned above, LSTM has only a forward pass. Every element is influenced only by the previous elements. In question answering tasks, the future content can also be useful to the previous words. In this study, we employ the bidirectional LSTM (Graves *et al.*, 2013) (Fig. 3). The bidirectional LSTM can efficiently make use of past words (via forward states) and future words (via backward states) for a specific time frame.

3.5 RNN encoder-decoder

Here, we describe briefly the underlying framework of an RNN encoder-decoder (Cho *et al.*, 2014; Sutskever *et al.*, 2014). An encoder-decoder contains two components:

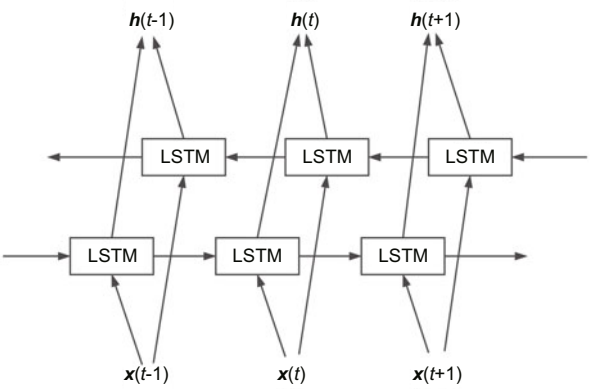

Fig. 3 Bidirectional LSTM structure

1. An encoder maps an input sequence $(\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_i)$ $(\boldsymbol{x}_i \in \mathbb{R}^m)$ into a fixed length latent representation $(\boldsymbol{r}_1, \boldsymbol{r}_2, \cdots, \boldsymbol{r}_l)$ $(\boldsymbol{r}_l \in \mathbb{R}^m)$ via a deterministic mapping function f_{e}. If the encoder-decoder employs RNNs, we can obtain

$$\boldsymbol{r}_l = f_{\mathrm{e}}(\boldsymbol{h}_1, \boldsymbol{h}_2, \cdots, \boldsymbol{h}_i),\ \boldsymbol{h}_t = q(\boldsymbol{x}_t, \boldsymbol{h}_{t-1}), \quad (8)$$

where $\boldsymbol{h}_t \in \mathbb{R}^n$ is a hidden state of RNN at time t and f_{e} and q are non-linear functions.

2. The decoder is often trained to predict the next word $\boldsymbol{y}_t$ given the latent representation $\boldsymbol{r}_l$ and all the previously predicted words $\boldsymbol{y}_1, \boldsymbol{y}_2, \cdots, \boldsymbol{y}_{t-1}$. The decoder outputs a conditional probability $\boldsymbol{y}$ by decomposing the joint probability into the ordered conditionals:

$$p(\boldsymbol{y}) = \prod_{t=1}^{T} p(\boldsymbol{y}_t | \{\boldsymbol{y}_1, \cdots, \boldsymbol{y}_{t-1}\}, \boldsymbol{c}), \quad (9)$$

$$\boldsymbol{r}_l = f_{\mathrm{e}}(\boldsymbol{x}_i) = S_{\mathrm{e}}(\boldsymbol{W}_{\mathrm{e}}\boldsymbol{x}_i + \boldsymbol{b}_{\mathrm{e}}), \quad (10)$$

where S_{e} is an activation function of the encoder, e.g., $\tanh(\cdot)$, $\boldsymbol{W}_{\mathrm{e}}$ is a weighted matrix, and $\boldsymbol{b}_{\mathrm{e}}$ is a bias vector. The decoder has a similar mechanism. The decoder translates the latent representation $\boldsymbol{r}_l$ back to an output sequence $\boldsymbol{y} = (\boldsymbol{y}_1, \boldsymbol{y}_2, \cdots, \boldsymbol{y}_j)$, and the translate function f_{d} is

$$\boldsymbol{y}_j = f_{\mathrm{d}}(\boldsymbol{r}_l) = S_{\mathrm{d}}(\boldsymbol{W}_{\mathrm{d}}\boldsymbol{r}_l + \boldsymbol{b}_{\mathrm{d}}). \quad (11)$$

With an RNN, each conditional probability is modeled as

$$p(\boldsymbol{y}_t | \{\boldsymbol{y}_1, \boldsymbol{y}_2, \cdots, \boldsymbol{y}_{t-1}\}, \boldsymbol{c}) = g(\boldsymbol{y}_{t-1}, \boldsymbol{s}_t, \boldsymbol{c}), \quad (12)$$

where g is a non-linear, potentially multi-layered function that outputs the probability of $\boldsymbol{y}_t$, and $\boldsymbol{s}_t$ is the hidden state of RNN.

3.6 Our model

There are two weak points in conventional RNNs. The first one is that it is difficult for RNNs to be trained to exploit long-range dependencies because of the gradient vanishing problem. Another weak point is the utilization of only previous context with no exploitation of future contexts. In this study, we use a bidirectional LSTM-based encoder-decoder (Fig. 4) to solve the answer selection task.

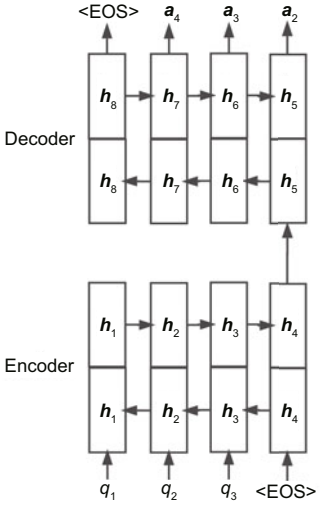

Fig. 4 Bidirectional LSTM based encoder-decoder

In contrast to the RNN encoder-decoder model, the bidirectional LSTM consists of forward and backward LSTMs. The forward LSTM $\overrightarrow{\boldsymbol{h}}$ reads a sequence and calculates a sequence of forward hidden states $(\overrightarrow{\boldsymbol{h}_1}, \overrightarrow{\boldsymbol{h}_2}, \cdots, \overrightarrow{\boldsymbol{h}_t})$ from $\boldsymbol{x}_1$ to $\boldsymbol{x}_t$. The backward LSTM reads the sequence in reverse order, from $\boldsymbol{x}_t$ to $\boldsymbol{x}_1$, and outputs a new sequence of backward hidden states $(\overleftarrow{\boldsymbol{h}_1}, \overleftarrow{\boldsymbol{h}_2}, \cdots, \overleftarrow{\boldsymbol{h}_t})$. Note that each word of the input query $\boldsymbol{x}_i$ is calculated jointly by the forward hidden state $\overrightarrow{\boldsymbol{h}_i}$ and the backward hidden state $\overleftarrow{\boldsymbol{h}_i}$. In this way, annotation $\boldsymbol{h}_i$ contains the summaries of both the preceding and the following words. Due to the tendency of RNNs, to better represent recent inputs, annotation $\boldsymbol{h}_i$ will be focused on the words around $\boldsymbol{x}_i$ (Bahdanau *et al.*, 2014).

3.6.1 Convolutional semantic filter

Most previous works employ single-word or word embedding (such as word2vec) as the model's input. These models can be regard as 'word level' semantic models. However, word-level models may

lead to semantic loss because some words have different meanings when those words are combined as phrases. In this study, instead of using every single word or individual sentence as an input, we try to use a 'phrase-level' semantic model. We employ a convolutional filter following the input layer to capture higher-level semantic concepts.

The input of our model is a query $\boldsymbol{x}$, which may have several sentences treated as a sequence of words $(\boldsymbol{w}_1, \boldsymbol{w}_2, \cdots, \boldsymbol{w}_s)$, where each word is drawn from a vocabulary V. Every word is represented by a continuous vector $\boldsymbol{w} \in \mathbb{R}^d$. Let $\boldsymbol{X} \in \mathbb{R}^{d\times s}$ be the representation matrix for the query $\boldsymbol{x}$. For each input query $\boldsymbol{x}$, there is a corresponding representation matrix $\boldsymbol{X} \in \mathbb{R}^{d\times s}$, of which the ith column represents the corresponding word's continuous representation $\boldsymbol{w}_i$. The convolutional semantic filter applies a transformation to the input query $\boldsymbol{x}$ using a convolution operation.

The convolution operation '$*$' between two vectors $\boldsymbol{a} \in \mathbb{R}^s$ and $\boldsymbol{f} \in \mathbb{R}^m$ is a filter of size m. The resulting vector $\boldsymbol{c}$ can be obtained as follows:

$$\boldsymbol{c}_i = (\boldsymbol{a} * \boldsymbol{f})_i = \sum_{k=i}^{i+m-1} \boldsymbol{a}_k \boldsymbol{f}_k. \tag{13}$$

The architecture of our convolutional semantic filter for mapping a query to feature vectors is shown in Fig. 5. The input query matrix is $\boldsymbol{X} \in \mathbb{R}^{d\times s}$ and a convolution filter is also a matrix $\boldsymbol{F} \in \mathbb{R}^{d\times m}$. As shown in Fig. 5, in each component the filter slides one element along the column dimension of input $\boldsymbol{X}$ and produces an output vector $\boldsymbol{c}$. In this study, we employ the wide type of convolution, which handles words better at boundaries by giving equal attention to all words in the query. We select the filter that is of the same dimensionality d as the input matrix.

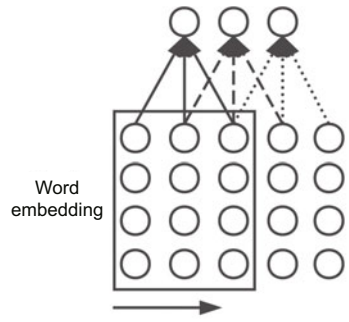

Fig. 5 Convolutional fitler

3.6.2 Attention-based encoder-decoder model

Attention-based neural networks have recently been demonstrated to be successful in a wide range of NLP tasks, such as machine translation (Bahdanau *et al.*, 2014), visual question answering (Xu *et al.*, 2015), and sentence summarization (Rush *et al.*, 2015). The idea of an attention mechanism is allowing the model to pay attention to the past output vectors (Fig. 6).

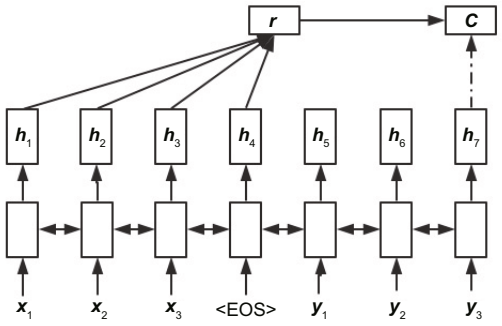

Fig. 6 The attention model

Bahdanau *et al.* (2014) and Rush *et al.* (2015) used similar attention models in various tasks. In contrast to other works, we first use a convolutional filter to obtain the phrase-level semantic representation of the question and answer. Then, in our case, the attention model can reread from the first bidirectional LSTM output vectors, while each word is read at the second bidirectional LSTM step by step. This can be regarded as a step attention model (Fig. 7). We denote the encoding outputs as $\boldsymbol{y}_{\mathrm{p}}(t)$, which are produced by the first LSTM when reading the q words of the question. Note that $\boldsymbol{y} \in \mathbb{R}^{q\times k}$ and k is a hyperparameter denoting the size of the embeddings. Furthermore, for the candidate passage, the composite output for each part at position t is $\boldsymbol{y}_d(t)$. The bidirectional embedding $\boldsymbol{y}_d(t) = \overrightarrow{\boldsymbol{y}_d}(t) \,||\, \overleftarrow{\boldsymbol{y}_d}(t)$ and the attention mechanism will produce a weighted representation $\boldsymbol{r}$ of the candidate passages and a vector $\boldsymbol{s}$ of the normalized attention weights via

$$\boldsymbol{m}(t) = \tanh(\boldsymbol{W}_{\mathrm{y}}\boldsymbol{y}_{\mathrm{p}} + \boldsymbol{W}_{\mathrm{r}}\boldsymbol{r}(t-1) + \boldsymbol{W}_{\mathrm{u}}\boldsymbol{y}_d(t)), \tag{14}$$

$$\boldsymbol{s}(t) = \mathrm{softmax}(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{m}(t)), \tag{15}$$

$$\boldsymbol{r}(t) = \boldsymbol{y}_d\boldsymbol{s} + \tanh(\boldsymbol{W}_{\mathrm{r}}\boldsymbol{r}(t-1)), \tag{16}$$

where $\boldsymbol{W}_{\mathrm{y}}$, $\boldsymbol{W}_{\mathrm{r}}$, and $\boldsymbol{W}_{\mathrm{u}}$ are trained projection matrices and $\boldsymbol{w}$ is a trained parameter vector and $\boldsymbol{w}^{\mathrm{T}}$

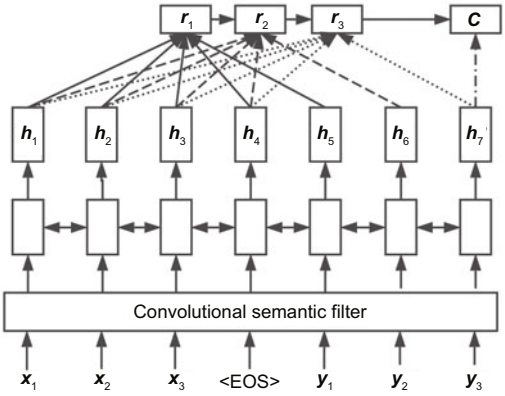

Fig. 7 The step attention model

denotes its transpose. $\boldsymbol{r}(t)$ is dependent on the previous attention representation $\boldsymbol{r}(t-1)$. The representation of the final question-answer pairs can be obtained from a non-linear combination of the weighted attention, given the attention score, the embedding of document $\boldsymbol{r}$, and the last output vector $\boldsymbol{h}_{\mathrm{n}}$ using

$$\boldsymbol{C} = \tanh(\boldsymbol{W}_{\mathrm{p}}\boldsymbol{r} + \boldsymbol{W}_{\mathrm{u}}\boldsymbol{h}_{\mathrm{n}}), \tag{17}$$

where $\boldsymbol{W}_{\mathrm{p}}$ and $\boldsymbol{W}_{\mathrm{u}}$ are trained projection matrices.

3.7 Training

In this study, we use the bidirectional LSTM encoder-decoder to match questions and answers for the answer selection task. In the experiments, we train a bidirectional LSTM encoder-decoder using many question and answer pairs. We train it by maximizing the logarithm probability of a correct answer A given the source question Q, so our training objective is

$$L = \frac{1}{|T|}\sum_{(Q,A)\in T}\log p(A|Q), \tag{18}$$

where T is the training dataset. After the training, we produce the answer by finding the most similar translated answer according to the model

$$A_{\mathrm{b}} = \arg\max_{A}\ p(A|Q). \tag{19}$$

In the training, we use a 10-fold cross-validation to test the performance of our model in the development dataset. Furthermore, to avoid overfitting, we add a dropout layer with the drop out ratio at 0.2. In addition, to obtain a better model, we adopt an AdaGrad optimizer to correct our model parameters. The word vector length is set to 300 and the batch size is 100.

4 Experiments

4.1 Experiment 1

4.1.1 Dataset

In the first experiment, we evaluate our model using one of the most widely used benchmark datasets in QA tasks from Wang *et al.* (2007). The questions are manually curated from the TREC QA tracks 8–13. All the questions in the dataset are short factoid questions and each question is associated with approximately 33 answer candidate sentences on average. In the dataset, if the answer sentence contains a question's related key information, the judgment is true.

There are two different training data sets (details are shown in Table 1). The Train-ALL dataset is a full training set containing 1229 questions labeled automatically by matching answer key regular expressions. The training dataset contains 94 questions, which are corrected manually for errors. Note that the Train-ALL dataset is very noisy and sometimes erroneously marks unrelated sentences as the correct answers but can provide significantly more question-answer pairs for training purposes.

Table 1 Summary of the TREC QA datasets for answer reranking

Dataset	Number of questions	Number of QA pairs	Correctness of dataset
Train-ALL	1229	53 417	12.0%
Train	94	4718	7.4%
Dev	82	1148	19.3%
Test	100	1517	18.7%

4.1.2 Evaluation metrics

We use the standard metrics mean average precision (MAP) and mean reciprocal rank (MRR). MRR measures the rank of any correct answer and MAP examines the ranks of all the correct answers. Specifically, MAP and MRR can be calculated by

$$\mathrm{MAP} = \frac{1}{n}\sum_{q=1}^{n}\mathrm{avg}(P(q)), \tag{20}$$

$$\mathrm{MRR} = \frac{1}{n}\sum_{q=1}^{n}\frac{1}{\mathrm{rank}(q)}, \tag{21}$$

where $\mathrm{avg}(P(q))$ is the average precision score of query q and $\mathrm{rank}(q)$ is the ranking position of the first

correct answer in the candidate answers. MRR can look at the rank of the first correct answer, whereas MAP examines the ranks of all the correct answers.

4.1.3 Results

We test several baseline methods for the comparative analysis. The first baseline method was proposed by Wang *et al.* (2007) and employs a generative probabilistic model with a quasi-synchronous grammar formulation. Wang and Manning (2010) improved this model by using a tree-edit conditional random field (CRF) model, which learns the latent alignment structure. Yao *et al.* (2013a) modified the design of the tree-edit distance (TED) model for QA ranking tasks in the TREC QA dataset. Yih *et al.* (2013) employed boosted decision trees (BDTs) and the learning constrained latent representation (LCLR), which is the algorithm for learning latent structures. Yu *et al.* (2014) proposed a model based on a convolutional neural network to learn question-answer intermediate representations. Wang and Nyberg (2015) employed a three-layer stacked bidirectional LSTM network and a joint model, which combines stacked BLSTM outputs with a keyword-matching baseline (BM25). We also compare our model with some other encoder-decoder models, such as the LSTM encoder-decoder (Sutskever *et al.*, 2014) and the RNN encoder-decoder (Cho *et al.*, 2014).

Table 2 shows the experimental results on the Train-ALL dataset. Our model performs better than the other methods. Note that the encoder-decoder model can solve the question reranking problem more effectively considering the baseline models' published results for this task. The output of the decoder is influenced by all the input encoders. It has an advantage in bridging the lexical gap between questions and answers. As shown in Table 2, the LSTM encoder-decoder model has a better performance than the RNN encoder-decoder. LSTM has a naturally stronger ability to learn long-range temporal dependency data for this QA task due to the considerable time lag between the questions and their corresponding key information in the answer sentences. It is also demonstrated that the bidirectional LSTM can match questions and answers more effectively since the bidirectional LSTM can use both previous and future contexts by processing the data from two directions. However, this model fails to match the performance of the attention-based model. The attention mechanism induces that the query can pay attention to key information in the answer and can neglect the irrelevant part of the answer, such as words that capture little meaning.

Table 2 The results using the TREC-QA dataset in Train-ALL

Model	MAP	MRR
Wang *et al.* (2007)	0.6029	0.6852
Wang and Manning (2010)	0.5951	0.6951
Yao *et al.* (2013a)	0.6307	0.7477
BDT (Yih *et al.*, 2013)	0.6940	0.7894
LCLR (Yih *et al.*, 2013)	0.7092	0.7700
Unigram (Yu *et al.*, 2014)	0.5387	0.6284
Bigram (Yu *et al.*, 2014)	0.5693	0.6613
Three-layer BLSTM (Wang and Nyberg, 2015)	0.5928	0.6721
BLSTM+BM25 (Wang and Nyberg, 2015)	0.7134	0.7913
RNN	0.6198	0.6831
LSTM	0.6428	0.7115
Our model without attention	**0.6871**	**0.7350**
Our model with step attention	**0.7261**	**0.8018**

4.2 Experiment 2

In experiment 2, we test our new answer selection model in the TREC 2015 LiveQA track.

In the TREC 2015 LiveQA track, there are 1087 questions provided by NIST. All the questions are collected from Yahoo! Answers' non-answered questions of the day. There is no specific answer set, and therefore we choose to search for the answer in Internet data using a search engine (Google Search), and specifically we search the posted similar questions in CQAs such as Yahoo! Answers. The answer quality is scored by manual annotations. The answer score is set to four levels, shown below:

1. Bad: containing no useful information for the question.

2. Fair: containing marginally useful information.

3. Good: partially answering the question.

4. Excellent: containing a significant amount of useful information, fully answering the question.

The evaluation index is an average score over all queries (transferring 1–4 level scores to 0–3, hence comparing the 1-level score with a no-answer score as 0).

In the TREC 2015 LiveQA track, we finally submit three runs using different strategies. Our best performing system achieves a 0.670 average score,

which is much better than the average score of all runs (0.465) and is ranked in third best position. Additionally, this average score is only 1% lower than that of the second best ranking system (Table 3). However, it still does not reach the state-of-the-art result, which is up to a 1.081 average score.

In this study, we employ our model for the LiveQA retrieval. We use a Yahoo! Answers training dataset, which contains nearly four million question-answer pairs. We select 10 000 questions and their answers as the training dataset (http://webscope.sandbox.yahoo.com). The testing question set is composed of the 1087 questions given by TREC. Our QA system searches the query in the Yahoo! Answer site and returns 20 candidate results. We employ our model to rerank the results and judge the answer manually, employing the same judgment mechanism. The results are shown in Table 3.

Table 3 The performance in the LiveQA track

Model	Average score	Succ@2+	Succ@4+
CMUOAQA	1.081	0.532	0.190
Ecnucs	0.677	0.367	0.086
Our pre-model	0.670	0.353	0.107
Monash	0.666	0.364	0.082
Yahoo	0.626	0.320	0.095
Average	0.467	0.262	0.060
Our model	**1.103**	**0.546**	**0.204**

Succ@i+ ($i = 1, 2, 3, 4$) means the number of questions with scores larger than i divided by the number of all the questions

As shown in Table 3, our new model can significantly improve the performance of our QA system in terms of all the indicators. The main score, i.e., the average score, is increased by over 50%. We analyze the main reasons as follows: this year, the LiveQA track questions are from real world and real users. These kinds of questions are non-factoid and the questions and answers are much noisier and harder to parse because the questions and answers are generated by real users. The lexical gap is more significant. In our previous work, we used external knowledge resources such as WordNet, which cannot cover all the requirements. The neural network-based encoder-decoder model learns the latent semantic features and semantic correlations effectively. Using an attention-based model also helps boost the performance of question and answer matching.

5 Conclusions

In this paper, we propose a model to address the answer selection problem for question answering tasks. The proposed model employs a bidirectional LSTM based encoder-decoder architecture, which can effectively bridge the lexical gap between questions and answers. Our approaches require no manual feature engineering or external resources, which may be expensive and not available. The long-length questions and passages created by users may be much noisier. To address this problem, we use a step attention model, which allows the questions to focus on a certain part of the candidate passage. We test our model with a widely used benchmark dataset—TREC Question Answering. The results show that our model is more effective compared to the baseline approaches. We also update our QA system using the new model and test it by repeating the TREC LiveQA evaluation. The results illustrate that the proposed model can significantly improve the performance of our QA system. In future investigations, we would like to evaluate the models for different tasks further and try to improve our model.

References

Bahdanau, D., Cho, K., Bengio, Y., 2014. Neural machine translation by jointly learning to align and translate. ArXiv:1409.0473.

Berger, A., Caruana, R., Cohn, D., *et al.*, 2000. Bridging the lexical chasm: statistical approaches to answer-finding. Proc. 23rd Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, p.192-199. http://dx.doi.org/10.1145/345508.345576

Cho, K., van Merriënboer, B., Gulcehre, C., *et al.*, 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. ArXiv:1406.1078.

Cui, H., Sun, R., Li, K., *et al.*, 2005. Question answering passage retrieval using dependency relations. Proc. 28th Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, p.400-407. http://dx.doi.org/10.1145/1076034.1076103

dos Santos, C., Barbosa, L., Bogdanova, D., *et al.*, 2015. Learning hybrid representations to retrieve semantically equivalent questions. Proc. 53rd Annual Meeting of the Association for Computational Linguistics and 7th Int. Joint Conf. on Natural Language Processing, p.694-699. http://dx.doi.org/10.3115/v1/P15-2114

Echihabi, A., Marcu, D., 2003. A noisy-channel approach to question answering. Proc. 41st Annual Meeting of the Association for Computational Linguistics, p.16-23. http://dx.doi.org/10.3115/1075096.1075099

Feng, M., Xiang, B., Glass, M.R., *et al.*, 2015. Applying deep learning to answer selection: a study and an open task. ArXiv:1508.01585.

Graves, A., Mohamed, A., Hinton, G.E., 2013. Speech recognition with deep recurrent neural networks. IEEE Int. Conf. on Acoustics, Speech and Signal Processing, p.6645-6649. http://dx.doi.org/10.1109/ICASSP.2013.6638947

Heilman, M., Smith, N.A., 2010. Tree edit models for recognizing textual entailments, paraphrases, and answers to questions. Human Language Technologies: Annual Conf. of the North American Chapter of the Association for Computational Linguistics, p.1011-1019.

Hochreiter, S., Schmidhuber, J., 1997. Long short-term memory. *Neur. Comput.*, **9**(8):1735-1780. http://dx.doi.org/10.1162/neco.1997.9.8.1735

Iyyer, M., Boyd-Graber, J.L., Claudino, L.M.B., *et al.*, 2014. A neural network for factoid question answering over paragraphs. Proc. Conf. on Empirical Methods in Natural Language Processing, p.633-644. http://dx.doi.org/10.3115/v1/D14-1070

Jeon, J., Croft, W.B., Lee, J.H., 2005. Finding similar questions in large question and answer archives. Proc. 14th ACM Int. Conf. on Information and Knowledge Management, p.84-90. http://dx.doi.org/10.1145/1099554.1099572

Kalchbrenner, N., Blunsom, P., 2013. Recurrent continuous translation models. Proc. Conf. on Empirical Methods in Natural Language Processing, p.1700-1709.

Kim, Y., 2014. Convolutional neural networks for sentence classification. ArXiv:1408.5882.

Punyakanok, V., Roth, D., Yih, W.T., 2004. Mapping dependencies trees: an application to question answering. Proc. 8th Int. Symp. on Artificial Intelligence and Mathematics, p.1-10.

Riezler, S., Vasserman, A., Tsochantaridis, I., *et al.*, 2007. Statistical machine translation for query expansion in answer retrieval. Annual Meeting of the Association for Computational Linguistics, p.464-471.

Robertson, S.E., Walker, S., Jones, S., *et al.*, 1995. Okapi at TREC-3. Overview of 3rd Text REtrieval Conf., p.109-126.

Rush, A.M., Chopra, S., Weston, J., 2015. A neural attention model for abstractive sentence summarization. ArXiv: 1509.00685.

Severyn, A., Moschitti, A., 2013. Automatic feature engineering for answer selection and extraction. Proc. Conf. on Empirical Methods in Natural Language Processing, p.458-467.

Severyn, A., Moschitti, A., 2015. Learning to rank short text pairs with convolutional deep neural networks. Proc. 38th Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, p.373-382. http://dx.doi.org/10.1145/2766462.2767738

Soricut, R., Brill, E., 2006. Automatic question answering using the web: beyond the factoid. *Inform. Retr.*, **9**(2):191-206. http://dx.doi.org/10.1007/s10791-006-7149-y

Surdeanu, M., Ciaramita, M., Zaragoza, H., 2011. Learning to rank answers to non-factoid questions from web collections. *Comput. Ling.*, **37**(2):351-383. http://dx.doi.org/10.1162/COLI_a_00051

Sutskever, I., Vinyals, O., Le, Q.V., 2014. Sequence to sequence learning with neural networks. Advances in Neural Information Processing Systems, p.3104-3112.

Wang, D., Nyberg, E., 2015. A long short-term memory model for answer sentence selection in question answering. Proc. 53rd Annual Meeting of the Association for Computational Linguistics and 7th Int. Joint Conf. on Natural Language Processing, p.707-712. http://dx.doi.org/10.3115/v1/P15-2116

Wang, M., Manning, C.D., 2010. Probabilistic tree-edit models with structured latent variables for textual entailment and question answering. Proc. 23rd Int. Conf. on Computational Linguistics, p.1164-1172.

Wang, M., Smith, N.A., Mitamura, T., 2007. What is the jeopardy model? A quasi-synchronous grammar for QA. Proc. Joint Conf. on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, p.22-32.

Xu, K., Ba, J., Kiros, R., *et al.*, 2015. Show, attend and tell: neural image caption generation with visual attention. ArXiv:1502.03044.

Xue, X., Jeon, J., Croft, W.B., 2008. Retrieval models for question and answer archives. Proc. 31st Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, p.475-482. http://dx.doi.org/10.1145/1390334.1390416

Yao, X., van Durme, B., Callison-Burch, C., *et al.*, 2013a. Answer extraction as sequence tagging with tree edit distance. Proc. NAACL-HLT, p.858-867.

Yao, X., van Durme, B., Callisonburch, C., *et al.*, 2013b. Semi-Markov phrase-based monolingual alignment. Proc. Conf. on Empirical Methods in Natural Language Processing, p.590-600.

Yih, W., Chang, M., Meek, C., *et al.*, 2013. Question answering using enhanced lexical semantic models. Proc. 51st Annual Meeting of the Association for Computational Linguistics, p.1744-1753.

Yih, W., He, X., Meek, C., 2014. Semantic parsing for single-relation question answering. Proc. 52nd Annual Meeting of the Association for Computational Linguistics, p.643-648. http://dx.doi.org/10.3115/v1/P14-2105

Yu, L., Hermann, K.M., Blunsom, P., *et al.*, 2014. Deep learning for answer sentence selection. ArXiv:1412.1632.

Zhou, G., Cai, L., Zhao, J., *et al.*, 2011. Phrase-based translation model for question retrieval in community question answer archives. Proc. 49th Annual Meeting of the Association for Computational Linguistics, p.653-662.

Zhou, G., Liu, F., Liu, Y., *et al.*, 2013. Statistical machine translation improves question retrieval in community question answering via matrix factorization. Proc. 51st Annual Meeting of the Association for Computational Linguistics, p.852-861.

Zhou, G., Zhou, Y., He, T., *et al.*, 2016. Learning semantic representation with neural networks for community question answering retrieval. *Knowl.-Based Syst.*, **93**:75-83. http://dx.doi.org/10.1016/j.knosys.2015.11.002