Multihop Attention Networks for Question Answer Matching

Nam Khanh Tran, Claudia Niederée
L3S Research Center, Leibniz Universität Hannover
Hannover, Germany
{ntran,niederee}@L3S.de

ABSTRACT

Attention based neural network models have been successfully applied in answer selection, which is an important subtask of question answering (QA). These models often represent a question by a single vector and find its corresponding matches by attending to candidate answers. However, questions and answers might be related to each other in complicated ways which cannot be captured by single-vector representations. In this paper, we propose Multihop Attention Networks (MAN) which aim to uncover these complex relations for ranking question and answer pairs. Unlike previous models, we do not collapse the question into a single vector, instead we use multiple vectors which focus on different parts of the question for its overall semantic representation and apply multiple steps of attention to learn representations for the candidate answers. For each attention step, in addition to common attention mechanisms, we adopt sequential attention which utilizes context information for computing context-aware attention weights. Via extensive experiments, we show that MAN outperforms state-of-the-art approaches on popular benchmark QA datasets. Empirical studies confirm the effectiveness of sequential attention over other attention mechanisms.

KEYWORDS

Answer selection, non-factoid QA, representation learning, attention mechanism

ACM Reference Format:
Nam Khanh Tran, Claudia Niederée. 2018. Multihop Attention Networks for Question Answer Matching. In *SIGIR '18: The 41st International ACM SIGIR Conference on Research and Development in Information Retrieval, July 8–12, 2018, Ann Arbor, MI, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3209978.3210009

1 INTRODUCTION

Answer selection (AS) is an important subtask of question answering (QA) that enables selecting the most suitable answer from a list of candidate answers in regard to the input question. One main challenge of this task lies in the complex and versatile semantic relations that can be observed between questions and answers. While for factoid QA the task of answer selection may be largely cast as a textual entailment problem, for non-factoid QA what makes an answer better than another often depends on many factors. Different from many other matching tasks, the linguistic similarities between questions and answers may or may not be indicative for the good answers; depending on what the question is looking for, a good answer may come in different forms. Sometimes a correct answer completes the question precisely with the missing information, and in other scenarios, good answers need to elaborate part of the question to rationalize it, and so on. In other cases, the best answers can also be noisy and include extraneous information irrelevant to the question. In addition, while a good answer must relate to the question, they might not share common lexical units. For example, in the question in Table 1, "*companies*" is not directly mentioned in the answer. This issue may confuse simple word-matching systems.

Table 1: An example of a question with an answer from the FiQA dataset. The segments in the answer are related to the segments in the question by the same color.

Question: Are companies in California obliged to provide invoices?
Ground-truth answer: We run into this all the time with our EU clients. As far as I can tell, the only requirements when it comes to invoicing have to do with sales tax, which is determined at the state level, and only in the case that items are taxable. It seems that the service provided to you is not taxable and so there is no obligation under Californian law to provide what you need.

These challenges consequently make the traditional models which are commonly based on lexical features [29, 30, 33] less effective compared to deep learning based methods [8, 28]. The neural models often follow the two step procedure: Firstly, representations of questions and answers are learned via a neural encoder such as long short-term memory (LSTM) networks or convolutional neural networks (CNN); Secondly, these representations of questions and answers are composed by an interaction function to produce an overall matching score. In the first step, each word in a question or an answer sequence is first represented with a hidden vector and then all the hidden vectors are aggregated for sequence representations. These models have shown very successful results in the AS task, however they still suffer from an important issue. The answers can be very long and contain lots of words that are not related to the question at hand, especially in non-factoid QA; consequently, the resulting representation might be distracted by non-useful information.

Recent years, attention-based models are proposed to deal with this challenge and have shown great success in many tasks such

SIGIR '18, July 8–12, 2018, Ann Arbor, MI, USA

ACM ISBN 978-1-4503-5657-2/18/07…$15.00
https://doi.org/10.1145/3209978.3210009

as machine translation [2, 23], machine reading comprehension [11] and textual entailments [19]. In the AS task, attention-based approaches aim to focus on segments within the candidate answer that are most related to the question [24, 27]. The segments with a stronger focus are treated as more important and have more influence on the resulting representations. For example, in attention-based LSTM models [24] as shown in Figure 2, a weight is automatically generated for each word in the answer via an attention model, and the answer is represented as the weighted sum of the hidden vectors. Various attention mechanisms have been proposed in previous studies in which additive attention [2] and multiplicative attention [20] are the two most commonly used. While additive attention is associated with a multi-layer perceptron for computing attention weights, multiplicative attention uses inner product for the weight estimations. Though these attention mechanisms have shown promising results in answer selection, they do not make use of surrounding word context when calculating attention weights, which has been proved to enhance the performance of LSTM based QA [5]. To address this issue, we adopt another attention mechanism, i.e. *sequential attention* [3] in which an additional LSTM is added to compute a context-aware weight for each hidden vector. This mechanism helps generate more accurate answer representation regarding to the question.

A common characteristic of the previous attention-based approaches is that the question is represented by one feature vector and a round of attention is applied for learning the representation of the answer. However, in many cases different segments of the answer can be related to different parts of the question. For example, in Table 1 the segment "*the only requirements when it comes to invoicing have to do with sales tax*" is relevant to "*provide invoices*" mentioned in the question, while "*there is no obligation under Californian law*" answers "*California obliged*" stated in the question. Consequently, using one feature vector pair for question answer matching may be not capable of capturing the complex semantic relations between questions and answers. This can lead to suboptimal results. Clearly, it is expected that the more aspects an answer covers the better the answer is. A good system should reflect this expectation.

In this paper, we propose Multihop Attention Networks (MANs) to deal with this problem. MANs locate, via multiple steps of attention, answer segments that are relevant to different aspects of the question. As illustrated in Figure 1(b), the MAN first uses the question vector to deduce the answer vector in the first attention layer, then the question vector is refined in the next step to learn the answer vector in the second attention layer. Each attention step gives a different attention distribution focusing on the segments that are relevant to one aspect of the question. Finally, we sum up the matching score in each step for scoring the answer. We perform experiments on both factoid QA and non-factoid QA datasets. Experimental results show that our proposed models obtain highly competitive results and outperform state-of-the-art approaches.

The main contributions of our paper can be summarized as follows:

- We are the first to investigate the effectiveness of sequential attention mechanism for answers' attentive representations in answer selection.
- We propose Multihop Attention Networks which represent questions by multiple vectors and use multiple steps of attention for learning the representation of answers. By doing this, MANs can capture different semantic relations between questions and answers.
- We provide extensive experimental evidence of the effectiveness of our model on both factoid question answering and community-based question answering on different domains. Our proposed approach outperforms many other neural architectures on these datasets.

2 RELATED WORK

Our work is concerned with ranking question and answer pairs to select the most relevant answer for each question. Previous work on this task have primarily used feature engineering, linguistic tools, or external resources [29, 30, 33]. In [33], Yih et al. constructed semantic features based on WordNet and paired semantically related words based on word semantic relations. In [29, 30], the answer selection problem was transformed to a syntactical matching between the question and answer parse trees. Some work tried to fulfill the matching using minimal edit sequences between dependency parse trees [9, 21, 32]. However, apart from relying on the availability of additional resources, the effort of feature engineering and the systematic complexity introduced by the linguistic tools, such models have limited performance and are outperformed by modern deep learning approaches [22, 35].

Yu et al. [35] employed a convolutional neural network (CNN) for feature learning of QA pairs and subsequently applied logistic regression for prediction. Despite its simplicity, the approach outperforms all traditional approaches [21, 32, 33]. Another attractive quality of deep learning architectures is that features can be learned in an end-to-end fashion. Severyn et al. [22] presented a unified architecture that trains a convolutional neural network together with a multi-layer perceptron, in which features are learned while the parameters of the network are optimized for the task at hand. In addition to CNN, recurrent neural networks such as the long short-term memory (LSTM) networks are also very popular for learning sequence representation and have been widely adopted in QA [24, 25, 28]. In [28], Wang and Nyberg incorporate stacked LSTMs to learn a joint feature vector of question and answer for classification. In [25], Tan et al. combined CNN and LSTM into a hybrid architecture which utilizes the advantages of both architectures. However, these approaches are outperformed by models with attention mechanism.

Recently, attention-based systems have shown very promising results on a variety of NLP tasks, such as machine translation [2, 23], machine reading comprehension [11], text summarization [20] and textual entailment [19]. Such models learn to focus their attention to specific parts of their input and most of them are based on a one-way attention, in which the attention is basically applied over one type of input based on another input (e.g. over target languages based on the source languages for machine translation, or over documents according to queries for reading comprehension). Most recently, several two-way attention mechanisms are proposed [7, 19, 34], where the information from two input items can influence the computation of each others representations. Both types of attention

show similar performances on AS [7, 25], thus in this paper we only use the one-way attention. However, unlike the previous work, our proposed models use multiple steps of attention instead of one attention step only.

Additive attention [2] and multiplicative attention [20] are the two most commonly used attention mechanisms. Self-attention or intra-attention is a special case of the additive attention mechanism. It relates elements at different positions from a single sequence by computing attention between each pair of tokens. In recent works, it has been shown effective in natural language inference [16], reading comprehension [13] and neural machine translation [26]. Sequential attention [3] is another type of attention mechanisms which uses an additional bi-directional RNN layer. This additional layer allows local alignment information to be used when computing the attentional score for each token. It has shown promissing results on reading comprehension [3]. In this paper, we show that sequential attention can be well adopted for the AS task and obtain highly competitive results.

Our proposed MANs are also related to Dynamic Memory Networks (DMNs) [14] in the sense that we both use an iterative attention process instead of only one round of attention. However, each memory in DMNs depends on the memory in the previous step, aiming to narrow down their focus on individual facts or sentences in regard to the single question representation, while in MANs each attention step is applied independently for selecting the informative parts of answers relating to different aspects of the question.

3 APPROACHES

Although CNNs can be used for representation learning in the AS task, LSTMs have been shown to obtain better performances [7, 24]. Hence, in this work we base our attention-based models on a variation of the LSTM model. We first describe the basic framework for answer selection based on LSTMs, called QA-LSTM [24]. Next we describe in detail different attention-based models that build on top of the QA-LSTM framework. After that, we present our Multihop Attention Networks.

3.1 Long Short-Term Memory (LSTM)

Long Short-Term Memory (LSTM) [12] networks are a type of recurrent neural network that are capable of learning long term dependencies across sequences. Given an input sequence $X = (x_1, x_2, ..., x_n)$, where each x_t is an E-dimension word vector ($x_t \in \mathbb{R}^E$), the LSTM returns a sequence embedding or hidden vector h_t with a size d ($h_t \in \mathbb{R}^d$) at every time step t defined as follows:

$$\begin{aligned} i_t &= \sigma\left(W_i x_t + U_i h_{t-1} + b_i\right) \\ o_t &= \sigma\left(W_o x_t + U_o h_{t-1} + b_o\right) \\ f_t &= \sigma\left(W_f x_t + U_f h_{t-1} + b_f\right) \\ u_t &= \tanh\left(W_u x_t + U_u h_{t-1} + b_u\right) \\ C_t &= i_t \odot u_t + f_t \odot C_{t-1} \\ h_t &= o_t \odot \tanh(C_t) \end{aligned} \tag{1}$$

where W_*, b_*, U_* are the parameters of the LSTM network ($W_* \in \mathbb{R}^{d \times E}$, $U_* \in \mathbb{R}^{d \times d}$, $b_* \in \mathbb{R}^d$) and $* = \{i, o, f, u\}$ in which the input i, forget f and output o are three gates, and C_t is the cell state. σ is the sigmoid function and $\odot$ denotes element-wise multiplication.

For the sake of brevity, we omit the technical details of LSTM which can be found in many related works.

Single-direction LSTMs suffer from the weakness of not making use of the contextual information provided by future tokens. Bidirectional LSTMs (biLSTMs) use both the previous and future context by processing the sequence in two directions, and generate two sequences of output vectors. The output for each token is the concatenation of the two vectors from both directions, i.e. $h_t = \overrightarrow{h_t} \parallel \overleftarrow{h_t}$. The output of biLSTM layer is a sequence of hidden vectors $H \in \mathbb{R}^{L \times 2d}$ where L is the maximum sequence length and d is the dimensional size of LSTM.

3.2 LSTM for Answer Selection

The basic LSTM-based framework for answer selection (QA-LSTM) [24] is shown as in Figure 1(c) (without the attention layer). Given an input pair (q, a), where $q = (q_1, ..., q_l)$ is a sequence of word indices for question and $a = (a_1, ..., a_m)$ is a sequence of word indices for candidate answer, the word embeddings (WEs) of both q and a are first retrieved by passing the sequences of word indices through a look-up layer. The parameters of this layer are $W \in \mathbb{R}^{V \times E}$ where V is the size of vocabulary and E is the dimensionality of the word embeddings. We initialize W with pretrained word embeddings which is inline with previous works [24, 35].

A biLSTM is then applied separately over the two sequences of WEs creating hidden vectors for the question and answer, i.e. $h_q(t) = \text{LSTM}(\overrightarrow{h_q}(t-1), q_t) \parallel \text{LSTM}(\overleftarrow{h_q}(t+1), q_t)$ and $h_a(t) = \text{LSTM}(\overrightarrow{h_a}(t-1), a_t) \parallel \text{LSTM}(\overleftarrow{h_a}(t+1), a_t)$. Subsequently, the final representations o_q and o_a for question and answer, respectively, can be taken by max or mean pooling over all the hidden vectors or the last hidden vector. As discussed in [8, 25], sharing the same network parameters is significantly better than using separate question and answer parameters, and converges much faster. Therefore, we follow the same procedure by using the same network parameters for processing questions and candidate answers.

Finally, a cosine similarity $sim(q, a)$ is defined to score the input pair (q, a) and the hinge loss function is used as training objective.

$$\mathcal{L} = \max\{0, M - sim(q, a_+) + sim(q, a_-)\} \tag{2}$$

where a_+ is a ground truth answer, a_- is an incorrect answer randomly chosen from the entire answer space, and M is the margin which controls the extent of discrimination between positive QA pairs and corrupted QA pairs. We treat any question with more than one ground truth as multiple training examples. During training, for each question we randomly sample N negative answers, but only use the one with the highest $\mathcal{L}$ to update the model similar to [18, 24].

3.3 Attention Mechanisms

The aforementioned QA-LSTM is basically a siamese network [6] which might fail to notice a potential issue. The answers might be extremely long and contain lots of words that are not related to the question at hand, especially in non-factoid question answering. For example, in Table 1 the first sentence "*we run into this all the time with our EU clients*" does not relate directly to the question. Even if advanced neural networks are exploited, the resulting representation might still be distracted by non-useful information. Thus,

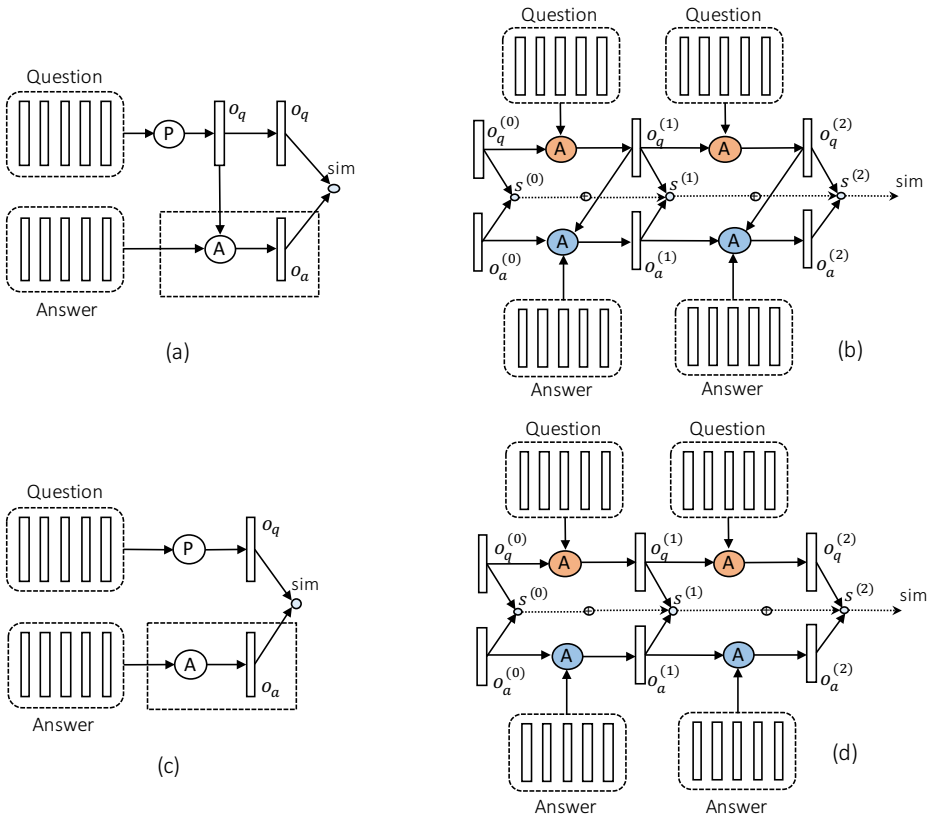

Figure 1: Traditional attention-based networks: (a) Interactive attention network; (c) Self-attention network; and our proposed MANs: (b) Multihop interactive attention network; (d) Multihop self-attention network. P: pooling layer, A: attention layer

a number of attention-based models for the answer vector generation have been proposed in order to alleviate this weakness by dynamically aligning the more informative parts of answers to the question. Conceptually, attention mechanisms give more weight to certain words which have more influence on the resulting representation. In the AS task the expectation is that words in the candidate answer that are more important with regard to the input question should receive larger weights. Most previous works such as [18, 24] proceed as follows: the input question is represented by a vector o_q using last, max or average pooling and an attention model is used over a sequence of hidden vectors to learn the representation of the answer o_a as shown in Figure 2. An attention mechanism which takes into account the question vector for computing the attention weights in learning the answer representation, is called as an interactive attention mechanism. In contrast, an attention mechanism which is employed only on the candidate answer, is considered as a self-attention or intra-attention mechanism. One of the advantages of the intra-attention mechanism is that questions and answers can be embedded into a joint vector space without being paired, so that arbitrary question and answer vectors in that space are directly comparable.

Let $H_a = \{h_a(1), h_a(2), ..., h_a(m)\}$ denote the hidden vectors of the answer after passing through the biLSTM layer. To produce the final representation of the answer, instead of using the last hidden

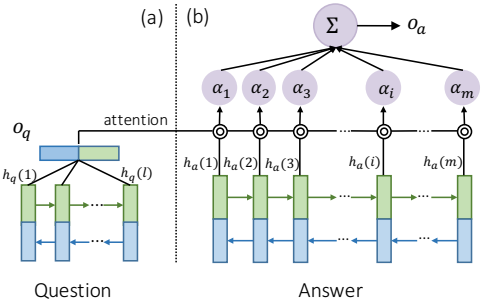

Figure 2: (a) The question vector representation and (b) The attention mechanism for answer vector generation

vector or average or max pooling, an additional attention layer is used as follows:

$$\alpha_t \propto f_{attention}\left(\tilde{o}_f, h_a(t)\right)$$
$$o_a = \sum_t \alpha_t h_a(t) \tag{3}$$

where $h_a(t)$ is the hidden vector of the answer at time t. When the interactive attention mechanism is used, $\tilde{o}_f$ is often equal to the question representation o_q as shown in Figure 1(a). When the

intra-attention mechanism is used, $f_{attention}$ only depends on $h_a(t)$ as in Figure 1(c). Next, we describe in detail the different implementations of $f_{attention}$ function.

MLP Attention: Additive attention (or multi-layer perceptron attention) [2] is one of the most commonly used attention mechanisms. It is first used for answer selection by Tan et al. [24]. In [24], the attention function $f_{attention}$ is computed by a multi-layer perceptron network as follows:

$$\begin{aligned} m(t) &= \tanh\left(W_a h_a(t) + W_q o_q\right) \\ f_{attention}\left(\tilde{o}_f, h_a(t)\right) &= \text{softmax}\left(w_m^T m(t)\right) \end{aligned} \tag{4}$$

where W_a, W_q are attentive weight matrices and w_m are attentive weight vector. The size of the matrices and vector are often equal to the size of input vectors $h_a(t)$ and o_q.

Bilinear Attention: This is another commonly used attention mechanism. For example, Chen et al. [4] found it effective in machine reading comprehension, Rush et al. [20] used it in abstractive summarization. Santos et al. [7] used this attention mechanism for AS task. In contrast to the additive attention, this attention mechanism makes use of a bilinear term instead of using a tanh layer to estimate the attention function $f_{attention}$:

$$f_{attention}\left(\tilde{o}_f, h_a(t)\right) = \text{softmax}_t\left(o_q^T W_s h_a(t)\right) \tag{5}$$

where W_s is a network parameter.

Sequential Attention: The previous approaches to attention select words with only very indirect consideration of their context, Brarda et al. [3] address this issue by taking into account explicit context sensitivity for computing the attention scoring function $f_{attention}$. Specifically, instead of producing a single value of $f_{attention}$ for each word in the answer by using a bilinear term as the bilinear attention, a vector γ_t is defined as

$$\gamma_t = o_q \odot h_a(t) \tag{6}$$

where $\odot$ is element-wise multiplication. The vector γ_t is then fed into a new biLSTM layer to get the hidden attention η_t vector representation: $\eta_t = \text{LSTM}(\overrightarrow{\eta}_{t-1}, \gamma_t) \parallel \text{LSTM}(\overleftarrow{\eta}_{t+1}, \gamma_t)$. Finally, the attention function $f_{attention}$ is computed as

$$f_{attention}\left(\tilde{o}_f, h_a(t)\right) = \text{softmax}_t\left(1^T \eta_t\right) \tag{7}$$

Chen et al. [5] showed that utilizing context information can enhance the performance of LSTM based QA. Therefore, in this paper we aim to investigate the effectiveness of sequential attention in answer selection. Experimental results show that sequential attention can be well adapted for the AS task and outperform other attention mechanisms on different QA datasets.

Self-Attention: In contrast to aforementioned attention mechanisms where the question vector o_q is used to learn the representation of the answer, in this attention mechanism the answer is autonomously embedded into the embedding space without being paired with the question, so that arbitrary question and answer vectors in the space are directly comparable. Generally, self-attention relates different positions of a single sequence in order to compute the final representation of the sequence. This has been successfully employed in a variety of tasks including reading comprehension, abstractive summarization and textual entailment [15]. In the context of answer selection, $f_{attention}\left(\tilde{o}_f, h_a(t)\right)$ can be estimated merely based on $h_a(t)$ as follows:

$$\begin{aligned} s(t) &= \tanh\left(W_s h_a(t) + b_s\right) \\ f_{attention}\left(\tilde{o}_f, h_a(t)\right) &= \text{softmax}\left(w_s^T s(t)\right) \end{aligned} \tag{8}$$

where W_s, b_s and w_s are attention parameters.

3.4 Multihop Attention Networks

In many cases, the semantic relations between a question and an answer can be very complex. Different parts of the answer can relate to different aspects addressed by the question. For example, in Table 1, the question *"are companies in California obliged to provide invoices"* refers to two aspects *invoices* and *California law* and each aspect is covered by different parts of the answer. Consequently, using single vectors for the question and answer representations might not be able to uncover their complex semantic relations. This can lead to suboptimal results.

In this paper, we propose Multihop Attention Networks (MANs) to tackle this problem. Our models are represented in Figure 1(b) and 1(d). Unlike existing models [7, 24], we do not compress the question to a single representation, but instead use multiple vectors for the question representation. Each question vector is then used to match with the answer representation which is learned via an attention layer. Specifically, given a question and a candidate answer, the model first reads the question and the answer using a biLSTM layer. Then, it deploys an iterative matching process to uncover the semantic relations between the question and the answer. In this phase, it first attends to some parts of the question, then finds their corresponding matches by attending to the answer. In the next step, it gives more attention to other parts of the question and searches for their matching parts in the answer. After a fixed number of iterations, the model uses the sum of the matching scores of each step to rank the question-answer pair.

Let $H_q = \{h_q(1), ..., h_q(l)\}$ denote the hidden vectors of the question after passing through the biLSTM layer. To obtain the representation of the question, one of three following mechanisms is usually used: last, mean, or max pooling. Last pooling takes the last vector $h_q(l)$, mean pooling averages all vectors and max pooling takes the element-wise maximum of H_q. In [15], Lin et al. proposed self-attention mechanism to replace the max pooling or averaging step. In this paper, we adapt this mechanism with some modifications for creating different question vector representations. Specifically, in each step k, the question representation $o_q^{(k)}$ is computed as follows:

$$\begin{aligned} s_t^{(k)} &= \tanh\left(W_q^{(k)} h_q(t)\right) \odot \tanh\left(W_m^{(k)} m_q^{(k-1)}\right) \\ \alpha_t^{(k)} &= \text{softmax}\left(w_s^{(k)^T} s_t^{(k)}\right) \\ o_q^{(k)} &= \sum_t \alpha_t^{(k)} h_q(t) \end{aligned} \tag{9}$$

where $W_q^{(k)}, W_m^{(k)}$ and $w_s^{(k)}$ are network parameters, $m_q^{(k)}$ is a separate memory vector for guiding the next attention step. It is recursively updated by

$$m_q^{(k)} = m_q^{(k-1)} + o_q^{(k)} \tag{10}$$

The initial memory vector $m_q^{(0)}$ is defined based on the context vector $o_q^{(0)}$ where

$$o_q^{(0)} = \frac{1}{l}\sum_t h_q(t)$$

In each step, the question is represented by a vector $o_q^{(k)}$ which focuses specifically on some aspects of the question. The vector $o_q^{(k)}$ is then used as input for the attention models described in the previous section to extract the answer representation $o_a^{(k)}$. After that, we compute the similarity between question and answer vectors by their cosine similarity. This similarity score reflects on how the answer relates to the corresponding parts of the question. After performing K matching steps, the final similarity between the given question and answer becomes

$$sim(q,a) = \sum_k^K \cos(o_q^{(k)}, o_a^{(k)}) \tag{11}$$

The overall architecture of this model when $K = 2$ is shown in Figure 1(b). Figure 1(d) presents MAN with using self-attention mechanism for the answer vector generation. In this case, we use two separate attention models described in Equation 9, one model for questions and another for answers. After each step, the same procedure is applied as implied by Equation 11. It is important to note that separate attention models are applied to questions and answers but the same attention parameters are used for questions (answers) in different steps. Therefore, our network parameters are comparable to the models with a single attention layer [7, 24] but we outperform the latter models on the datasets tested.

4 EXPERIMENTAL SETUP

4.1 Datasets

To evaluate the proposed approaches, we conduct an empirical evaluation based on three popular and well-studied benchmark datasets for both factoid and non-factoid question answering. In addition, we use another newly released dataset for the financial domain. These four datasets cover different domains and exhibit different characteristics:

- **TREC-QA** - This is a benchmark dataset created by Wang et al. [30] based on Text REtrieval Conference (TREC) QA track (8-13) data. The dataset contains a set of factoid questions, where candidate answers are limited to a single sentence. To enable direct comparison with the previous work, we follow the approach of train/dev/test questions selection from [28], in which all questions with only positive or negative answers are removed. In total, we have 1162 training questions, 65 development questions and 68 test questions. The maximum number of tokens for questions and answers are set to 11 and 60, respectively, the length of the vocabulary $|V|$=55060 and for each question there are 38 candidate answers on average.
- **WikiQA** - This is a recent popular benchmark dataset for open-domain question answering, based on factual questions from Wikipedia and Bing search logs. For each question, Yang et al. [31] selected Wikipedia pages and used sentences in the summary paragraph as candidates, which are then annotated on a crowdsourcing platform. We follow the same preprocessing steps as Yang et al., where questions with no correct candidate answers are excluded and answer sentences are truncated to 40 tokens. In total, we end up with 873 training questions, 126 development questions and 243 test questions. Since there are only few negative answers for each question in WikiQA, we extend it by randomly selecting a bunch of negative candidates from the answer pool.
- **InsuranceQA** - This is a recently released large-scale non-factoid QA dataset from the insurance domain created by Feng et al. [8]. In this work we use the first version of the dataset. The dataset is already divided into a training set, a validation set, and two test sets, in which a question may have multiple correct answers and normally the questions are much shorter than the answers. The average length of questions and answers in tokens are 7 and 95, respectively. Such difference imposes additional challenges for the AS task. For each question in the development and test sets, there is a set of 500 candidate answers, which include the ground-truth answers and randomly selected negative answers.
- **FiQA** - This is a new non-factoid QA dataset from the financial domain which has been recently released for WWW 2018 Challenges.[1] The dataset is built by crawling Stackexchange, Reddit and StockTwits in which part of the questions are opinionated, targeting mined opinions and their respective entities, aspects, sentiment polarity and opinion holder. We minimally preprocess the data only performing tokenization and lowercasing all words. To reduce the size of resulting vocabulary, we remove all rare words which occur less than 5 times. In this dataset questions and answers are longer than in other datasets, which will consequently bring extra challenges. The maximum number of tokens for questions and answers are set to 20 and 150 respectively. Following the setup for other datasets, we split this dataset into training, development and test sets as shown in Table 2. For each question in the development and test sets, we construct the answer pools by including the correct answer(s) and randomly selected candidates from the complete set of unique answers, as similar to InsuranceQA.

Table 2 presents some statistics about the datasets, including the number of questions in each set, average length of questions and answers as well as average number of candidate answers in the development and test sets.

4.2 Employed Baselines

In this section, we introduce the baselines used for comparison. For all datasets, we report performances of the basic matching model QA-LSTM and the models with a single attention layer described in Section 3.3 including MLP-LSTM, Bilinear-LSTM, Self-LSTM and Sequential-LSTM. Furthermore, we also introduce other baselines for each dataset separately.

- **TREC-QA** - The key competitors of this dataset are the CNN model of Severyn and Moschitti [22], the Attention-based Neural Matching model [7, 24] and the RNN with Positional Attention proposed by Chen et al. [5]. In addition, due to the

[1] https://sites.google.com/view/fiqa/home

Table 2: The statistics of the four employed answer selection datasets. For WikiQA and TREC-QA we remove all questions that have no right or wrong answers.

Dataset	TREC-QA	WikiQA	InsuranceQA	FiQA
# of questions (train/dev/test)	1162/65/68	873/126/243	12887/1000/1800x2	5999/323/324
Avg length of questions	8	6	7	11
Avg length of answers	28	25	95	135
Avg # of candidate answers	38	9	500	500

long standing nature of this dataset, we also report works based on traditional feature engineering approaches [10, 30].

- **WikiQA** - The competitors of this dataset include the Paragraph Vector (PV) and PV + Cnt models [33], CNN + Cnt model [35] which are reported in the original WikiQA paper [31]. Furthermore, we report additional strong baselines including AP-CNN and AP-LSTM [7], ABCNN [34] and RNN-POA [5]. We also report the Pairwise Ranking MP-CNN model [18].
- **InsuranceQA** - The key competitors of this dataset are the CNN-based ARC-I/II architecture by Feng et al. [8], QA-LSTM from [24] along with AP-LSTM which are attentive pooling improvements of the former and Inner attention-based RNN [27].
- **FiQA** - For this dataset, we reimplemented QA-LSTM [24] and different attention mechanisms on top of QA-LSTM for comparison.

We denote our proposed models as *Multihop-MLP-LSTM*, *Multihop-Bilinear-LSTM*, *Multihop-Sequential-LSTM* and *Multihop-Self-LSTM* which are MANs based on additive attention, bilinear attention, sequential attention and self-attention, respectively, for learning answers' attentive representations.

4.3 Hyperparameters and Training

This section describes the key evaluation protocol and metrics as well as implementation details of our experiments.

4.3.1 Evaluation Metrics. For the evaluation protocols we follow the prior work. Specifically, in TREC-QA and WikiQA we use the Mean Reciprocal Rank (MRR) and Mean Average Precision (MAP) metrics which are commonplace in IR and QA research. On the other hand, InsuranceQA and FiQA evaluate on Precision@1 (P@1) which is determined based on whether the top predicted answer is the ground truth. For all competitor methods, we report the performance results from the original paper.

4.3.2 Implementation Details and Hyperparameters. The models are implemented in Pytorch. The model parameters are optimized using Adam [1] optimizer with a learning rate of 0.001. A batch size of 100 are used for all datasets. The parameters are regularized with a per-minibatch L2 regularization strength of 10^{-5} and a dropout of $d = 0.3$ is also applied to prevent overfitting. The hidden layer size of LSTM models are the same as in previous works for a fair comparison. Specifically, for TREC-QA and InsuranceQA the sizes are set to 300 and 141 respectively as in [24]; the number for WikiQA is 141 as in [7]. For FiQA, we tried different numbers and found out that the size of 512 yields the best results. We tried different margins M in the hinge loss function and finally fixed the margin to $M = 0.2$. A number of negative answers $N = 50$ was used during training. The number of attention steps K is tuned amongst $\{1, 2, 3\}$ and we also experimented with the set of three vectors by using last, max and average pooling as different representations for the questions. We initialized the word embeddings with 300-dimensional Glove vectors [17] trained on 840 billion words. Embeddings for words not present in the Glove vectors are randomly initialized with each component sampled from the uniform distribution over $[-0.25, 0.25]$. The word embeddings are also part of the parameters and are optimized during training. Since sequences within a mini-batch have different lengths, we use a mask matrix to indicate the real length of each sequence. We trained all models for a maximum of 40 epochs. We take MAP scores for TREC-QA and WikiQA and P@1 scores for InsuranceQA and FiQA on the development set at every epoch and save the parameters of the network for the top three models. We report the best test score from the saved models. All experiments were conducted on a Linux machine with Nvidia GTX Ti 1080 GPU (12GB RAM). The code to reproduce the reported results and FiQA splits are publicly available at https://github.com/namkhanhtran/nn4nqa.

5 EXPERIMENTAL RESULTS

In this section, we present our empirical results on all datasets. For all reported results the best result is in boldface.

5.1 TREC-QA

Our results on TREC-QA dataset is summarized in Table 3. Firstly, we observe that all attention-based models outperform the basic matching model QA-LSTM by large margins. Second, the model based on sequential attention mechanism obtains a clear performance gain of around 3% on MAP/MRR against the model with additive or multiplicative attention mechanism. Compared to these models, Self-LSTM achieves comparable results though it does not use any query information for extracting answer representation. It also performs better than QA-LSTM, which indicates that self-attention mechanism can give better representations than simply max or average pooling method.

Furthermore, Table 3 shows that MANs outperform all models with only one attention layer. When using the same attention mechanism, the averages increase over the baselines with a single attention layer are 2% – 3% in terms of MAP/MRR. Specifically, Multihop-Sequential-LSTM gains an improvement of 1.6% on MAP and 2.8% on MRR compared to Sequential-LSTM. Similarly, Multihop-Bilinear-LSTM obtains 3.3% and 3.2% improvements in terms of MAP and MRR respectively. Multihop-MLP-LSTM also

Table 3: Experimental results on TREC-QA. Baselines for TREC-QA are reported in the first group. The second group shows the performance of models with a single attention layer. We report the performance of MANs in the last group.

Model	MAP	MRR
Wang et al. [30]	0.603	0.685
Heilman & Smith [10]	0.609	0.692
Wang & Nyberg [28]	0.713	0.791
CNN (Severyn & Moschitti) [22]	0.746	0.808
AP-LSTM (Tan et al.) [24]	0.753	0.830
AP-CNN (Santos et al.) [7]	0.753	0.851
RNN-POA (Chen et al. [5])	0.781	0.851
QA-LSTM	0.737	0.810
MLP-LSTM	0.764	0.839
Bilinear-LSTM	0.755	0.832
Self-LSTM	0.759	0.830
Sequential-LSTM	0.797	0.865
Multihop-MLP-LSTM	0.768	0.849
Multihop-Bilinear-LSTM	0.788	0.864
Multihop-Self-LSTM	0.771	0.864
Multihop-Sequential-LSTM	**0.813**	**0.893**

shows some degree of improvement compared to MLP-LSTM. Based on self-attention mechanism, MAN outperforms the model with one attention layer by 1.2% on MAP and 3.4% on MRR. Overall, Multihop-Sequential-LSTM obtains the best results on TREC-QA dataset and surpasses the strong baseline RNN-POA [5] by 3.2% on MAP and 4.2% on MRR.

5.2 WikiQA

Table 4 reports the experimental results on WikiQA. First, we observe that MAN-based models outperform the models with a single attention layer. Multihop-Sequential-LSTM outperforms Sequential-LSTM by 2% in terms of MAP and 2.3% in terms of MRR. Multihop-Bilinear-LSTM shows improvements of 3.8% on MAP and 3.9% on MRR compared to Bilinear-LSTM. Multihop-MLP-LSTM and Multihop-Self-LSTM also perform slightly better than MLP-LSTM and Self-LSTM, respectively. Overall, Multihop-Sequential-LSTM achieves the best results on WikiQA and shows some degree of improvement compared to the strongest baseline RNN-POA [5].

5.3 InsuranceQA

Table 5 reports the experimental results on InsuranceQA. Our proposed approaches achieve highly competitive performances on this dataset, where Multihop-Sequential-LSTM obtains the best P@1 performance overall. Our best model surpasses the strong baseline IARNN-Gate on both test sets. Although most MAN-based models show some degree of improvement compared to the models with a single attention layer, applying one step of attention seems to be sufficient on this dataset. Interestingly, Self-LSTM performs quite well on InsuranceQA dataset even outperforms some interactive attention-based models. Multihop-Self-LSTM does not show any improvement against Self-LSTM. In addition, Sequential-LSTM again shows better results than MLP-LSTM and Bilinear-LSTM on this dataset.

Table 4: Experimental results on WikiQA. Baselines for WikiQA are reported in the first group. The second group shows the performance of models with a single attention layer. We report the performance of MANs in the last group.

Model	MAP	MRR
PV	0.511	0.516
PV + Cnt	0.599	0.609
CNN + Cnt	0.652	0.665
AP-LSTM (Santos et al. [7])	0.670	0.684
AP-CNN (Santos et al. [7])	0.689	0.696
ABCNN (Yin et al. [34])	0.692	0.710
Rank MP-CNN (Rao et al. [18])	0.701	0.718
RNN-POA (Chen et al. [5])	0.721	0.731
QA-LSTM	0.654	0.665
MLP-LSTM	0.686	0.695
Bilinear-LSTM	0.677	0.686
Self-LSTM	0.693	0.704
Sequential-LSTM	0.702	0.715
Multihop-MLP-LSTM	0.703	0.712
Multihop-Bilinear-LSTM	0.715	0.725
Multihop-Self-LSTM	0.702	0.710
Multihop-Sequential-LSTM	**0.722**	**0.738**

Table 5: Experimental results on InsuranceQA. Baselines for InsuranceQA are reported in the first group. The second group shows the performance of models with a single attention layer. We report the performance of MANs in the last group.

Model	Test1	Test2
CNN (Feng et al. [8])	0.628	0.592
CNN with GESD (Feng et al. [8])	0.653	0.610
AP-LSTM (Tan et al. [24])	0.690	0.648
IARNN-Gate (Wang et al [27])	0.701	0.628
QA-LSTM	0.643	0.617
MLP-LSTM	0.693	0.648
Bilinear-LSTM	0.689	0.658
Self-LSTM	0.699	0.653
Sequential-LSTM	0.702	0.665
Multihop-MLP-LSTM	0.695	0.655
Multihop-Bilinear-LSTM	0.694	0.662
Multihop-Self-LSTM	0.682	0.648
Multihop-Sequential-LSTM	**0.705**	**0.669**

5.4 FiQA

The results of the proposed models are shown in Table 6. On this new dataset we observe similar behaviours to other datasets. Firstly,

Table 6: Experimental results on FiQA. The first group shows the performance of models with a single attention layer. We report the performance of MANs in the second group.

Model	MAP	MRR	P@1
QA-LSTM	0.433	0.566	0.469
MLP-LSTM	0.497	0.616	0.509
Bilinear-LSTM	0.492	0.606	0.506
Self-LSTM	0.493	0.608	0.509
Sequential-LSTM	0.504	0.621	0.522
Multihop-MLP-LSTM	0.498	0.613	0.519
Multihop-Bilinear-LSTM	0.507	0.631	0.534
Multihop-Self-LSTM	0.488	0.619	0.522
Multihop-Sequential-LSTM	**0.529**	**0.655**	**0.567**

Table 7: Effect of different number of attention steps on FiQA

Model	MAP	MRR	P@1
Sequential-LSTM	0.504	0.621	0.522
Multihop-Sequential-LSTM*	0.514	0.636	0.543
Multihop-Sequential-LSTM (K=1)	0.527	0.649	0.561
Multihop-Sequential-LSTM (K=2)	**0.529**	**0.655**	**0.567**
Multihop-Sequential-LSTM (K=3)	0.523	0.644	0.546

attention-based models outperform the basic matching model QA-LSTM. MAN-based models perform better than the models with a single attention layer, in which Multihop-Sequential-LSTM obtains the best performance overall. More specifically, Multihop-Sequential-LSTM improves Sequential-LSTM by 4.5% in terms of P@1 while Multihop-Bilinear-LSTM shows an improvement of 2.8% on P@1 against Bilinear-LSTM. Multihop-MLP-LSTM indicates 1% enhancement on P@1 compared to MLP-LSTM whereas Multihop-Self-LSTM also increases over Self-LSTM by 1.3% in terms of P@1. Amongst interactive attention-based models Sequential-LSTM outperforms MLP-LSTM and Bilinear-LSTM. Compared to these models, Self-LSTM shows highly competitive results.

Overall, we summarize the key findings of our experiments.

- Similar to previous work, we observe that attention-based models perform significantly better than basic matching models.
- Multihop Attention Networks are better in capturing the complex semantic relations between questions and answers and outperform the models with only one attention layer.
- Sequential attention can be well adopted for the AS task and gains considerably improvements compared to traditional attention mechanisms.
- Self-attention can produce better representations than simply max or average pooling method and obtain competitive results on the AS task.

5.5 Effect of Attention Steps

Table 7 shows the influence of the number of steps on performance on FiQA dataset. Multihop-Sequential-LSTM* is the model where we consider the vectors returned by using max, mean and last pooling as different representations for the questions. Overall, Multihop-Sequential-LSTM performs better than Sequential-LSTM in which with $K = 2$ Multihop-Sequential-LSTM obtains the best performance. This might be due to the fact that the questions are rather short and often express two different aspects at most.

5.6 A Case Study

Figure 3 depicts the heat map of a test question from FiQA that was correctly answered by Multihop-Sequential-LSTM. The stronger the color of a word in the question (answer), the larger the attention weight of that word. As can be seen in the figure, in the first step Multihop-Sequential-LSTM puts more focus on some segments of the question and the parts of the answer that have some interactions with the question segments. In the second step, the MAN gives more attention to other segments of the question and consequently some other parts of the answer get more attention.

6 CONCLUSIONS

In this paper, we have presented Multihop Attention Networks for question answer selection. Our proposed MANs use multiple vectors which focus on different parts of a question to represent the overall semantics of the question and then apply multiple steps of attention to learn representations for the candidate answers. In addition, we also showed that sequential attention mechanism can be well adapted for this task. The mechanism allows local alignment information to be used when computing attention weight for each token in a sequence. Experimental results showed that MANs outperform state-of-the-art approaches on popular benchmark QA datasets. Empirical studies also confirm the effectiveness of sequential attention over other attention mechanisms. For future work, we want to employ this architecture on other tasks such as natural language inference, reading comprehension. In addition, combining multihop attention with convolutional neural networks is another direction to explore.

ACKNOWLEDGMENTS

We would like to thank Tuan-Anh Hoang and anonymous reviewers for their helpful comments. This work was partially funded by the BMBF project eLabour (01UG1512C), and the DFG project Managed Forgetting (NI-1760/1-1).

REFERENCES

[1] Jimmy Ba and Diederik Kingma. 2015. Adam: A Method for Stochastic Optimization. In *Proceedings of International Conference of Learning Representations*.
[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of International Conference of Learning Representations*.
[3] Sebastian Brarda, Philip Yeres, and Samuel R. Bowman. 2017. Sequential Attention: A Context-Aware Alignment Function for Machine Reading. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*. 75–80.
[4] Danqi Chen, Jason Bolton, and Christopher D. Manning. 2016. A Thorough Examination of the CNN/Daily Mail Reading Comprehension Task. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2358–2367.

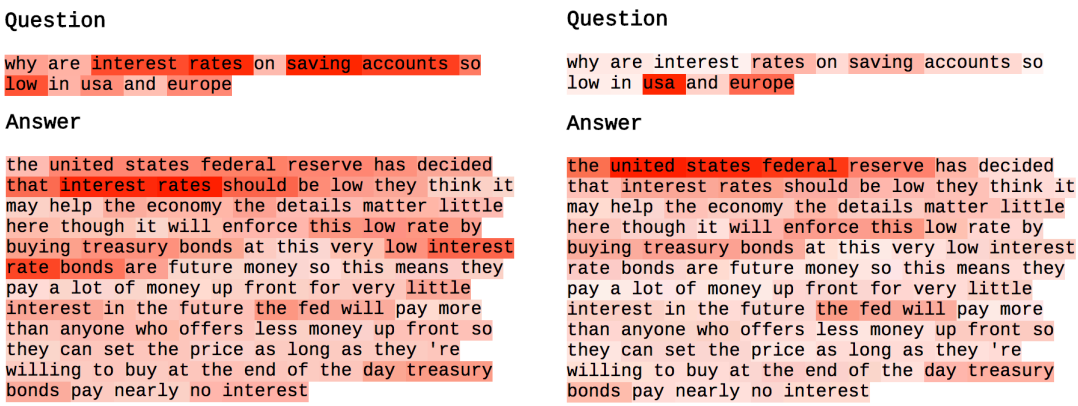

Figure 3: Attention heat map from Multihop-Sequential-LSTM (K=2) for a correctly selected answer.

[5] Qin Chen, Qinmin Hu, Jimmy Xiangji Huang, Liang He, and Weijie An. 2017. Enhancing Recurrent Neural Networks with Positional Attention for Question Answering. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 993–996.

[6] Sumit Chopra, Raia Hadsell, and Yann LeCun. 2005. Learning a Similarity Metric Discriminatively, with Application to Face Verification. In *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1 - Volume 01*. 539–546.

[7] Cicero dos Santos, Ming Tan, Bing Xiang, and Bowen Zhou. 2016. Attentive pooling networks. In *CoRR, abs/1602.03609*.

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

[9] Michael Heilman and Noah A. Smith. 2010. Tree Edit Models for Recognizing Textual Entailments, Paraphrases, and Answers to Questions. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. 1011–1019.

[10] Michael Heilman and Noah A. Smith. 2010. Tree Edit Models for Recognizing Textual Entailments, Paraphrases, and Answers to Questions. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. 1011–1019.

[11] Karl Moritz Hermann, Tomáš Kočiský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching Machines to Read and Comprehend. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*. 1693–1701.

[12] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Comput.* (1997), 1735–1780.

[13] Minghao Hu, Yuxing Peng, and Xipeng Qiu. 2017. Reinforced Mnemonic Reader for Machine Comprehension. *arXiv preprint arXiv:1705.02798* (2017).

[14] Ankit Kumar, Ozan Irsoy, Peter Ondruska, Mohit Iyyer, James Bradbury, Ishaan Gulrajani, Victor Zhong, Romain Paulus, and Richard Socher. 2016. Ask Me Anything: Dynamic Memory Networks for Natural Language Processing. In *Proceedings of The 33rd International Conference on Machine Learning*. 1378–1387.

[15] Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A Structured Self-Attentive Sentence Embedding. In *International Conference on Learning Representations 2017 (Conference Track)*.

[16] Yang Liu, Chengjie Sun, Lei Lin, and Xiaolong Wang. 2016. Learning Natural Language Inference using Bidirectional LSTM model and Inner-Attention. *CoRR* (2016).

[17] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. In *Empirical Methods in Natural Language Processing (EMNLP)*. 1532–1543.

[18] Jinfeng Rao, Hua He, and Jimmy Lin. 2016. Noise-Contrastive Estimation for Answer Selection with Deep Neural Networks. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. 1913–1916.

[19] Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiský, and Phil Blunsom. 2015. Reasoning about Entailment with Neural Attention. In *arXiv preprint arXiv:1509.06664*.

[20] Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A Neural Attention Model for Abstractive Sentence Summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. 379–389.

[21] Aliaksei Severyn and Alessandro Moschitti. 2013. Automatic Feature Engineering for Answer Selection and Extraction. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. 458–467.

[22] Aliaksei Severyn and Alessandro Moschitti. 2015. 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*. 373–382.

[23] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to Sequence Learning with Neural Networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*. 3104–3112.

[24] Ming Tan, Cicero dos Santos, Bing Xiang, and Bowen Zhou. 2016. Improved Representation Learning for Question Answer Matching. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 464–473.

[25] Ming Tan, Bing Xiang, and Bowen Zhou. 2015. LSTM-based Deep Learning Models for non-factoid answer selection. *CoRR* abs/1511.04108 (2015).

[26] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems 30*. 6000–6010.

[27] Bingning Wang, Kang Liu, and Jun Zhao. 2016. Inner Attention based Recurrent Neural Networks for Answer Selection. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 1288–1297.

[28] Di Wang and Eric Nyberg. 2015. A Long Short-Term Memory Model for Answer Sentence Selection in Question Answering. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*. 707–712.

[29] Mengqiu Wang and Christopher Manning. 2010. Probabilistic Tree-Edit Models with Structured Latent Variables for Textual Entailment and Question Answering. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*. 1164–1172.

[30] Mengqiu Wang, Noah A. Smith, and Teruko Mitamura. 2007. What is the Jeopardy Model? A Quasi-Synchronous Grammar for QA. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*. 22–32.

[31] Yi Yang, Wen-tau Yih, and Christopher Meek. 2015. WikiQA: A Challenge Dataset for Open-Domain Question Answering. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. 2013–2018.

[32] Xuchen Yao, Benjamin Van Durme, Chris Callison-Burch, and Peter Clark. 2013. Answer Extraction as Sequence Tagging with Tree Edit Distance. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 858–867.

[33] Wen-tau Yih, Ming-Wei Chang, Christopher Meek, and Andrzej Pastusiak. 2013. Question Answering Using Enhanced Lexical Semantic Models. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 1744–1753.

[34] Wenpeng Yin, Hinrich Schutze, Bing Xiang, and Bowen Zhou. 2016. ABCNN: Attention-Based Convolutional Neural Network for Modeling Sentence Pairs. *Transactions of the Association for Computational Linguistics* (2016), 259–272.

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