ADVANCED REVIEW

Text-based Question Answering from Information Retrieval and Deep Neural Network Perspectives: A Survey

Zahra Abbasiantaeb, Saeedeh Momtazi

Computer Engineering Department, Amirkabir University of Technology (Tehran Polytechnic), Tehran, 1591634311, Iran

[1]Computer Engineering Department, Amirkabir University of Technology (Tehran Polytechnic), Tehran, 1591634311, Iran

Correspondence
Saeedeh Momtazi, Computer Engineering Department, Amirkabir University of Technology (Tehran Polytechnic), Tehran, 1591634311, Iran
Email: momtazi@aut.ac.ir

Funding information

Text-based Question Answering (QA) is a challenging task which aims at finding short concrete answers for users' questions. This line of research has been widely studied with information retrieval techniques and has received increasing attention in recent years by considering deep neural network approaches. Deep learning approaches, which are the main focus of this paper, provide a powerful technique to learn multiple layers of representations and interaction between questions and texts. In this paper, we provide a comprehensive overview of different models proposed for the QA task, including both traditional information retrieval perspective, and more recent deep neural network perspective. We also introduce well-known datasets for the task and present available results from the literature to have a comparison between different techniques.

KEYWORDS
Text-based Question Answering, Deep Learning, Information Retrieval

Abbreviations: QA, question answering; KB, knowledge base; CNN, Convolutional Neural Network; RNN, Recurrent Neural Network; BERT, Bidirectional Encoder Representations from Transformers.

1 | INTRODUCTION

Question Answering (QA) is a fast-growing research problem in computer science that aims to find short concrete answers. There are two major approaches for QA systems: text-based QA, and knowledge-based QA. Knowledge-based QAs rely on knowledge bases (KB) for finding the answer to the user's question. Freebase is one of the most popular KBs (Bollacker et al., 2008) which has been widely used as a benchmark in many recent works on knowledge-based QA. KBs include entities, relations, and facts. Facts in the knowledge base are stored in (subject, predicate, object) format where the subject and the object are entities and the predicate is a relation, indicating the relation between the object and the subject. For example, the answer to the question 'In which city was Albert Einstein born?' could be stored in a fact like '(Albert Einstein, place-of-birth, Ulm)'. In this task, there are two types of questions: single-relation and multi-relation questions. A simple question is answered by one fact in KB, while the answer of a multi-relation question is found by reasoning over more than one fact in the KB (Yu et al., 2017). SimpleQuestions and WebQSP are the major datasets of the single-relation and multi-relation questions, respectively.

In text-based QA, the answer to a candidate question is obtained by finding the most similar answer text between candidate answer texts. Consider the question Q and set of answers $\{A_1, A_2, ..., A_n\}$, the goal of this system is finding the best answer among these answers. Recent works have proposed different deep neural models in text-based QA which compares two segments of texts and produces a similarity score. In this paper, we focus on this type of QA and review the available methods on text-based QA.

Figure 1 presents an overall taxonomy of QA systems including the main representative models of each category. Diefenbach et al. (2018) provided a survey that only covers QA over knowledge-base and contains the proposed models until 2017. Soares and Parreiras (2018) is another survey article that does not discuss the proposed models and it also includes the works before 2018. Kodra (2017) named a wide variety of QA systems and only discusses a small number of proposed models in each type of QA while focusing on neural models. (Wu et al., 2019) is another recent survey which only covers the QA over knowledge-base. Dimitrakis et al. (2019) provided an overview of different components of a QA system, but it does not discuss the architecture of the proposed models. (Lai et al., 2018) published a survey that provides a different perspective for classifying deep learning methods and does not cover the architecture of proposed models.

Our paper provides a coherent and complete overview of the architecture of the representative models in QA over text from different perspectives, including the state-of-the-art models in this area. Also, noted features of proposed models are discussed in this paper. Moreover, in recent years, pre-trained contextualized language models, such as ELMO, BERT, RoBERTa, and ALBERT, has demonstrated great advances in Natural Language Processing (NLP) downstream tasks including QA, but none of the above-mentioned articles discussed these models.

This paper organizes as follows: In section 2, we present the architecture of QA systems. We discuss information retrieval-based models used for question answer similarity in section 3 and deep learning models in section 4, respectively. In sections 5 and 6, we introduce the most popular QA datasets and evaluation metrics used in QA, respectively. We report and compare the results of reviewed models in section 7 and discuss the paper in section 8. The paper is concluded in section 9.

2 | ARCHITECTURE OF TEXT-BASED QUESTION ANSWERING

The architecture of Text-based QA, as illustrated in Figure 2, includes three major phases: question processing, document and passage retrieval, and answer extraction. Each of these phases is described below (Jurafsky and Martin, 2009).

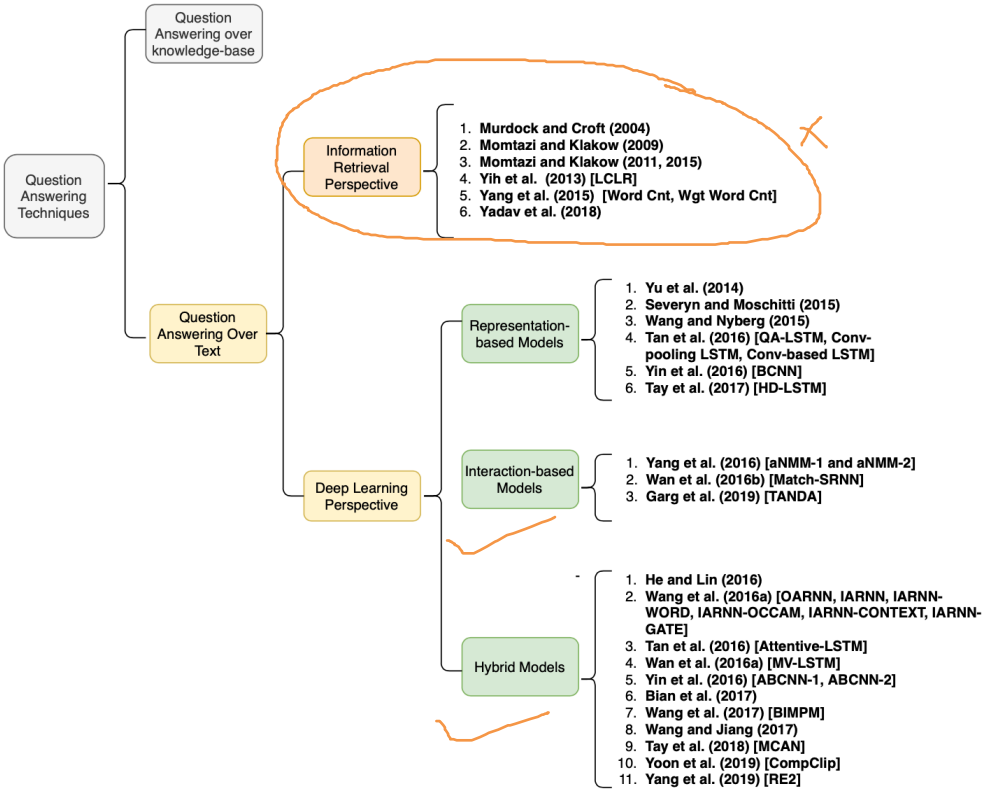
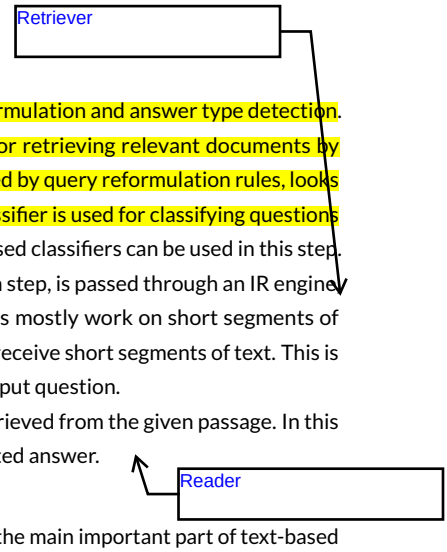


FIGURE 1 Taxonomy of QA techniques provided in this paper



1. Question processing: This phase includes two major steps, namely query formulation and answer type detection. In the query formulation step, a query for a given question is generated for retrieving relevant documents by employing an Information Retrieval (IR) engine. The query, which is generated by query reformulation rules, looks like a subset of the intended answer. In the answer type detection step, a classifier is used for classifying questions based on the type of expected answer. Different neural-based or feature-based classifiers can be used in this step.
2. Document and passage retrieval: Generated query in the query formulation step, is passed through an IR engine and top n retrieved documents are returned. As answer extraction models mostly work on short segments of documents, a passage retrieval model is applied on retrieved documents to receive short segments of text. This is the core component of QA that can find similar passages/sentences to the input question.
3. Answer extraction: In the final phase of QA, the most relevant answer is retrieved from the given passage. In this step, we need to measure the similarity of the input question and the extracted answer.

As mentioned, estimating the similarity of question and answer sentences is the main important part of text-based QA systems. Since the retrieved sentences are short enough to satisfy users, the output of this step can also be represented to users without any further answer extraction. The similarity of question and answer sentences can be measured by information retrieval or deep learning approaches. In Sections 3 and 4, we review related works from the

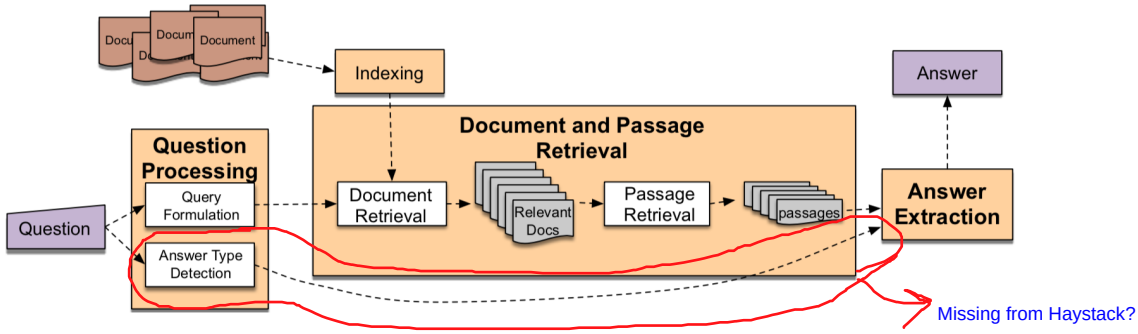


FIGURE 2 Genaral architecture of text-based QA (Jurafsky and Martin, 2009)

information retrieval perspective and the deep learning approaches, respectively.

3 | QUESTION ANSWER SIMILARITY FROM INFORMATION RETRIEVAL PERSPECTIVE

Although lexical-based information retrieval models have been widely used in ad-hoc retrieval, they have less applied to QA tasks because the length of answer sentences is shorter than normal web documents and the vocabulary gap between question and answer sentence in QA is more pronoun than ad-hoc retrieval. It motivated researches to use advanced information retrieval approaches in QA. Some of these approaches are described in this section. At the end of the section a brief overview of information retrieval approaches is presented in Table 1.

Yang et al. (2015)presented the WikiQA dataset for open domain QA. They have evaluated WikiQA and QASent datasets with information retrieval-based models like Word Count (Word Cnt), Weighted Word Count (Wgt Word Cnt), Learning Constrained Latent Representation (LCLR) (Yih et al., 2013), and Paragraph Vector (PV) (Le and Mikolov, 2014). Word Cnt model works by counting the non-stop words in question which also have occurred in the answer sentence. Wgt Word Cnt is the same as Word Cnt, but it also re-weights the counts by Inverse document Frequency (IDF) weight of the question words. The Idea behind the model proposed by Yih et al. (2013) is adopting a probabilistic classifier for predicting whether a pair of question and answer are related or not, using semantic model of the question and answer. They used synonym/antonym, hypernym/hyponym and semantic word similarity of each pair of words from question and answer sentences for creating the semantic model. They adopted Learning Constrained Latent Representation (LCLR) (Chang et al., 2010) for classifying a pair of question and answer. Details of this classifier is shown in the following equations.

$$\begin{aligned} &\min_{\theta} \quad \frac{1}{2}\|\theta\|^2 + C\sum_i \xi_i^2 \\ &\text{s.t.} \quad \xi_i \geq 1 - y_i \max_h \theta^T \phi(x,h) \\ &\arg\max_h \theta^T \phi(x,h) \end{aligned} \tag{1}$$

Murdock and Croft (2004) suggested a translation model for QA. In their model, probability of the question (Q)

given the answer (A), denoted as $P(Q|A)$, is computed by the following equations:

$$p(Q|A) = \prod_{i=1}^{m} \left[\beta \left(\lambda \sum_{j=1}^{n} p\left(q_i|a_j\right) p\left(a_j|A\right) + (1-\lambda) p\left(q_i|C\right) \right) + (1-\beta)\left(\lambda p\left(q_i|D_A\right) + (1-\lambda) p\left(q_i|C\right)\right) \right] \tag{2}$$

$$p\left(q_i|a_j\right) p\left(a_j|A\right) = t_i p\left(q_i|A\right) + (1-t_i) \sum_{1 \le j \le n, a_j \ne q_i} p\left(q_i|a_j\right) p\left(a_j|A\right) \tag{3}$$

where λ is the smoothing parameter, D_A is the document containing answer A, and C is the collection. The idea is based on the model proposed by Berger and Lafferty (1999) while different similarity model is used for calculating $P(q_i|a_j)$.

Momtazi and Klakow (2009) proposed class-based language models for sentence retrieval in QA. Their class-based language model aims to mitigate the word mismatch problem by finding the relation between words. The Brown word clustering algorithm is adopted for clustering the words in this model and the probability of generating question Q, having the answer sentence S is calculated by the following equations:

$$P_{\text{class}}(Q|S) = \prod_{i=1}^{M} P\left(q_i|C_{q_i}, S\right) P\left(C_{q_i}|S\right) \tag{4}$$

$$P_{\text{class}}\left(C_{q_i}|S\right) = \frac{f_S\left(C_{q_i}\right)}{\sum_w f_S(w)}$$

where $P(q_i|Cq_i, S)$ is the probability of term q_i having its cluster (C_{q_i}) and answer sentence model (S), $P(C_{q_i}|S)$ is the probability of cluster C_{q_i} given the sentence model S, $f_s(C_{q_i})$ is the number of occurrences of all the words in the cluster of term q_i in sentence S, and w represents the vocabulary words.

Momtazi and Klakow (2011, 2015) proposed a trained trigger language model. In their model, the word mismatch problem is mitigated by using the contextual information between words. The Idea behind this model is to find the pair of trigger and target words, whereas appearance of a target word in answer sentence and trigger word in question sentence means as a relation between the related words. They have trained a model for extracting these trigger-target pairs from a corpus, and this model is used for calculating the probability of question word q_i, having the answer S, denoted as $P(q_i|S)$, as follows:

$$P_{\text{trigger}}\left(q_i|S\right) = \frac{1}{N} \sum_{j=1}^{N} P_{\text{trigger}}\left(q_i|s_j\right) \tag{5}$$

$$P_{\text{trigger}}\left(q_i|s_j\right) = \frac{f_C\left(q_i, s_j\right)}{\sum_{q_r} f_C\left(q_i, s_j\right)}$$

where s_j and q_i denote the j^{th} term in answer sentence, and i^{th} term in the question sentence, respectively. $f_c(q_i, s_j)$ is

the number of times term q_i triggers term s_j in the model created upon corpus C. $P(q_i|S)$ is the probability of q_i having the sentence S, and N is sentence length.

Having the above probabilities, the probability of question (Q), having the answer(S), is calculated as follows:

$$P_{\text{trigger}}(Q|S) = \left(\frac{1}{N}\right)^{M} \prod_{i=1}^{M} \sum_{j=1}^{N} \frac{f_C(q_i, s_j)}{\sum_q f_C(q_i, s_j)} \tag{6}$$

where M is question length.

Yadav et al. (2018) proposed a model which for each question-answer pair calculates a matching score in three steps. In the first step, IDF weight of each word is calculated by the following equation:

$$idf(q_i) = \log \frac{N - \text{docfreq}(q_i) + 0.5}{\text{docfreq}(q_i) + 0.5} \tag{7}$$

where N is the count of questions, and $docfreq(q_i)$ is the number of questions which word q_i has occurred in. In the second step, one-to-many alignments are performed between terms in question and answer. Cosine similarity between Glove word embedding (Pennington et al., 2014) of each question word q_i and each answer word a_i is considered as their similarity. Then the top K^+ most similar words $\{a^+_{q^i,1}, a^+_{q^i,2}, a^+_{q^i,3}, ..., a^+_{q^i,K^+}\}$ and K^- least similar words $\{a^-_{q^i,1}, a^-_{q^i,2}, a^-_{q^i,3}, ..., a^-_{q^i,K^+}\}$ of answer are found. Finally in the third step the similarity score $S(Q, A)$ between each question and answer sentence is calculated by the following equations:

$$s(Q, A) = \sum_{i=1}^{N} idf(q_i) \cdot \text{align}(q_i, A) \tag{8}$$

$$\text{align}(q_i, A) = \text{pos}(q_i, A) + \lambda \cdot \text{neg}(q_i, A)$$

$$\text{pos}(q_i, A) = \sum_{k=1}^{K^+} \frac{1}{k} \cdot a^+_{q_i,k}$$

$$\text{neg}(q_i, A) = \sum_{k=1}^{K^-} \frac{1}{k} \cdot a^-_{q_i,k}$$

where $align(q_i, A)$ is the alignment score between q_i and answer A, λ is the negative information's weight, $pos(q_i, A)$ and $neg(q_i, A)$ represent the one-to-many alignment score for the K^+ most and K^- least similar words. They have also proposed two other baselines single-alignment (one-to-one), and one-to-all. In one-to-one approach, just the single alignment score $K^+ = 1$, the most similar word, is used. In the one-to-all approach, similarity between the question term q_i and all the answer terms is considered with the same weight in calculating $align(q_i, A)$ which changes the $align(q_i, A)$ to the following equation:

$$\text{align}(q_i, A) = \sum_{k=1}^{m} \frac{1}{k} \cdot \text{cosSim}\left(q_i, a^{+}_{q_i,k}\right) \tag{9}$$

where M is the count of the words in the answer sentence.

TABLE 1 Overview of QA Models from Information Retrieval Perspective

Model	Main Idea	Datasets
Yang et al. (2015) Word Cnt	uses count of co-occurred words in question and answer sentences	WikiQA
Yang et al. (2015) Wgt Word Cnt	uses Inverse document Frequency (IDF) for re-weighting the counts in Word Cnt model	WikiQA
(Yih et al., 2013) LCLR	uses probabilistic classifier for classifying question and answer pairs	WikiQA TREC-QA
Murdock and Croft (2004)	uses translation model for predicting probability of question given the answer	TREC-QA
Momtazi and Klakow (2009)	uses class-based language model for mitigating the word mismatch problem	TREC-QA
Momtazi and Klakow (2011, 2015)	uses a trained trigger language model for mitigating the word mismatch problem by using the contextual information between words	TREC-QA
Yadav et al. (2018)	uses one-to-many alignments, and IDF weight for calculating the matching score of question-answer pair	

4 | QUESTION ANSWER SIMILARITY FROM DEEP LEARNING PERSPECTIVE

Deep learning models can be divided into three major categories: representation-based, interaction-based, and hybrid (Guo et al., 2016). Representation-based models construct a fixed-dimensional vector representation for both the question and the candidate answer separately and then perform matching within the latent space. Interaction-based models compute the interaction between each individual term of question and candidate answer sentences where interaction can be identity or syntactic/semantic similarity. Hybrid models combine both interaction and representation models. They consist of a representation component that combines a sequence of words into a fixed-dimensional representation and an interaction component. These components could occur in parallel or serial. In this section, we will

review the structure of the proposed deep neural models and specify the type of model according to the mentioned categories. Similar to the previous section, we provide a brief overview of deep learning-based models at the end of this section.

4.1 | Representation-based Models

Yu et al. (2014) proposed a generative neural network-based model for binary classification of each question/answer pair is related or not. This model captures the semantic features of question and answer sentences. Each sample is represented with a triple (q_i, a_{ij}, y_{ij}) where $q_i \in Q$ is question, a_{ij} is a candidate answer for question q_i, and label y_{ij} shows whether a_{ij} is a correct answer for q_i or not. For each answer, a related question is generated and then the semantic similarity of generated question and the given question is captured by using the dot product. This similarity is used for predicting whether the candidate answer is a correct answer for the given question or not. The probability of the answer being correct is formulated as:

$$\mathrm{P(y = 1 \,|q, a)} = \sigma\left(\mathrm{q_m^T Ma + b}\right) \tag{10}$$

where $q' = Ma$ is the generated question. The model is trained by minimizing the cross-entropy of all labeled data QA pairs as:

$$\mathcal{L} = -\log \prod_n p\left(y_n \,|q_n, a_n\right) + \frac{\lambda}{2}\|\theta\|_F^2 \tag{11}$$

$$= -\sum_n y_n \log \sigma\left(q_n^T f M a_n + b\right) + (1 - y_n) \log\left(1 - \sigma\left(q_n^T M a_n + b\right)\right) + \frac{\lambda}{2}\|\theta\|_F^2$$

where $\|\theta\|_F^2$ is the Frobenius norm of θ.

Each sentence is modeled by the bag of words and bigram approaches. In the bag of words model, a sentence is represented by averaging embeddings of all the words (except stop words) within it. The bigram model has the ability to capture features of bigrams independent of their positions in the sentence. As the architecture of the bigram model is shown in Figure 3, one convolutional layer, and one pooling layer are used for modeling the sentence in the bigram model. Every bigram is projected into a feature value c_i, which is computed as:

$$c_\mathrm{i} = \tanh\left(\mathrm{T} \cdot \mathrm{s_{i:}}_{i+1} + \mathrm{b}\right) \tag{12}$$

where s is the vector representation of the sentence. All bigram features are combined in average pooling layer and finally a full-sentence representation with the same dimensionality as the initial word embeddings are produced by the following equation:

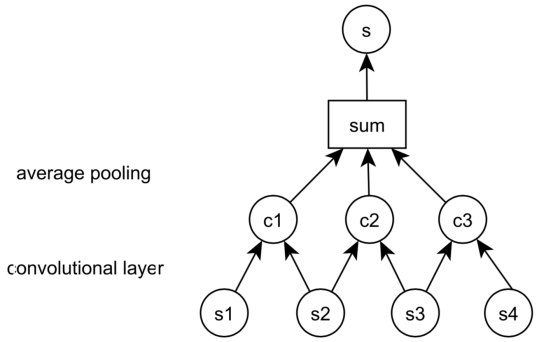


FIGURE 3 Architecture of Yu et al. (2014) model

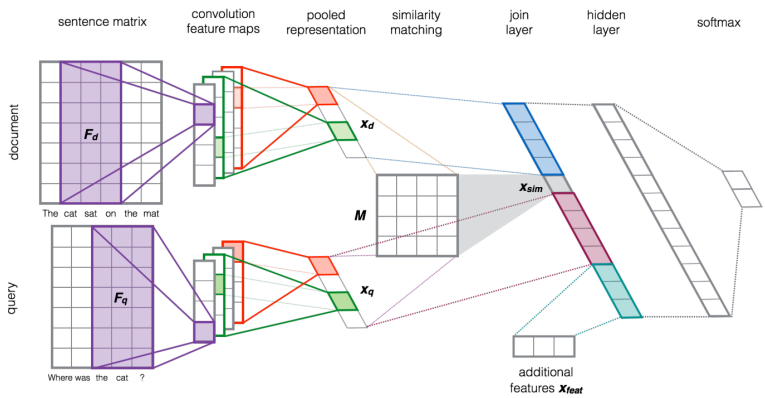


FIGURE 4 Architecture of Severyn and Moschitti (2015) model

$$s = \sum_{i=1}^{|s|-1} \tanh\left(T_L s_i + T_R s_{i+1} + b\right) \tag{13}$$

Severyn and Moschitti (2015) proposed a framework for answer sentence selection. They divided their task into two main subtasks: (1) mapping the original space of words to a feature space encoding, and (2) learning a similarity function between pairs of objects. They used a Convolutional Neural Network (CNN) architecture for learning to map input text, either query or document, to a vector space model. For the second part, they used the idea of noisy channel approach for finding a transformation of the document to be as close as possible to the query: $Sim(x_q, x_d) = x_q^T M x_d$. To this end, they used a neural network architecture to train the similarity matrix M. According to Figure 4, the vector representation of the query and the document that is derived from the first CNN model are jointly fed to the second CNN to train and build the similarity matrix.

Wang and Nyberg (2015) used a multilayer stacked Bidirectional Long Short-term Memory (BiLSTM) for answer sentence selection task. As represented in Figure 5, a sequence of word2vec representation of question and answer sentence terms are fed to this model. Symbol, $< S >$, is placed between question q and answer a for distinguishing the

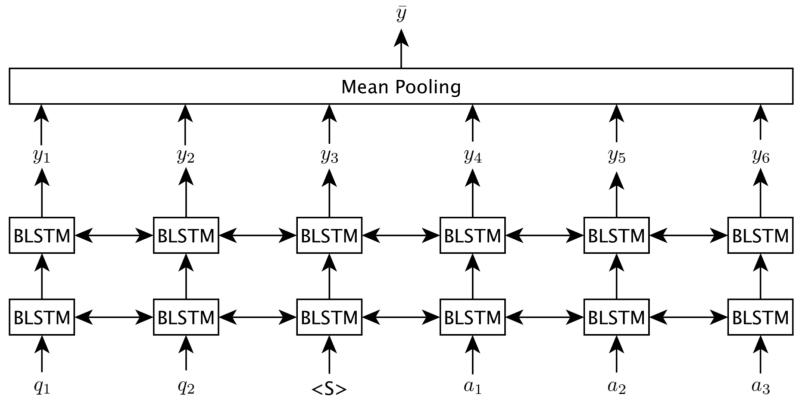


FIGURE 5 Architecture of Wang and Nyberg (2015) model

question and answer. Among different Recurrent Neural Network (RNN) architectures, stacked BiLSTM is chosen as first bidirectional RNN extracts the contextual information of question and answer pair from both directions, in other words, it uses the future information, second stacked BiLSTM provides better results due to its ability in extracting higher levels of abstraction, and third LSTM is a more complicated RNN block which mitigates the gradient vanishing problem of standard RNNs. The final output of each time step indicates whether the given answer is a correct answer for the question or not.

In this model, the stacked BiLSTM relevance model is combined by Gradient Boosted Regression Tree (GBDT) method for exact matching the proper nouns and cardinal numbers in question and answer sentences.

Tan et al. (2016) proposed a basic model called QA-LSTM, shown in Figure 6, for sentence matching. According to this figure, in the basic model, word embeddings of question and answer sentences are fed into a BiLSTM network. A fixed-size representation is obtained for each sentence in three different ways: (1) concatenating the last output of both directions, (2) average pooling and max pooling over all the outputs of the BiLSTM, and finally, (3) using the cosine similarity, semantic matching between question and answer sentences are scored. LSTM is a powerful architecture in capturing long-range dependencies, but it suffers from not paying attention to local n-grams. Although convolutional structures pay more attention to local n-grams, they do not consider long-range dependencies. Therefore each of the CNN and RNN blocks has its own pros and cons. Three different variants of the basic QA-LSTM are proposed in this work which one of them belongs to the hybrid models and is described in section 4.3. In the following, two other variants of QA-LSTM, belonging to the representation-based models, are described.

1. Convolutional pooling LSTM: As is shown in Figure 7, the pooling layer is replaced with a convolutional layer for capturing richer local information and on top of this layer an output layer is placed for generating a representation of the input sentence. Representation of the input sentence is generated by the following equations:

$$C = tanh(W_{cp}Z), [O_j] = max_{1<l<L}[C_{j,l}] \tag{14}$$

where $Z \in R^{k|h|\times L}$ and m-th column is generated by concatenation of the k hidden vectors of BiLSTM centralized in the m-th token of the sequence, L is the length of the sequence, and W_{cp} is the network parameter.

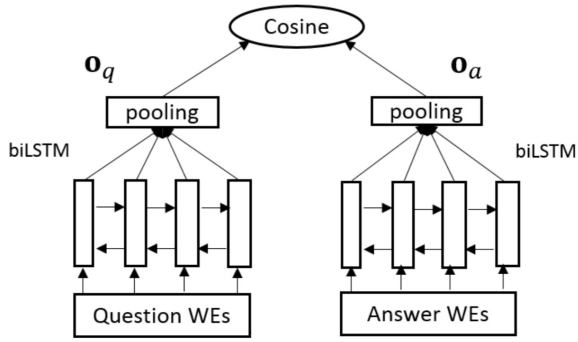


FIGURE 6 Architecture of QA-LSTM basic model (Tan et al., 2016)

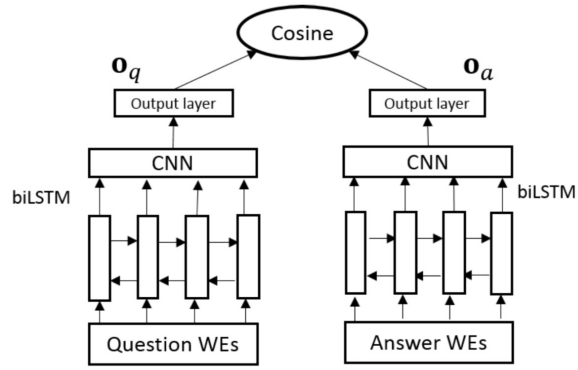


FIGURE 7 Architecture of Conv-pooling LSTM model (Tan et al., 2016)

2. Convolutional-based LSTM: Architecture of this model is shown in Figure 8. In this model, word embeddings are first fed to a CNN for retrieving the local n-gram interactions at the lower level. The output of the convolution is then fed into the BiLSTM network for capturing long-range dependencies. Max pooling is used over the output of the BiLSTM for producing the sentence representation. The output of the convolution layer, named X, is obtained by the following equation:

$$X = tanh(W_{cb}D) \tag{15}$$

where $D \in R^{kE \times L}$ is input of this model and column l of D is concatenation of k word vectors of size E centered at the l-th word.

Yin et al. (2016) proposed a Basic CNN (BCNN) model and three Attention-based CNN (ABCNN) models for text matching. ABCNN models belong to the hybrid category and are described in section 4.3. In the following, the architecture of BCNN is described.

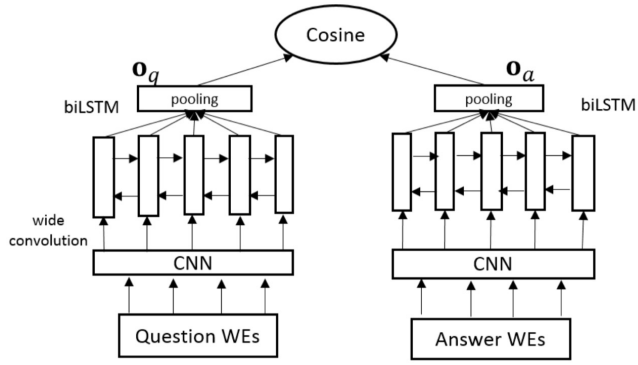


FIGURE 8 Architecture of Conv-based LSTM model (Tan et al., 2016)

Basic CNN (BCNN): Architecture of this model is shown in Figure 9. This model is based on the Siamese architecture (Bromley et al., 1993). The model provides a representation of each sentence using convolutional, $w-ap$ pooling and $all-ap$ pooling layers, and then compares these two representations with logistic regression. Different layers in BCNN are as follows:

1. Input layer: Each sentence is passed to the model with a $d_0 \times s$ matrix, where d_0 is the dimension of word2vec (Mikolov et al., 2013) embedding of each word and s is the maximum length of the two sentences (the shorter sentence is padded to the larger sentence length).
2. Convolution layer: Embedding of words within a sentence with window size of w are concatenated and represented as $c_i \in R^{w.d_0}$ where $0 < i < s + w$ (s is length of the sentence). Then each c_i is converted to p_i by the following equation:

$$P_i = tanh(W.c_i + b) \tag{16}$$

   let $W \in R^{d_1 \times w.d_0}$ be the convolution weights, and $b \in R^{d_1}$ be the bias.
3. Average pooling layer: This model utilizes two types of average pooling, namely $all-ap$ and $w-ap$, for extracting the robust features from convolution. $All-ap$ pooling is used in the last convolution layer and calculates the average of each column. $W-ap$ pooling is used in the middle convolution layers and calculates the average of each w consecutive columns.
4. Output layer: In the output layer logistic regression is applied to final representations in order to classify the question and answer sentences as related or not.

Tay et al. (2017) proposed Holographic-dual LSTM (HD-LSTM), a binary classifier model for QA task. As is shown in Figure 10, HD-LSTM consists of four major parts. In the representation layer, two multi-layered LSTMs denoted as Q-LSTM and A-LSTM are used for learning the representations of the question and answer. A holographic composition is used for measuring the similarity of the outputs of Q-LSTM and A-LSTM. Finally, a fully connected hidden layer is used for performing the binary classification of the QA pair as correct or incorrect. Each part of HD-LSTM is described in the following:

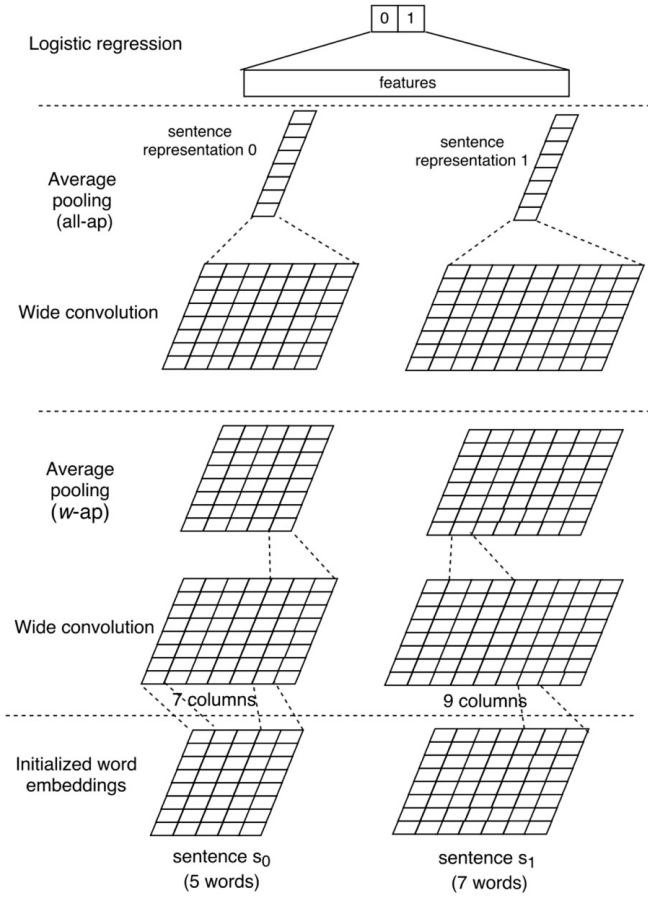


FIGURE 9 Architecture of BCNN model (Yin et al., 2016)

1. Learning QA Representations: Instead of learning word embeddings, pre-trained weights of SkipGram embeddings (Mikolov et al., 2013) denoted as W are used in this layer. Embeddings of both the question and the answer sequences are fed into Q-LSTM and A-LSTM. Representation of question and answer is generated in the last hidden output of Q-LSTM and A-LSTM.
2. Holographic Matching of QA pairs: Embeddings of the question and answer which learned in the previous layer are passed into the holographic layer and circular correlation of vectors is used for modeling the similarity of them. The similarity of question and answer is modeled by the following equation:

$$[q \star a]_k = \sum_{i=0}^{d-1} q_i a_{(k+i) \bmod d}$$
$$q \star a = \mathcal{F}^{-1}(\overline{\mathcal{F}(q)} \odot \mathcal{F}(a)) \tag{17}$$

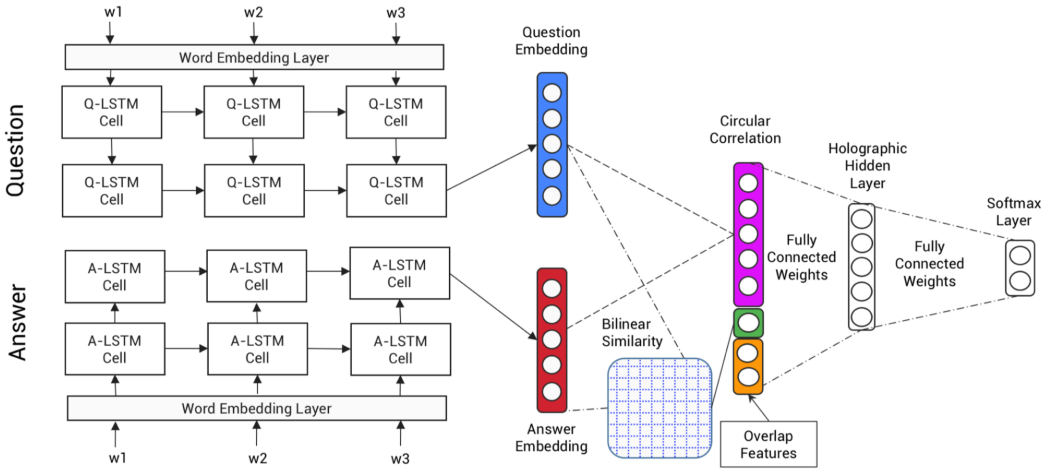


FIGURE 10 Architecture of HD-LSTM model (Tay et al., 2017)

where F is Fast Fourier transform, q is question, a is answer, and d is the dimension of embeddings. In the above equation, question and answer embeddings must have the same dimension.

3. Holographic Hidden Layer: This is a fully connected dense layer. Input and output of this layer are $[[q \star a], Sim(q, a), X_{feat}]$ and h_{out}, respectively. (X_{feat}) is word overlap feature, and $Sim(q, a)$ is bilinear similarity function between q and a which is defined as:

$$\text{Sim}(q, a) = \vec{q}^T M \vec{a} \tag{18}$$

where $M \in R^{n \times n}$ is a similarity matrix between $q \in R^n$ and $a \in R^n$. Concatenation of $Sim(q, a)$ with $[q \star a]$ makes the model perform worse. So, in order to mitigate this weakness X_{feat} is concatenated to make the model work better.

4. SoftMax layer: A softMax layer with the following equation is used at last:

$$\text{P} = \text{SoftMax}\ (\text{W}_\text{f}.\text{h}_{\text{out}} + \text{b}_\text{f}) \tag{19}$$

where W_f and b_f are network parameters.

4.2 | Interaction-based Models

Yang et al. (2016) proposed aNMM-1 and aNMM-2 neural matching models. ANMM-1 works in three major steps as follows:

1. Building QA matching matrix: Each cell in this matrix represents the similarity of the corresponding question and answer words. The similarity is calculated by the dot product of the normalized word embeddings.
2. Learning semantic matching: Various length of answer sentences results in variable size for the QA matrix. To fix

this problem, value shared weights method is used. In value shared weights method, each node is weighted based on its value where the value of a node represents the similarity between two words. Input to the hidden layer for each question term is defined as follows:

$$h_j = \delta\left(\sum_{k=0}^{K} w_k \cdot x_{jk}\right) \tag{20}$$

where j is the index of the question term, w_k is the model parameter, and x_{jk} is the sum of all matching signals within the range k (the range of possible matching signals is divided to k equal bins, and each matching score is assigned to one bin).

3. Question attention network: An attention layer with question word embedding weights is applied to hidden states h_j. Finally match score is computed by the following equation:

$$y = \sum_{j=1}^{M} g_j \cdot h_j = \sum_{j=1}^{M} \frac{\exp\left(v \cdot q_j\right)}{\sum_{l=1}^{L} \exp\left(v \cdot q_l\right)} \cdot \delta\left(\sum_{k=0}^{K} w_k \cdot x_{jk}\right) \tag{21}$$

where v is the model's parameter and dot product of the question word embedding and v are fed to the softMax function.

In aNNM-2, more than one value-shared weights are used for each question answer matching vector then in the first hidden layer, there are multiple intermediate nodes. As architecture of aNMM-2 is shown in Figure 11, the final output of the model y is defined as:

$$y = \sum_{j=1}^{M} \tau\left(v \cdot q_j\right) \cdot \delta\left(\sum_{t=0}^{T} r_t \cdot \delta\left(\sum_{k=0}^{K} w_{kt} x_{jk}\right)\right) \tag{22}$$

where T is the number of nodes in hidden layer 1, r_t is the model parameter from hidden layer 1 to hidden layer 2, and $\tau(v, q_j)$ is calculated as follows:

$$\frac{\exp(v.qj)}{\sum_{l=1}^{L} \exp(v.ql)} \tag{23}$$

Wan et al. (2016b) proposed a recursive semantic matching model called Match-SRNN. According to Figure 12, which shows the architecture of Match-SRNN, Match-SRNN works in three major steps: In the first step word-level interactions are modeled. In the second step a special case of interaction between two prefixes of two different sentence ($S1[1:i]$ and $S2[1:j]$) is modeled as a function of the interaction between $S1[1:i-1]$ and $S2[1:j]$, $S1[1:i]$ and $S2[1:j-1]$, $S1[1:i-1]$ and $S2[1:j-1]$, and interaction between words w_i and v_j. Then the equation for this special interaction is:

$$h_{ij} = f\left(h_{i-1,j}, h_{i,j-1}, h_{i-1,j-1}, s\left(w_{i,}, v_j\right)\right) \tag{24}$$

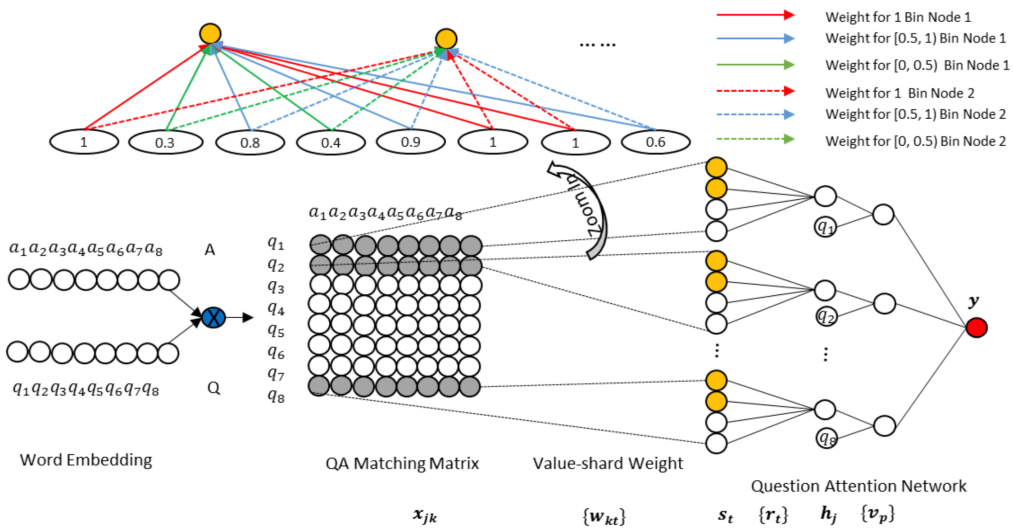


FIGURE 11 Architecture of ANMM-2 model (Yang et al., 2016)

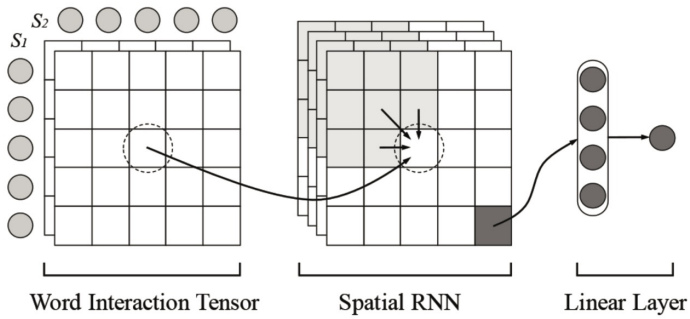


FIGURE 12 Wan et al. (2016b) (Match-SRNN) model

where h_{ij} is the interaction between $S1[1:i]$ and $S2[1:j]$. This recursive way of modeling the interaction helps to capture the long-term dependencies between two sentences. In the third step, a linear function is used for measuring the matching score of two given sentences. More details about these steps are below.

1. Neural tensor network: The interaction between two words w_i and v_j is captured by a neural tensor network according to the following equation:

$$\vec{s}_{ij} = F\left(u\left(w_i\right)^T T^{[1:c]} u\left(v_j\right) + W\begin{bmatrix} u\left(w_i\right) \\ u\left(v_j\right) \end{bmatrix} + \vec{b}\right) \tag{25}$$

where s_{ij} is a vector representation of the similarity between w_i and v_j words, T_i is one slice of the tensor parameters,

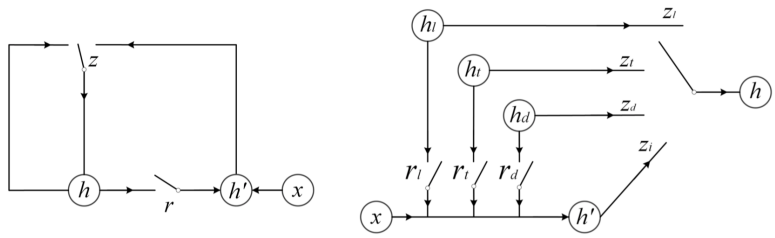


FIGURE 13 Architecture of Spatial-GRU (right) and GRU (left)

W and b are parameters and $F(Z) = max(0, Z)$.

2. Spatial RNN: In this layer, GRU is used as an RNN because of its easy implementation for implementing a Spatial-GRU which models the h_{ij}. Figure 13 shows the architecture of the 1D-GRU and Spatial-GRU which is used in this work.

   According to the right part in Figure 13 Spatial-GRU has four updating gates, and three reset gates. Function f in the Spatial-GRU is computed as follow:

$$\vec{q}^T = \left[\vec{h}^T_{i-1,j}, \vec{h}^T_{i,j-1}, \vec{h}^T_{i-1,j-1}, \vec{s}^T_{ij}\right]^T \tag{26}$$

$$\vec{r}_l = \sigma\left(W^{(r_l)}\vec{q} + \vec{b}^{(r_l)}\right), \overrightarrow{r_t} = \sigma\left(W^{(r_t)}\vec{q} + \vec{b}^{(r_t)}\right)$$

$$\vec{r}_d = \sigma\left(W^{(r_d)}\vec{q} + \vec{b}^{(r_d)}\right), \vec{r}^T = \left[\vec{r}^T_l, \vec{r}^T_t, \vec{r}^T_d\right]^T$$

$$\vec{z}'_i = W^{(z_i)}\vec{q} + \vec{b}^{(z_i)}, \overrightarrow{z'_l} = W^{(z_l)}\vec{q} + \vec{b}^{(z_l)}$$

$$\vec{z}'_t = W^{(z_t)}\vec{q} + \vec{b}^{(z_t)}, \vec{z}'_d = W^{(z_d)}\vec{q} + \vec{b}^{(z_d)}$$

$$\left[\vec{z}_i, \vec{z}_l, \vec{z}_t, \vec{z}_d\right] = \text{SoftmaxByRow}\left(\left[\vec{z}'_i, \vec{z}'_l, \vec{z}'_t, \vec{z}'_d\right]\right)$$

$$\vec{h}'_{ij} = \phi\left(W\vec{s}_{ij} + U\left(\vec{r} \odot \left[\vec{h}^T_{i,j-1}, \vec{h}^T_{i-1,j}, \vec{h}^T_{i-1,j-1}\right]^T\right) + \vec{b}\right)$$

$$\vec{h}_{ij} = \vec{z}_l \odot \vec{h}_{i,j-1} + \vec{z}_t \odot \vec{h}_{i-1,j} + \vec{z}_d \odot \vec{h}_{i-1,j-1} + \vec{z}_i \odot \vec{h}'_{ij}$$

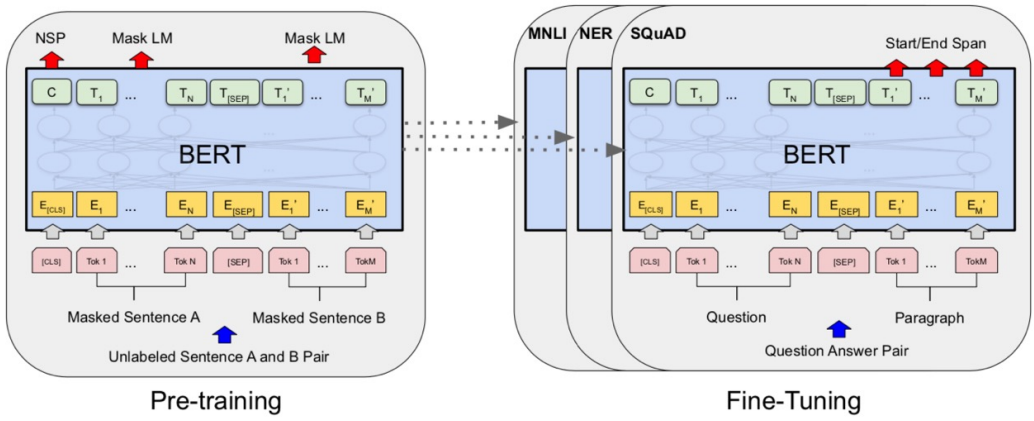


FIGURE 14 Architecture of BERT in pre-training and fine-tuning (Devlin et al., 2019)

3. Linear Scoring Function: Final matching score of two given sentences is computed by the equation: $M(S_1, S_2) = W^{(s)}h_{mn} + b^{(s)}$ where h_{mn} is the global interaction between two sentences, and $W^{(s)}$ and $b^{(s)}$ are network parameters.

Devlin et al. (2019) proposed Bidirectional Encoder Representations from Transformers (BERT) model which is a language modeling neural network. BERT has a multi-layered bidirectional architecture, in which each layer is a transformer encoder. The transformer was proposed by Vaswani et al. (2017) and has an encoder-decoder architecture. The same decoder segment of the transformer model is used in BERT. BERT is used in a wide range of NLP downstream tasks, including QA, natural language inference, and text classification for capturing the textual dependency among given sequences. BERT is pre-trained on large corpora by two different approaches, namely masked language model and next sentence prediction, and then fine-tuned on each specific downstream task based on the application. The architecture of BERT including both pre-training and fine-tuning steps is shown in Figure 14. Input of BERT is a sequence of input representation of words. Input representation of each word is built by summing the word embedding, segment embedding, and position embedding as shown in Figure 15. In the QA domain, a (CLS) token is used in the first position of the sequence, then the question and the candidate answer followed by a (SEP) token are placed in the sequence, respectively. The output of BERT is an encoded representation for each token. BERT is also used in question answering task and As BERT uses cross-match attention between question and answer sentences it is considered as an interaction-based model.

Garg et al. proposed the TANDA model, which utilizes BERT and RoBERTa pre-trained language models for modeling the dependency between two sequences of sentences for Answer Sentence Selection (AS2). The small size of data for fine-tuning BERT may lead to an unstable, and noisy model. For mitigating this problem, they have used two distinct fine-tuning steps for AS2. In the first fine-tuning step which is performed on a large corpus for the AS2 task, BERT is transferred to an AS2 model instead of being a language model only. Then the model is adapted to a specific domain of question types by fine-tuning the model on the target dataset. The architecture of the TANDA model is shown in Figure 16. A pair of a question and an answer is attached with a $[SEP]$ token and passed to the BERT model. The encoded representation of $[CLS]$ token is passed to a fully-connected layer followed by a sigmoid function for predicting the matching score of the given question and answer pair.

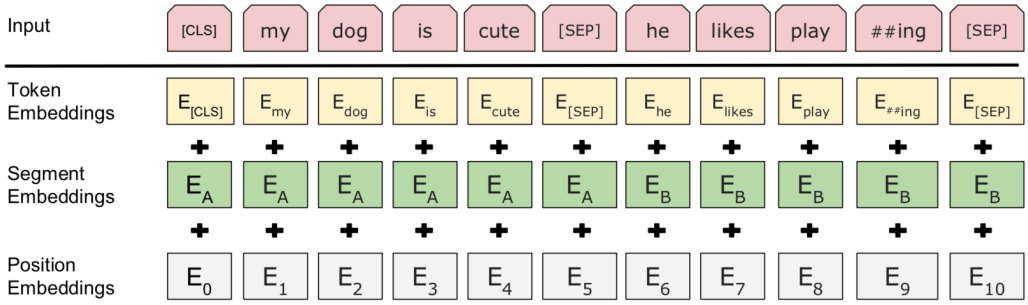


FIGURE 15 Building input representation of BERT (Devlin et al., 2019)

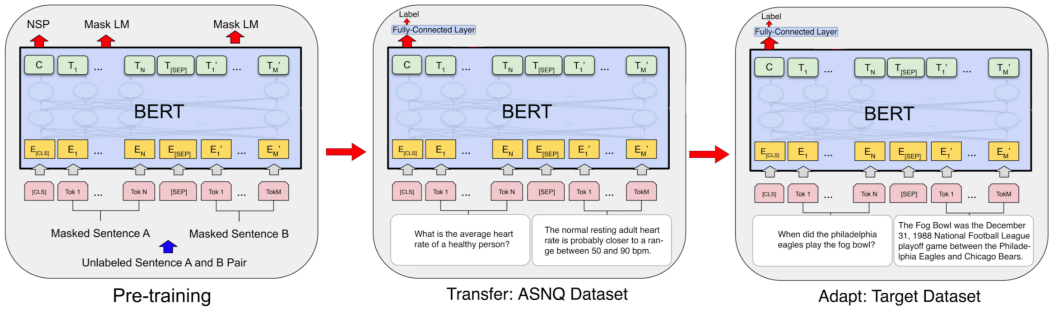


FIGURE 16 Architecture of the TANDA model (Garg et al.)

4.3 | Hybrid Models

He and Lin (2016) proposed a model for QA task which consists of four major components. Architecture of the model is shown in Figure 17. Different components of this model are described below:

1. Context modeling: This is the first component and uses a BiLSTM for modeling context of each word.
2. Pairwise word interaction modeling: This component compares two hidden states of BiLSTM with Cosine, L2 Euclidean, and dot product distance measures:

$$CoU(h_1, h_2) = \{Cos(h_1, h_2), L2Euclid(h_1, h_2)), DotProduct(h_1, h_2))\} \tag{27}$$

   Output of this component is a cube with size $R^{13.|sent1|.|sent2|}$,where $|sent1|$ and $|sent2|$ are the size of the first and the second sentences, respectively. For each pair of words 12 different similarity distances and one extra padding are considered.
3. Similarity focus: In this layer word interactions are assigned weights by maximizing the weight of the important word interactions. The output of this component is a cube named FocusCube and words identified as important have more weight in this cube.

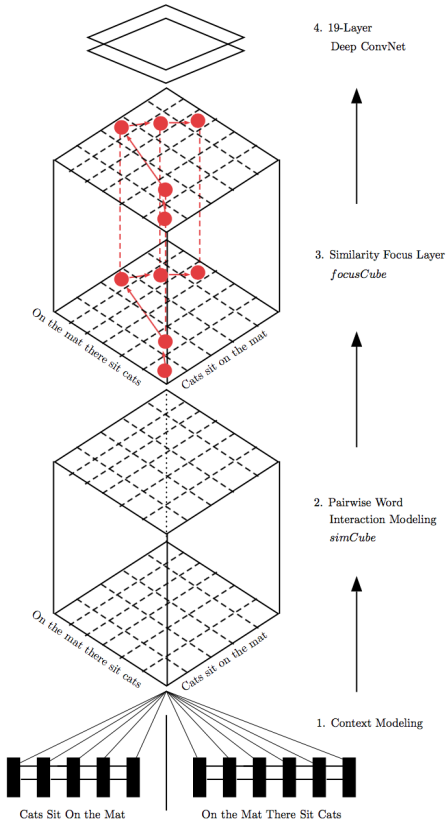


FIGURE 17 Architecture of He and Lin (2016) model

4. Similarity classification: In this layer, CNN is used for finding the patterns of strong pairwise word interactions. Question and answer sentences in FocusCube are fed to this layer and a similarity score is computed.

Wan et al. (2016a) proposed MV-LSTM for matching two sentences using representation of different positions of sentences. As shown in Figure 18, representation of different positions in each sentence is created and multiple tensors are created by calculating the interaction between different positions of these two sentences with different similarity metrics. Then a k-max pooling layer and a multi-layered LSTM are used for modeling the matching score of two given sentences. Architecture of MV-LSTM is explained in more details in following three steps:

1. Positional Sentence Representation: Positional sentence representation or representation of sentence at one position is obtained by BiLSTM . BiLSTM is used for capturing the long and short-term dependencies in one sentence. An LSTM layer similar to the implementation used in (Graves et al., 2013) is used here. Given a sentence $S = (x_0, x_1, ..., x_T)$, LSTM represents position t of the sentence h_t as follows:

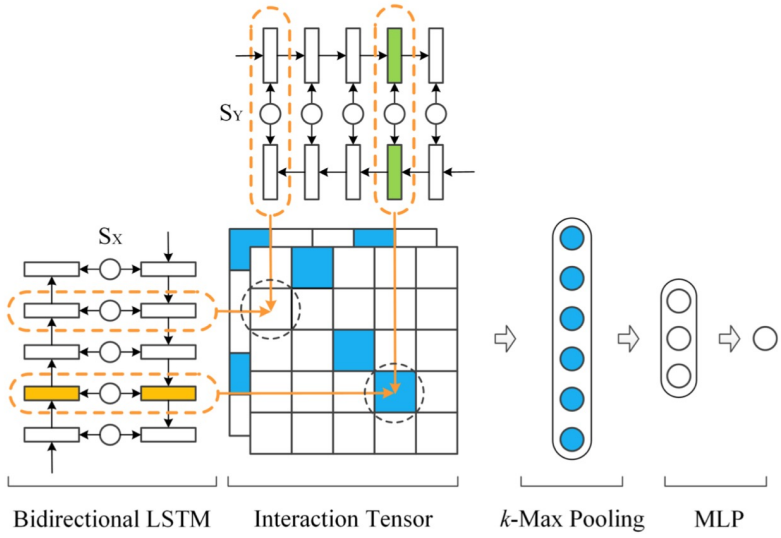


FIGURE 18 Architecture of MV-LSTM model (Wan et al., 2016a)

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i) \tag{28}$$

$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f)$$

$$c_t = f_t c_{t-1} + i_t \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c)$$

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o)$$

$$h_t = o_t \tanh(c_t)$$

Utilizing the BiLSTM layer, two different representations $\overrightarrow{h}_t$ and $\overleftarrow{h}_t$ are generated for each position and final representation of each position is considered as the concatenation of these two representations $[\overrightarrow{h}_t . \overleftarrow{h}_t]$.

2. Interactions between two sentences: Cosine, bilinear, and tensor similarity functions are used in this step for modeling the interaction between two positions. Bilinear function which captures more complicated interactions compared to cosine is as follows:

$$S(u, v) = u^t M v + b \tag{29}$$

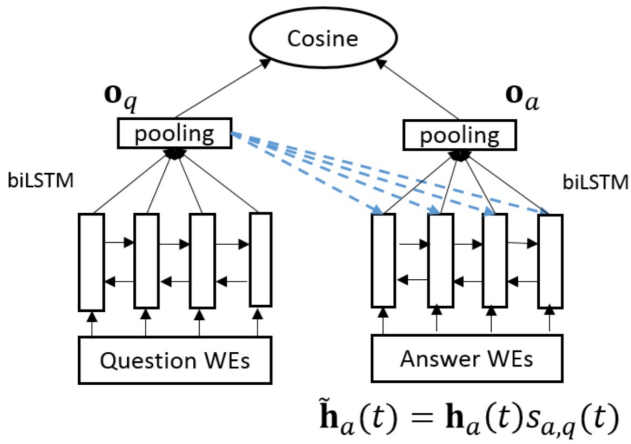


FIGURE 19 Architecture of Attentive-LSTM model (Tan et al., 2016)

where b is bias and M is a matrix for reweighting u and v in different dimensions. Tensor function models the interaction between two vectors more powerfully. It uses the following equation for modeling the interaction:

$$s(u, v) = f\left(u^T M^{[1:c]} v + W_{uv}\begin{bmatrix} u \\ v \end{bmatrix} + b\right) \tag{30}$$

where W_{uv} and b are parameters, M_i is one slice of the tensor parameter, and f is rectifier function. Output of the cosine and bilinear similarities are interaction matrices while the output of the tensor layer is an interaction tensor.

3. Interaction aggregation: The third step uses the interaction between different positional sentence representations in order to measure the matching score of two given sentences. K-max pooling is applied to extract a vector q which includes the top k values of a matrix, or the top k values of each slice of the tensor. A new representation r is obtained by feeding the output of k-max pooling into a fully connected hidden layer. And finally, the matching score s is obtained by the following function:

$$r = f(W_r q + b_r), s = W_s r + b_s \tag{31}$$

where W_r and W_s are model parameters and b_s and b_r are biases.

Tan et al. (2016) proposed attentive LSTM, a variant of QA-LSTM model, for mitigating some problems of two other variants of QA-LSTM: convolutional-based LSTM and convolutional pooling LSTM. These two previous models, which are described in section 4.1, suffer from a common issue that happens when the answer is very long and contains a lot of not related words to the question sentence. Attention mechanism by considering the question in constructing the answer sentence's representation can solve this issue. In this model, the attention mechanism works by learning weights for hidden vectors of BiLSTM. As shown in Figure 19, output of the BiLSTM is multiplied by a softMax weight,

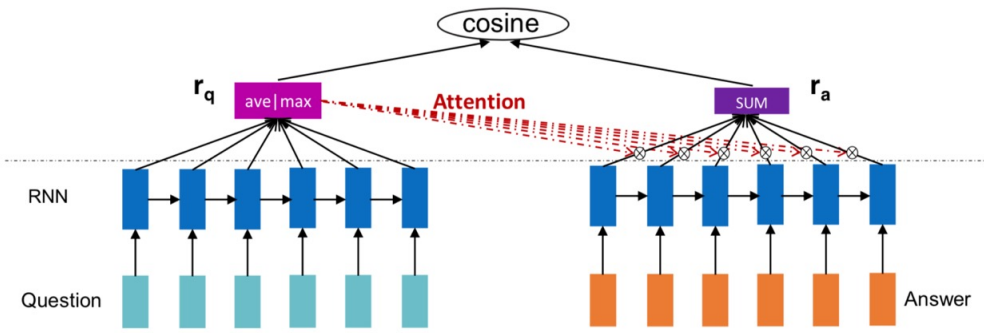


FIGURE 20 Architecture of OARNN model (Wang et al., 2016a)

which is obtained from the question representation. The model gives more weight to each word of the answer, based on the information from question representation. Finally representation of the answer sentence is obtained by the following equations:

$$m_{a,q}(t) = W_{am} h_a(t) + W_{qm} o_q \tag{32}$$

$$s_{a,q}(t) \propto \exp\left(w_{ms}^T \tanh\left(m_{a,q}(t)\right)\right)$$

$$\widetilde{h}_a(t) = h_a(t) s_{a,q}(t)$$

where $h_a(t)$ is the output vector of answer BiLSTM at time step t, o_q is question representation, W_{am}, W_{qm}, and W_{ms} are attention parameters, and $\widetilde{h}_a(t)$ indicates the attention-based representation of $h_a(t)$.

Wang et al. (2016a) proposed four inner attention-based RNN models. These models try to mitigate the attention bias problem which traditional attention-based RNN models suffer from. In the following, first a traditional attention-based RNN model and then four variants of inner attention-based RNN (IARNN) models are described.

1. Traditional attention based RNN models (OARNN): In OARNN first of all an RNN block is used for encoding sentences, and then attention weights from question embedding are used in generating answer sentence's representation. This type of attention mechanism, which is done after learning embeddings, biases toward the later hidden states, because they contain more information than the nearer ones about the sentence. Architecture of OARNN is shown in Figure 20. This model is named OARNN (stands for outer attention-based RNN) as it adds the attention layer after RNN block. Last hidden layer or average of all hidden states are considered as representation of the question sentence, where the representation of answer is obtained by using attention weights from question representation.
2. Inner attention-based RNNs (IARNN): IARNN models are proposed to mitigate the bias problems of OARNN in generating the representation of the answer sentence. In these models, the attention weights are added before that

RNN blocks generate hidden layers. The architecture of four different IARNN models is described in the following.

a. IARNN-WORD: Architecture of this model is shown in Figure 21. Representation of each word is generated using the question attention weights, then the whole sentence's representation is obtained by using the RNN model. GRU is chosen among RNN blocks because it has fewer parameters and trains fast. Representation of the sentence is generated by a weighted average of the hidden states h_t.

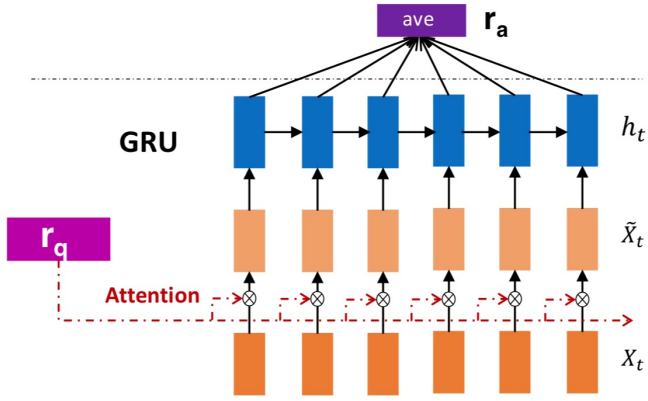


FIGURE 21 Architecture of IARNN-word model (Wang et al., 2016a)

b. IARNN-Context: Due to the inability of IARNN-WORD model in capturing the multiple related words, in IARNN-Context, contextual information of answer sentence is fed into the attention weights. Architecture of this model is shown in the Figure 22.

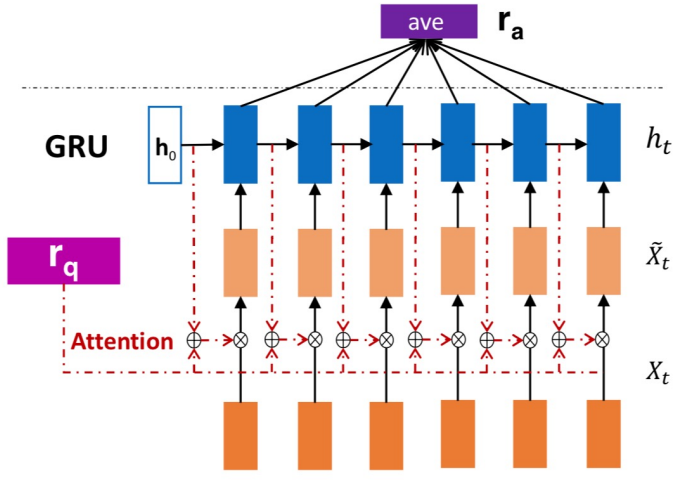


FIGURE 22 Architecture of IARNN-Context model (Wang et al., 2016a)

c. IABRNN-GATE: As GRU gates control the flow of the information in hidden stages, attention information is fed

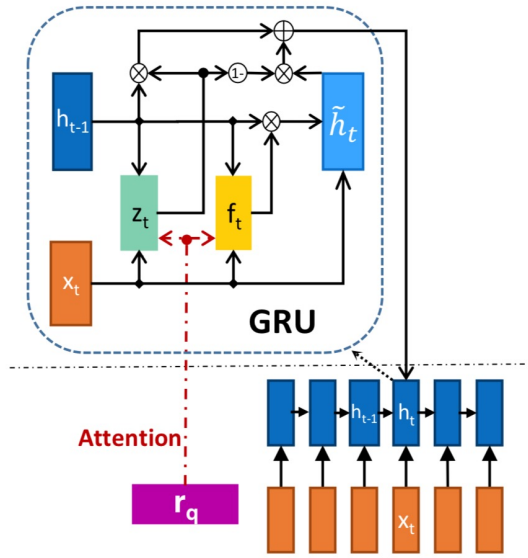


FIGURE 23 Architecture of IABRNN-GATE model (Wang et al., 2016a)

to these gates. Architecture of this model is shown in the Figure 23.

d. IARNN-OCCAM: This model is named after the Occam's Razor which says: "Between the whole words set, the fewest number of words which can represent the sentence must be chosen". Based on the type of question, a different number of relevant words to the question are required for answering the question. For example "what" and "where" questions need a smaller number of relevant words than "why" and "how" questions in answer sentence. This issue is handled in IARNN-OCCAM by using a regulation value. Therefore more sparsity should be imposed on the summation of the attention in "what" and "where" questions and a smaller number should be assigned to the regulation value in "why" and "how" questions. This regulation model just could be used in IARNN-context and IARNN-word models.

Yin et al. (2016) proposed four different attention-based variants of BCNN (ABCNN) for text-matching task. In the following, the architecture of these ABCNN models is described.

- Attention-based CNN (ABCNN): Three different attention-based models are proposed in this work. In ABCNN-1, which is shown in Figure 24, an attention matrix A is generated by comparing each unit of two feature maps. Let S_1 and S_0 be feature maps representing a sentence. Each row in the matrix A shows the attention distribution of the corresponding unit in S_0 respect to S_1, and each column of A represents the attention distribution of the corresponding unit in S_1 respect to S_0. Then matrix A is transformed into two attention feature map matrices with the same dimension of the representation feature map. According to Figure 24 representation feature map and attention feature map are fed to the convolution layer as order-3 tensors. Given representation of two feature maps of sentences $i = 0, 1$ ($F_{i,r} \in R^{d \times s}$), each cell in attention matrix ($A \in R^{s \times s}$) is computed as follows:

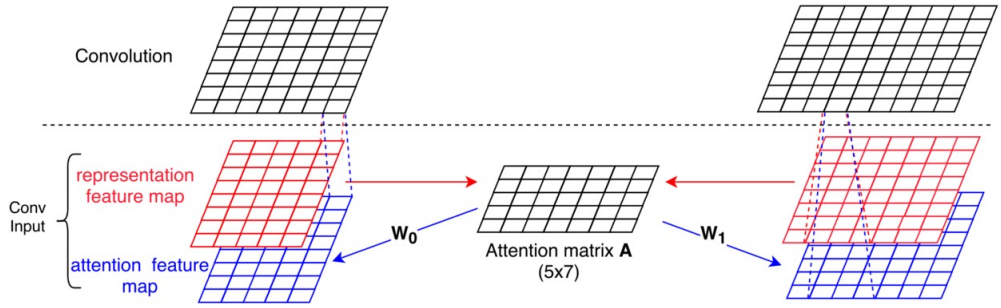


FIGURE 24 Architecture of ABCNN-1 model (Yin et al., 2016)

$$A_{i,j} = \text{match-score}\left(F_{0,r}[:, i], F_{1,r}[:, j]\right) \tag{33}$$

where $\frac{1}{(1+|x-y|)}$ is match-score function for inputs x and y. Attention matrix A is converted to two given feature maps ($F_{0,a}$ and $F_{1,a}$) by the following equations where $W_0, W_1 \in R^{d \times s}$ are model parameters to be learned.

$$F_{0,a} = W_0 \cdot A^{\top}, \quad F_{1,a} = W_1 \cdot A \tag{34}$$

A higher-level representation for the corresponding sentence is generated by passing these matrices to the convolution layer.

- ABCNN-2: In this architecture (shown in Figure 25) attention mechanism is applied to the output of convolutional layers. Each cell in the attention matrix A is calculated by comparing corresponding units from convolution outputs. Each row in the convolution output matrix represents one unit of the given sentence. The attention weight of each unit is computed by summing all the attention values of that unit. Attention weight of unit j in sentence i is shown with $a_{i,j}$ and computed by:

$$a_{0,j} = \sum A[j, :], a_{1,j} = \sum A[:, j] \tag{35}$$

then the new feature map $F^p_{i,r} \in R^{d \times s_i}$ is calculated with $w - ap$ pooling as follows:

$$F^p_{i,r}[:, j] = \sum_{k=j:j+w} a_{i,k} F^c_{i,r}[:, k], \quad j = 1 \ldots s_i \tag{36}$$

- ABCNN-3: As it is shown in Figure 26, ABCNN-3 combines two previous models by employing the attention mechanism before and after the convolution layer. The output of the convolution layer has a larger granularity than its input. That means if the input of the convolution layer has a word-level granularity, then its output has phrase-level granularity. Therefore in the ABCNN-1 model, attention mechanism is employed on a smaller level of

granularity than the ABCNN-2 model, and in the ABCNN-3, it is employed on two different levels of granularity.

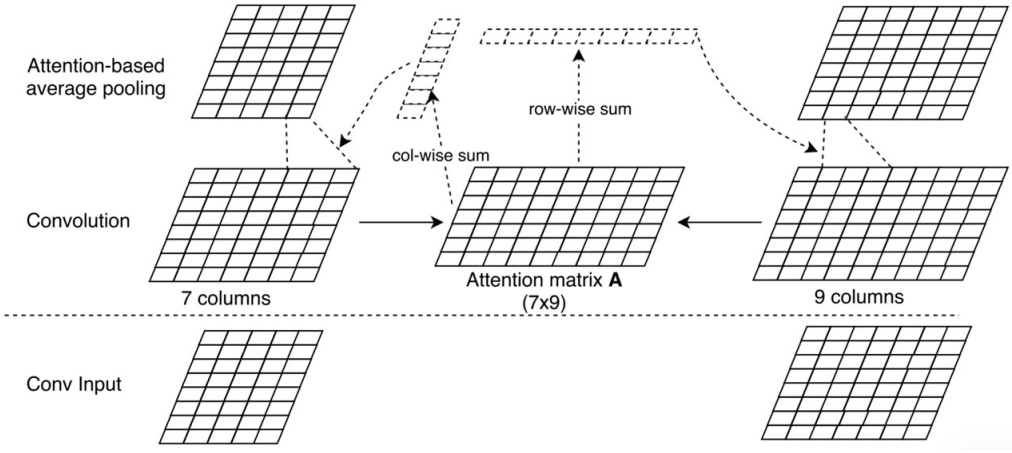


FIGURE 25 Architecture of ABCNN-2 model (Yin et al., 2016)

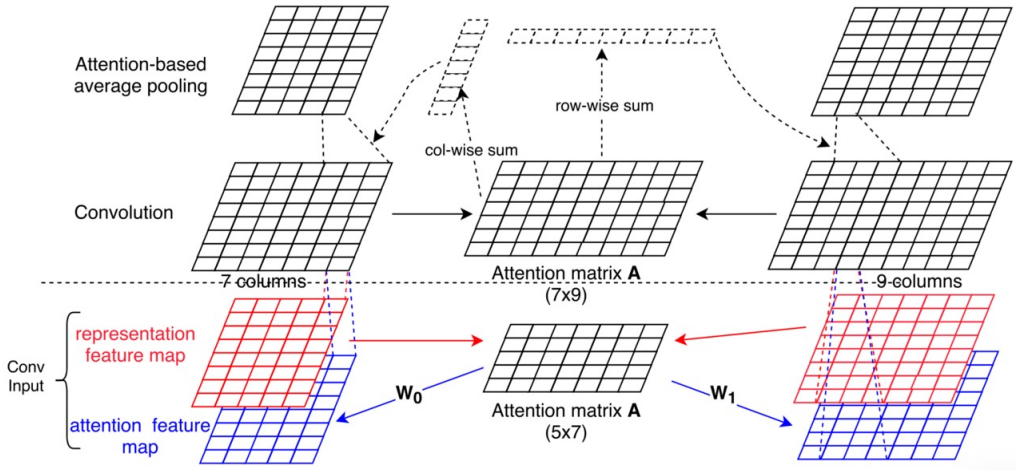


FIGURE 26 Architecture of ABCNN-3 model (Yin et al., 2016)

Bian et al. (2017) proposed a model which estimates the relevance score $P(y|Q, A)$ between question (Q) and answer (A). As we see in Figure 27, this model consists of four major layers. First, word representation of each sentence is passed to an attention layer and then the output of the attention layer is compared by sentence representation. The output of comparison layers is passed to a CNN layer for aggregating and then the matching score of two given sentences is obtained in this layer. These layers are described below.

1. Word representation layer: Word representations of question $q = (q_1, ..., q_{l_q})$ and answer $a = (a_1, ..., a_{l_a})$ are fed

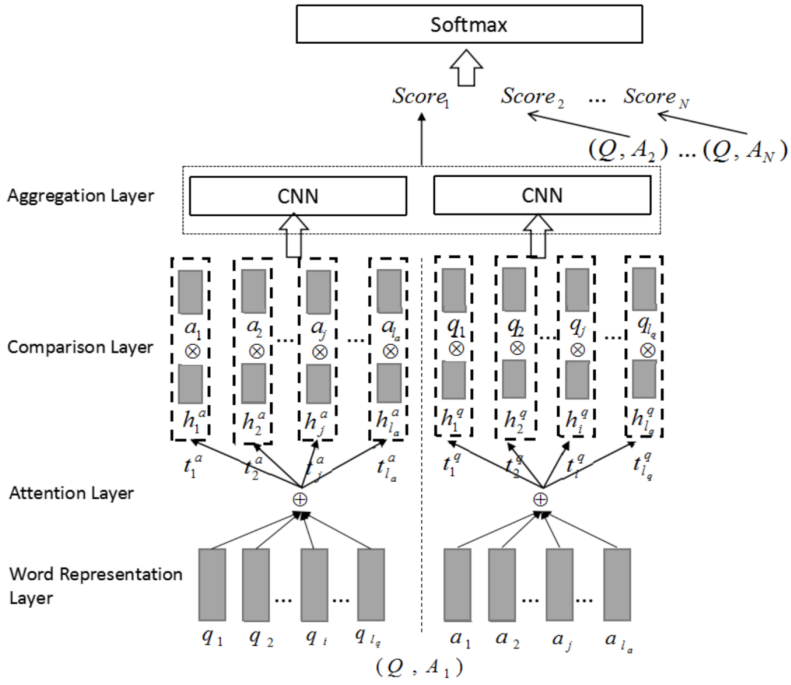


FIGURE 27 Architecture of Bian et al. (2017) model

to the attention layer.

2. Attention layer: The aim of applying the attention layer is finding the relevance between local text substructure of question and answer pairs. h_j^a and h_i^q are obtained in this layer by the following equations:

$$w_{ij}^a = \frac{\exp\left(e_{ij}\right)}{\sum_{k=1}^{\ell_q} \exp\left(e_{kj}\right)}, \quad w_{ij}^q = \frac{\exp\left(e_{ij}\right)}{\sum_{k=1}^{\ell_a} \exp\left(e_{ik}\right)} \tag{37}$$

$$h_j^a = \sum_{i=1}^{\ell_q} w_{ij}^a q_i, \quad h_i^q = \sum_{j=1}^{\ell_a} w_{ij}^q a_j$$

where $e_{ij} = q_i.a_j$ and w indicate attention weight. This attention model has two problems: First, only a small number of interactions between two sentences are related and the semantic relation is being ambiguous by considering irrelevant interactions. It is proper to consider just relevant interactions. Second, if one token from answer sentence doesn't have any semantic matching with all the words from the question sentence, it is better to omit that token. For tackling the mentioned problems, two filtering approaches, which are called k-max attention and k-threshold attention, are proposed. Implementation of these two filtering models in computing h_{ja} is described below.

- K-max attention: This filtering model helps to discard irrelevant fragments, by sorting w in decreasing order and preserving the top k weights and setting other weights to zero.

- K- threshold attention: This filtering just preserves attention weights which are larger than K. This filtering works by omitting the units with no semantic matching in another sentence.

3. Comparison: Each sentence and weighted version of the other sentence which is obtained in attention layer are compared in this layer. for example a_j is compared with h_{ja} by using comparison function f as follows:

$$t_j^a = f\left(a_j, h_j^a\right) = a_j \otimes h_j^a \tag{38}$$

where t_{ja} represents the comparison result.

4. Aggregation: Comparison vectors of the previous layer for each sentence are aggregated using a one-layer CNN, and finally, the relevance score between question and answer sentences is computed by the following equation:

$$r_a = \text{CNN}\left(\left[t_1^a, \ldots, t_{l_a}^a\right]\right), \quad r_q = \text{CNN}\left(\left[t_1^q, \ldots, t_{l_q}^q\right]\right) \tag{39}$$

$$\text{Score} = \left[r_a, r_q\right]^T W$$

Wang et al. (2017) proposed Bilateral Multi-Perspective Matching Model (BiMPM), a paraphrase-based method for the QA task. In BiMPM, each sample is represented with (P, Q, y), where $P = (p_1, p_2, ..., p_M)$ is answer sentence with length M, $Q = (q_1, q_2, ..., q_N)$ is question sentence with length N, and $y = 0, 1$ is a label indicating whether the answer is related to the question or not. $y = 1$ means P is a relevant answer for question Q and $y = 0$ means P is not a relevant answer for question Q. Figure 28 shows the architecture of BiMPM. BiMPM consists of five major layers which are described in the following.

1. Word representation layer: Each word is represented with a d-dimensional vector constructed by a word embedding and a character-composed embedding. Word embeddings are obtained from pre-trained GloVe (Pennington et al., 2014) or word2vec (Mikolov et al., 2013) embeddings. Character-composed embeddings are generated by feeding characters of words into an LSTM (Hochreiter and Schmidhuber, 1997).
2. Context representation layer: A BiLSTM is used in order to combine contextual information of a sentence with its representation.
3. Matching layer: In this layer, each contextual embedding of one sentence is compared with all the contextual representations of the other sentence using a multi-perspective matching operation. Also, question and answer sentences are matched in two directions. Multi-perspective cosine matching function is defined as:

$$M = f_m(v_1, v_2; W) \tag{40}$$

where v_1 and v_2 are d-dimensional vectors, $W \in R^{l \times d}$ is trainable parameter and each perspective is controlled by one row of W, and M is a l-dimensional vector. Each M_k is calculated as follows:

$$M_k = cosine(w_k \circ v_1, w_k \circ v_2) \tag{41}$$

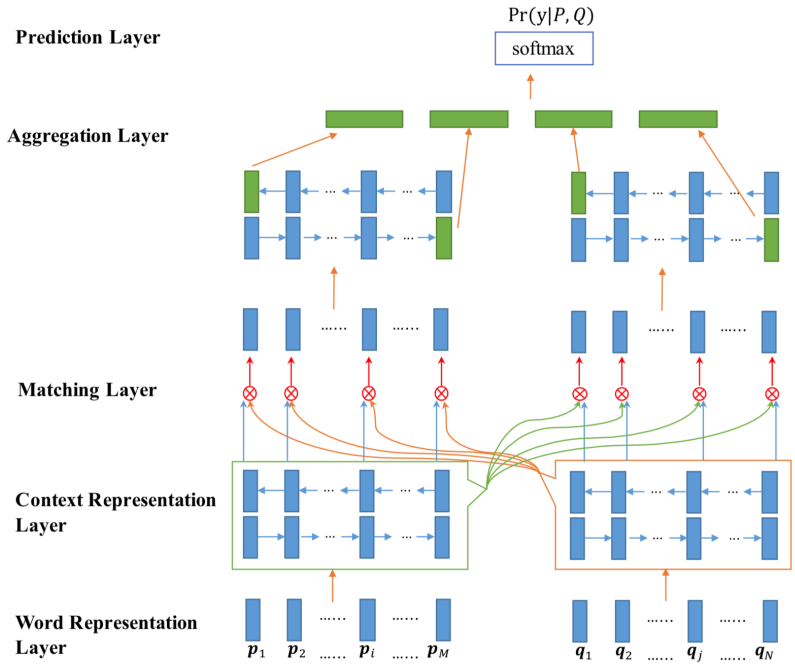


FIGURE 28 Architecture of BiMPM model (Wang et al., 2017)

let w_k be k-th row of W and $\circ$ be element-wise multiplication.

Four different matching strategies are proposed based on different f_m functions. These matching strategies are shown in Figure 29 and described just for one direction in the following.

- Full-Matching: Each forward time step representation of the answer sentence $\overrightarrow{h_i^p}$ is compared with every forward time step representations of the question sentence $\overrightarrow{h_N^q}$. This strategy is shown in Figure 29 (a).
- Maxpooling-Matching: Maximum similarity for each forward time step representation of the answer sentence with all the time steps of the forward representation of the question sentence is returned. This strategy is shown in Figure 29 (b).
- Attentive-Matching: Cosine similarity between each forward time step representation of the answer sentence $\overrightarrow{h_i^p}$ and each forward time step representation of the question sentence $\overrightarrow{h_i^q}$ is considered as attention weight. This strategy is shown in Figure 29 (c). Then the new representation of the question sentence (Q) called h_i^{mean} is generated by calculating the weighted average of its forward time steps representations by using attention weights. And finally, the matching vector for each time step representation of the answer sentence is calculated with its corresponding attentive vector h_i^{mean}.
- Max-Attentive-Matching: This strategy is different from an attentive-matching strategy just in generating attentive vector (h_i^{mean}). Attentive vector here is the contextual embedding with the highest similarity. This strategy is shown in Figure 29 (d). Finally, for each direction, all of these strategies are applied for each time-step and eight generated vectors are concatenated and considered as the matching in that direction.

4. Aggregation layer: Two sequences of matching vectors of both sentences, obtained from the matching layer, are fed into a BiLSTM. Then a fixed-length matching vector is obtained by concatenating the last four output vectors of two

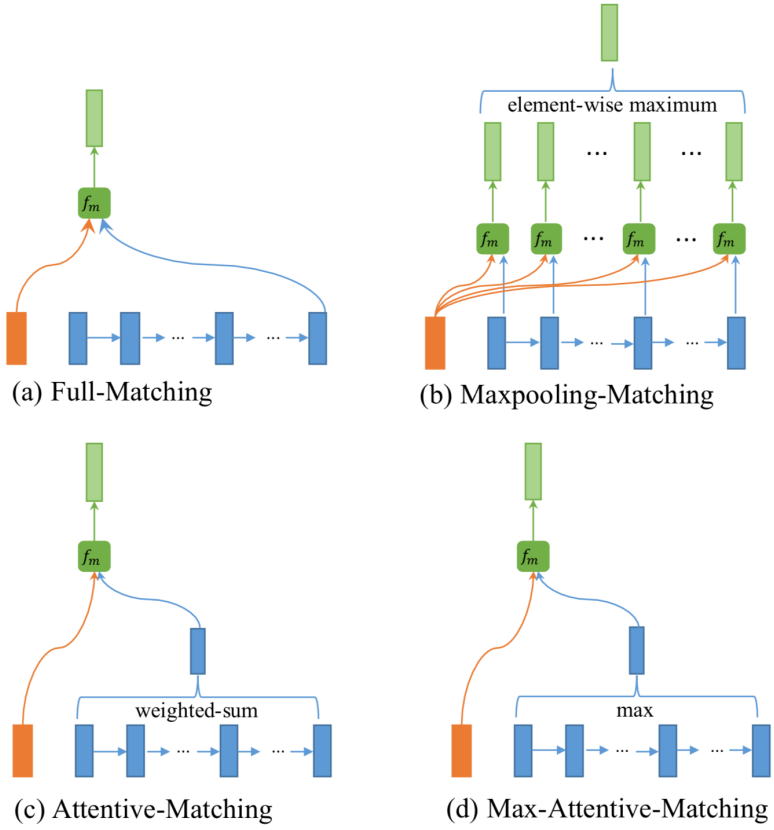


FIGURE 29 Architecture of BiMPM-matchings (Wang et al., 2017)

BiLSTMs.

5. Prediction layer: In this layer, $P(y|P, Q)$ is predicted using a two-layered feed-forward neural network followed by a softMax layer. The fixed-length matching-vector is fed to this layer.

Wang and Jiang (2017) proposed a compare-aggregate model for matching two sentences. This model for each pair of Q and A predicts a label y which shows whether the candidate answer A is a correct answer for question Q or not. According to the architecture of this model, which is shown in the Figure 30, this model consists four following major layer:

1. Preprocessing layer: This layer constructs an embedding for each word which represents the word and its contextual information. Q and A are inputs of this layer. A version of LSTM/GRU, which uses only the input gates, is applied for generating $\overline{Q} \in R^{l \times Q}$ and $\overline{A} \in R^{l \times A}$ matrices.

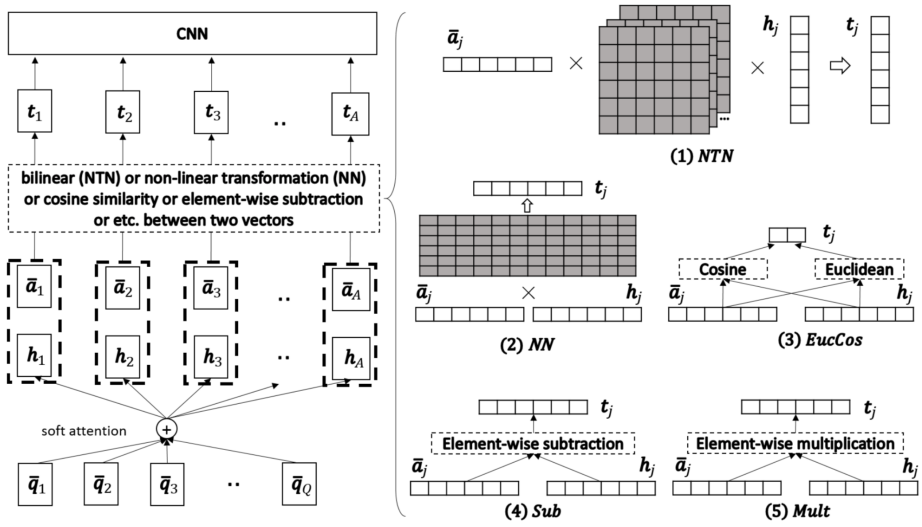


FIGURE 30 Architecture of Wang and Jiang (2017) (compare-aggregate) model

$$\begin{aligned}\overline{Q} &= \sigma\left(W^{\mathrm{i}}Q + b^{i} \otimes e_{Q}\right) \odot \tanh\left(W^{\mathrm{u}}Q + b^{\mathrm{u}} \otimes e_{Q}\right)\\ \overline{A} &= \sigma\left(W^{\mathrm{i}}A + b^{\mathrm{i}} \otimes e_{A}\right) \odot \tanh\left(W^{\mathrm{u}}A + b^{\mathrm{u}} \otimes e_{A}\right)\end{aligned} \tag{42}$$

where $W^{i}, W^{u} \in R^{l\times d}$ and $b^{i}, b^{u} \in R^{l}$ are parameters, and $(. \otimes e_{X})$ generates a matrix by repeating the vector on the left for X times.

2. Attention layer: This layer is applied to the output of the previous layer. Attention-weighted vector $H \in R^{l\times A}$ is obtained by the following equations. The j^{th} column of H indicates the part of Q that best matches the j^{th} word in A.

$$\begin{aligned}G &= \mathrm{softmax}\left(\left(W^{\mathrm{g}}\overline{Q} + b^{g} \otimes e_{Q}\right)^{\mathsf{T}} A\right)\\ H &= \overline{Q}G\end{aligned} \tag{43}$$

where $W^{g} \in R^{l\times l}$ and $b^{g} \in R$ are parameters, and $G \in R^{Q\times A}$ is the attention weight matrix.

3. Comparison layer: Embedding of each word a_j in the answer is matched with the corresponding attention weight h_j and the comparison result is indicated with vector t_j. In this work, six different comparison functions are introduced.
   - Neural Net (NN):

$$t_{j} = f\left(\overline{a}_{j}, h_{j}\right) = \mathrm{ReLU}\left(W\begin{bmatrix} a_{j} \\ h_{j}\end{bmatrix} + b\right) \tag{44}$$

- Neural Tensor Net (NTN):

$$t_j = f\left(\overline{a}_j, h_j\right) = \text{ReLU}\left(a_j^{\mathsf{T}} T^{[1\ldots l]} h_j + b\right) \tag{45}$$

- Euclidean distance or cosine similarity (EucCos):

$$t_j = f\left(\overline{a}_j, h_j\right) = \begin{bmatrix} \left\|a_j - h_j\right\|_2 \\ \cos\left(\overline{a}_j, h_j\right] \end{bmatrix} \tag{46}$$

- Subtraction (Sub):

$$t_j = f\left(\overline{a}_j, h_j\right) = \left(\overline{a}_j - h_j\right) \odot \left(\overline{a}_j - h_j\right) \tag{47}$$

- Multiplication (Mult):

$$t_j = f\left(\overline{a}_j, h_j\right) = \overline{a}_j \odot h_j \tag{48}$$

- Submult + NN:

$$t_j = f\left(\overline{a}_j, h_j\right) = \text{ReLU}\left(W \begin{bmatrix} \left(\overline{a}_j - h_j\right) \odot \left(\overline{a}_j - h_j\right) \\ \overline{a}_j \odot h_j \end{bmatrix} + b\right) \tag{49}$$

Among these comparison functions, NN and NTN do not capture the similarity well. EucCos may ignore some important information. Sub and Mult are similar to the Euclidean distance and Cosine similarity, and the last model is the combination of the Sub, Mult, and NN.

4. Aggregation: A one-layer CNN is used for combining t_j vectors. The output of the aggregation is $r \in R^{nl}$ which is used in the final classifier.

$$r = \text{CNN}\left(\left[t_1, \ldots, t_A\right]\right) \tag{50}$$

Tay et al. (2018) proposed Multi-Cast Attention Network (MCAN) for retrieval-based QA. Inputs of MCAN are two sentences: question q and document d sentences. As is shown in Figure 31, MCAN has five major layers. These layers are described in the following.

1. Input Encoder: Input sentences are fed to this network as one-hot encoded vectors and word embeddings are generated by passing through an embedding layer. Highway encoders like RNNs control the flow of the information by using a gating mechanism. A highway encoder layer is used for detecting important and not important words in a given sentence. A single highway network is formulated as:

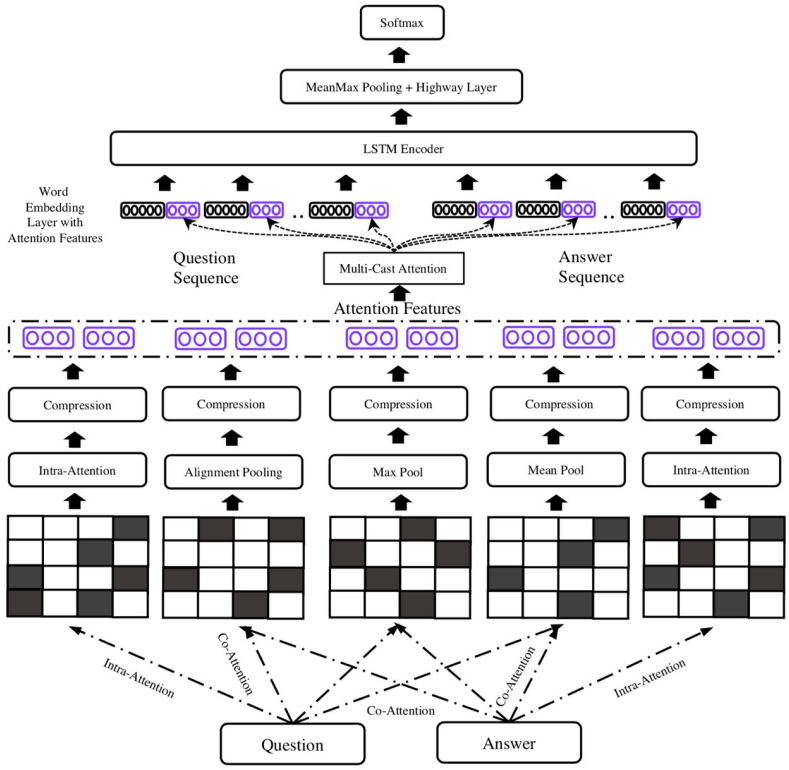


FIGURE 31 Architecture of MCAN model (Tay et al., 2018)

$$y = H(x, W_H) \cdot T(x, W_T) + (1 - T(x, W_T)) \cdot x \tag{51}$$

where $H(.)$ and $T(.)$ are one-layer affine transforms with ReLU and sigmoid activation functions, and $W_H, W_T \in R^{r \times d}$.

2. Co-Attention: In this layer, a similarity matrix which denotes the similarity between each pair of words across both sentences is learned by the following formulations:

$$s_{ij} = F(q_i)^T F(d_j) \tag{52}$$

$$s_{ij} = q_i^T M d_j$$

$$s_{ij} = F[q_i; d_j]$$

where F could be a multi-layered perceptron.

a. Extractive Pooling: Max-pooling and mean-pooling are two variants of this type. Formulation of these two poolings are as below:

$$\begin{aligned} q' &= \text{Soft}\left(\max_{col}(s)\right)^{\top} q \text{ and } d' = \text{Soft}\left(\max_{row}(s)\right)^{\top} d \\ q' &= \text{Soft}(\text{mean}(s))^{\top} q \text{ and } d' = \text{Soft}\left(\text{mean}_{row}(s)\right)^{\top} d \end{aligned} \tag{53}$$

where Soft is softMax function, and d' and q' are co-attentive representations of the document and question. Performance of these poolings varies on different datasets, but in general, max-pooling pays attention to words based on their maximum influence, and mean-pooling pays attention to words based on their total influence on the words of the other sentence.

b. Alignment pooling: Word pairs from two sentences are realigned in this pooling strategy. Co-attentive representations are learned as below:

$$d'_i := \sum_{j=1}^{\ell_q} \frac{\exp\left(s_{ij}\right)}{\sum_{k=1}^{\ell_q} \exp\left(s_{ik}\right)} q_j \text{ and } q'_j := \sum_{i=1}^{\ell_d} \frac{\exp\left(s_{ij}\right)}{\sum_{k=1}^{\ell_d} \exp\left(s_{kj}\right)} d_i \tag{54}$$

let d'_i be the sub-phrase of q which is aligned to d_i.

c. Intra-attention: Intra-attention attempts to represent long-term dependencies in one sentence. Representation of each sentence is learned regardless of the other sentence. So, it is applied on both the document and the question separately. Co-attentive representations are learned as below:

$$x'_i := \sum_{j=1}^{\ell} \frac{\exp\left(s_{ij}\right)}{\sum_{k=1}^{\ell} \exp\left(s_{ik}\right)} x_j$$

where x'_i is Intra-attention representation of x_i.

3. Multi-Cast Attention: This model utilizes all of the mentioned pooling functions. The following values are calculated for the output of each co-attention function.

$$f_c = F_c([\overline{x}; x]), f_m = F_c(\overline{x} \odot x), f_s = F_c(\overline{x} - x) \tag{55}$$

where $\overline{x}$ denotes the co-attention representation of x, and F_C is a compression function. In the above formulations, $\overline{x}$ and x are compared by three different operators for modeling the difference between $\overline{x}$ and x from different perspectives. Difference between $\overline{x}$ and x is an n-dimensional vector which is compressed by a compression function to a scalar. Three different compression functions: sum, fully-connected layer, and Factorization Machines (FM) are used. As is shown in Figure 31, given a document question pair, co-attention with three different poolings (1) mean-pooling, (2) Max-pooling, (3) alignment-pooling are applied on pair of question and document and (4) Intra-attention is applied on document and question separately. 12 scalars are generated for each word and concatenated with word embedding. Then each word w_i is represented as $w_i = [w_i; z_i]$ where $z \in R^{12}$ is output of multi-cast layer.

4. LSTM encoder: Casted representation of words of a sentence that are generated by multi-cast attention are fed to an LSTM encoder, and a meanMax pooling is applied to hidden states of the LSTM. Casted representations of words help the LSTM network with its knowledge about each sentence and between question and document, in extracting long-term dependencies.

$$\mathrm{H_i} = \mathrm{LSTM(u, i)}, \forall \mathrm{i} \in [1, 2, \ldots, 1] \tag{56}$$

$$\mathrm{H} = \mathrm{MeanMax}\,[\mathrm{h}_1 \ldots \mathrm{h}_l]$$

5. Prediction layer and optimization: Given representation of the document and the question, prediction is computed by using two-layer highway network and a softMax layer as follows:

$$\begin{aligned} y_{out} &= H_2\left(H_1\left(\left[x_q; x_d; x_q \odot x_d; x_q - x_d\right]\right)\right) \\ y_{\text{pred}} &= \text{softmax}\left(W_F \cdot y_{out} + b_F\right) \end{aligned} \tag{57}$$

where H_1, and H_2 are highway network layers with ReLU activation and $W_F \in R^{h\times 2}$, $b_F \in R^2$.

Yoon et al. (2019) proposed CompClip for predicting matching score (y) of given pair of question $(Q = q_1, \ldots, q_n)$ and answer $(A = a_1, \ldots, a_n)$. For improving the performance of CompClip, they have applied the transfer learning technique by training it on question-answering NLI (QNLI) corpus (Wang et al., 2018). They have also used pointwise learning to rank approach for training this model. The most prominent feature of their work is using the ELMo language model for achieving more meaningful contextual information of question and answer sentences. The architecture of CompClip consists of six layers, as is illustrated in Figure 32. These layers are described below.

1. Language model: Instead of using a word embedding layer, the Elmo language model (Peters et al., 2018) is used for extracting contextual information of the given sentence in a more efficient way. After applying the ELMo language model, a new representation of question and answer is denoted as L^Q and L^A, respectively.
2. Context representation: This part of model learns the weight W for extracting contextual information of given sentence and generating its contextual representation by following equations:

$$\begin{aligned} \overline{\mathbf{Q}} &= \sigma\left(\mathbf{W}^i\mathbf{Q}\right) \odot \tanh\left(\mathbf{W}^u\mathbf{Q}\right) \\ \overline{\mathbf{A}} &= \sigma\left(\mathbf{W}^i\mathbf{A}\right) \odot \tanh\left(\mathbf{W}^u\mathbf{A}\right) \end{aligned} \tag{58}$$

after applying Elmo language model, Q and A are replaced by L^Q and L^A, respectively.

3. Attention: Attentional representation of question H^Q and answer H^A sentences are generated utilizing dynamic-clip attention (Bian et al., 2017) as follows:

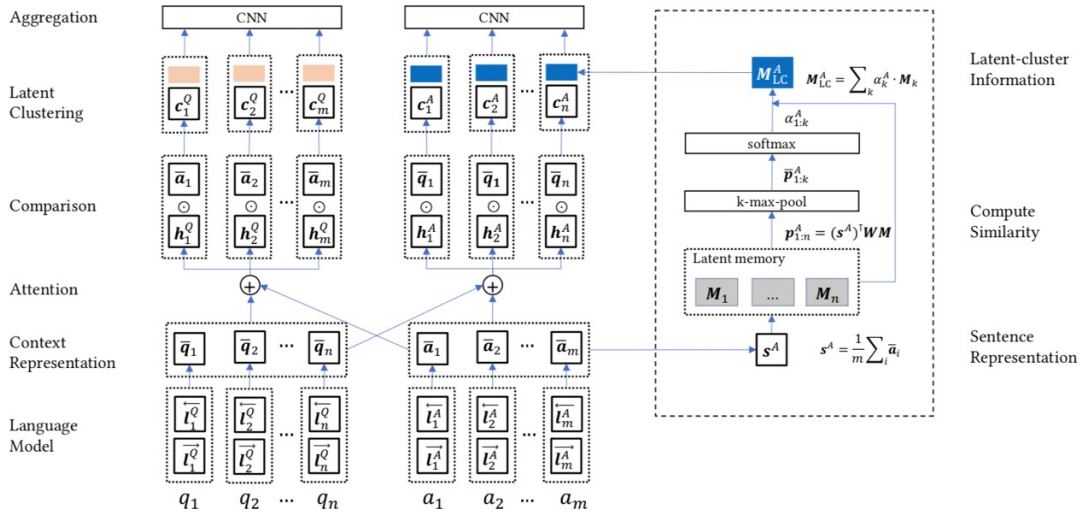


FIGURE 32 Architecture of CompClip model (Yoon et al., 2019)

$$\begin{aligned} \mathbf{H}^{Q} &= \overline{\mathbf{Q}} \cdot \text{softmax}\left(\left(\mathbf{W}^{q}\overline{\mathbf{Q}}\right)^{\top}\overline{\mathbf{A}}\right) \\ \mathbf{H}^{A} &= \overline{\mathbf{A}} \cdot \text{softmax}\left(\left(\mathbf{W}^{a}\overline{\mathbf{A}}\right)^{\top}\overline{\mathbf{Q}}\right) \end{aligned} \tag{59}$$

4. Comparison: Each term form question and answer sentences are compared by element-wise multiplication of question and answer representations with H^A and H^Q, respectively.

$$\begin{aligned} \mathbf{C}^{Q} &= \overline{\mathbf{A}} \odot \mathbf{H}^{Q}, \left(\mathbf{C}^{Q} \in R^{l \times A}\right) \\ \mathbf{C}^{A} &= \overline{\mathbf{Q}} \odot \mathbf{H}^{A}, \left(\mathbf{C}^{A} \in R^{l \times Q}\right) \end{aligned} \tag{60}$$

5. Aggregation layer: For aggregating outputs of comparison layer, a CNN with n-types of filters is employed. Aim of this layer is computing the matching score ($score$) between question and answer as follows:

$$\mathbf{R}^{Q} = \text{CNN}\left(\mathbf{C}^{Q}\right), \mathbf{R}^{A} = \text{CNN}\left(\mathbf{C}^{A}\right) \tag{61}$$

$$\text{score} = \sigma\left(\left[\mathbf{R}^{Q}; \mathbf{R}^{A}\right]^{\top}\mathbf{W}\right)$$

6. Latent clustering: In order to improve performance of model, latent clustering information of corpus is used for obtaining cluster information of question and answer sentences. Latent clustering information of sentence s is

generated using the following equations:

$$\mathbf{p}_{1:n} = \mathbf{s}^{\top}\mathbf{W}\mathbf{M}_{1:n} \tag{62}$$

$$\overline{\mathbf{p}}_{1:k} = k - \max - \text{pool}\left(\mathbf{p}_{1:n}\right)$$

$$\alpha_{1:k} = \text{softmax}\left(\overline{\mathbf{p}}_{1:k}\right)$$

$$\mathbf{M}_{\text{LC}} = \Sigma_k \bar{\alpha}_k \mathbf{M}_k$$

where $M_{1:n} \in R^{d' \times n}$ is latent memory and $W \in R^{d \times d'}$ is parameter of the model. Latent clustering function f is applied on context representation of question and answer sentences and cluster information of question and answer, M_{LC}^{Q} and M_{LC}^{A} vectors, are generated, respectively. M_{LC}^{Q} and M_{LC}^{A} are concatenated with C^{Q} and C^{A} which results in generating C_{new}^{Q} and C_{new}^{A} representations, respectively. C_{new}^{Q} and C_{new}^{A} could be considered as input of aggregation layer.

$$\mathbf{M}_{\text{LC}}^{Q} = f\left(\left(\Sigma_i \bar{q}_i\right)/n\right), \bar{q}_i \subset \overline{\mathbf{Q}}_{1:n} \tag{63}$$

$$\mathbf{M}_{\text{LC}}^{A} = f\left(\left(\Sigma_i \bar{a}_i\right)/m\right), \bar{a}_i \subset \overline{\mathbf{A}}_{1:m}$$

$$\mathbf{C}_{\text{new}}^{Q} = \left[\mathbf{C}^{Q}; \mathbf{M}_{\text{LC}}^{Q}\right], \mathbf{C}_{\text{new}}^{A} = \left[\mathbf{C}^{A}; \mathbf{M}_{\text{LC}}^{A}\right]$$

Yang et al. (2019) presented the RE2 model, which is a text matching model. RE2 is a simple and fast model for extracting more effective features from the two given sequences. The name RE2 comes from augmented features used in this model: residual vectors, embedding vectors, and encoded vectors. RE2 includes two distinct components with shared parameters for processing each of the sequences as well as one prediction layer on top of these two components. The component for processing the given text includes N similar blocks. The architecture of RE2 is shown in Figure 33. Each component of this model is described in the following:

1. Augmented Residual Connections: Blocks are connected to each other by residual connections. Input to the first block is word embeddings. The input of other blocks are constructed as follows:

$$x_i^{(n)} = \left[x_i^{(1)}; o_i^{(n-1)} + o_i^{(n-2)}\right] \tag{64}$$

where $x_i^{(n)}$ is the i-th token of the input of the n-th block for $n > 2$, and $o_i^{(n)}$ is the i-th token of the output of the n-th

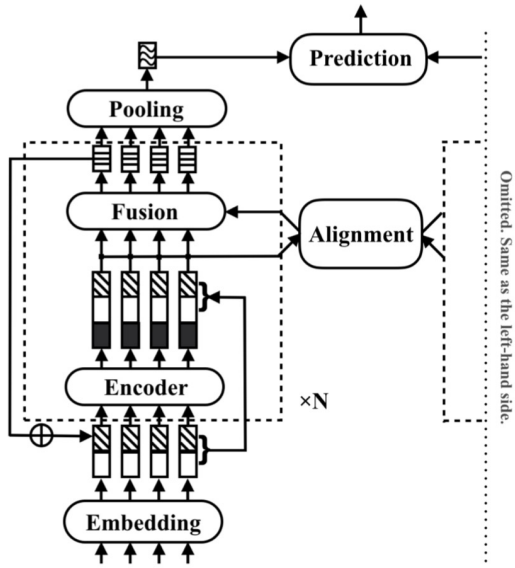


FIGURE 33 Architecture of the RE2 model(Yang et al., 2019)

block. As we see, pure embedding vectors, aligned features which are fed through the previous block, and extracted features from the encoder are passed to the Alignment layer.

2. Alignment Layer: For aligning two text sequences a and b, a simple attention mechanism is used for modeling the attentional representation of two given sequences a' and b'.

$$e_{ij} = F(a_i)^T F(b_j) \tag{65}$$

$$a'_i = \sum_{j=1}^{l_b} \frac{\exp(e_{ij})}{\sum_{k=1}^{l_b} \exp(e_{ik})} b_j$$

$$b'_j = \sum_{i=1}^{l_a} \frac{\exp(e_{ij})}{\sum_{k=1}^{l_a} \exp(e_{kj})} a_i$$

where F is a feed-forward neural network.

3. Fusion Layer: This layer compares the aligned and simple representations from three perspectives and merges the results.

$$\bar{a}_i^1 = G_1([a_i; a'_i]) \tag{66}$$

$$\bar{a}_i^2 = G_2\left(\left[a_i; a_i - a'_i\right]\right)$$

$$\bar{a}_i^3 = G_3\left(\left[a_i; a_i \circ a'_i\right]\right)$$

$$\bar{a}_i = G\left(\left[\bar{a}_i^1; \bar{a}_i^2; \bar{a}_i^3\right]\right)$$

where G_1, G_2, G_3, and G are single-layer feed-forward networks.

4. Prediction Layer: Output of the two parallel components after deploying a pooling layer is used for creating the input of the multi-layer feed-forward network for predicting the score y as follows:

$$\hat{\mathbf{y}} = H\left([v_1; v_2]\right) \tag{67}$$

where v_1 and v_2 are outputs of two text matching components and H is a multi-layer feed-forward network.

Considering all the described models in this section, a brief overview of deep learning-based models is presented in Table 2.

TABLE 2 Overview of Deep Learning-based Models (I: Interaction-based, R: Representation-based, H: Hybrid)

Model	Name and Type	Main Idea	Datasets
Yu et al. (2014)	R	uses count of co-occurring words as an additional feature, captures more complex semantic features by CNN	TREC-QA
Wang and Nyberg (2015)	R	uses GBDT method for exact matching of the proper nouns and cardinal numbers, models contextual information by using BiLSTM	TREC-QA
Severyn and Moschitti (2015)	R	uses CNN for effectively learning the representation of a given sentence, has capability of including any additional similarity features	TREC-QA
(Tan et al., 2016)	R: QA-LSTM R: Conv-pooling LSTM R: Conv-based LSTM H: Attentive-LSTM	uses different combination of BiLSTM and CNN by attention mechanism for representing answer according to question	TREC-QA InsuranceQA
(Yin et al., 2016)	R: BCNN H: ABCNN H: ABCNN-2 H: ABCNN-3	uses attention in CNN with different approaches	WikiQA
(Yang et al., 2016)	I: aNMM-1 I: aNMM-2	uses attention for estimating the question term importance	TREC-QA
(Wan et al., 2016b)	I: Match-SRNN	uses a recursive method for modeling the interaction-matrix	Yahoo!

(Wan et al., 2016a)	H: MV-LSTM	uses BiLSTM for generating a rich model of context, matches different positions of two sentences	Yahoo!
(Wang et al., 2016a)	H: OARNN H: IARNN-WORD H: IARNN-Context H: IARNN-OCCAM H: IABRNN-GATE	introduces an attention mechanism for mitigating the bias problem, incorporates attention to different positions of an RNN	TREC-QA WikiQA InsuranceQA
(Tay et al., 2017)	R: HD-LSTM	uses circular correlation for modeling the relationship of question and answer sentences	TREC-QA Yahoo!
(Bian et al., 2017)	H	incorporates dynamic-clip attention for avoiding noise in attentional representation	TREC-QA WikiQA
(Wang et al., 2017)	H: BiMPM	encodes question and answer using BiLSTM and matches them in two directions using different matching strategies	TREC-QA WikiQA
(Wang and Jiang, 2017)	H	uses six different comparison functions	WikiQA InsuranceQA MovieQA
(Tay et al., 2018)	H: MCAN	attaches casted attention to word-level representation for hinting the LSTM	TREC-QA
(Yoon et al., 2019)	H: CompClip	uses pre-trained ELMo language model, transfer learning, and latent clustering	TREC-QA WikiQA
(Yang et al., 2019)	H: RE2	a simple and fast text-matching model for extracting augmented features	WikiQA
(Garg et al.)	I: TANDA	uses pre-trained language models: BERT and RoBERTa, adapts pre-trained model to QA by additional fine-tuning step	TREC-QA WikiQA

5 | DATASETS FOR QA

In this section, we describe five datasets that have been widely used for evaluating the QA tasks.

1. WikiQA is an open domain QA dataset (Yang et al., 2015). This dataset is collected from Bing query logs. Question-like queries that are issued by at least 5 different users and have clicked to Wikipedia pages are selected as questions in this dataset and the sentences of the summary section of the corresponding Wikipedia page are considered as candidate answers to the related question. The candidate answers are labeled as correct or incorrect with crowdsourcing and then the correct answer is selected. This dataset consists of 3047 questions and 1473 answers, more statistics about this dataset is presented in Table 3. A noted feature of WikiQA is that not all the questions in this dataset have the correct answer which makes it possible to use this dataset in answer triggering component. The answer triggering component's task is finding whether the question has an answer or not.

2. TREC-QA is collected from the Text REtrieval Conference (TREC) 8-13 QA dataset (Wang et al., 2007). Questions of TREC 8-12 are used as training dataset and questions of TREC 13 are used as development and test dataset. Statistics of TREC-QA dataset are presented in Table 4. TREC-QA contains two training datasets: TRAIN, and TRAIN-ALL. TRAIN dataset includes the first 94 questions of TREC 8-12 and its candidate answers are judged manually, while in the TRAIN-ALL dataset, correct answers are recognized by matching the answers with predefined patterns of the answer regular expression.

TABLE 3 Statistics of the WikiQA Dataset

	Train	Validation	Test	Total
# of questions	2118	296	633	3047
# of sentences	20360	2733	6165	29258
# of answers	1040	140	293	1473
Average length of questions	7.16	7.23	7.26	7.18
Average length of sentences	25.29	24.59	24.95	25.15
# of questions w/o answers	1245	170	390	1805

TABLE 4 Statistics of the TREC-QA Dataset

	Train-all	Train	Validation	Test
# Questions	1229	94	82	100
# QA pairs	53417	4718	1148	1517
% correct	12.00%	7.40%	19.30%	18.70%
#Answers/Q	43.46	50.19	14.00	15.17
judgement	automatic	manual	manual	manual

3. MovieQA is gathered from diverse data sources (Tapaswi et al., 2016) which is a unique feature of this dataset. It contains 14944 questions where each question is associated with five answers, including one correct answer and four deceiving answers. Statistics of this dataset is shown in Table 5.
4. InsuranceQA is a close domain dataset for QA in the insurance domain (Feng et al., 2015). Question/answer pairs of this dataset are collected from the internet. This dataset includes Train, Development, Test1, and Test2 parts. More detailed statistics about InsuranceQA is presented in Table 6.
5. Yahoo! Dataset is collected from Yahoo! Answers QA system. Yahoo! includes 142,627 question/answer pairs. But in the literature (Wan et al., 2016b,a) a subset of this dataset is selected as positive pair and for each question in this subset, four other negative pairs are constructed. The (question, answer) pairs in which the question and its answer length are between 5 to 50 words are selected as positive pairs (including 60564 pairs). Four negative answers for each question are selected by querying the whole answers set by the correct answer and selecting the four answers randomly among 1000 top retrieved answers.

TABLE 5 Statistics of the MovieQA Dataset

	Train	Validation	Test	Total
Movies with Plots and Subtitles				
# of Movies	269	56	83	408
# of QA	9848	1958	3138	14944
Q # words	9.3	9.3	9.5	9.3 ± 3.5
CA. # words	5.7	5.4	5.4	5.6 ± 4.1
WA. # words	5.2	5.0	5.1	5.1 ± 3.9

TABLE 6 Statistics of the InsuranceQA Dataset

	Train	Dev	Test1	Test2
Questions	12887	1000	1800	1800
Answers	18540	1454	2616	2593
Question Word Count	92095	7158	12893	12905

6 | EVALUATION

For evaluation of QA, three metrics are used: Mean Reciprocal Rank (MRR), Mean Average Precision (MAP), and accuracy. MRR indicates the ability of the system to answer a question. Reciprocal Rank (RR) for one question is inverse of the rank of the first correct answer, if there exists any correct answer, or zero if there exists no correct answer. RR of each question is computed and then the average of RRs is considered as MRR.

$$\text{MRR} = \frac{1}{|Q|}\sum_{i=1}^{|Q|}\frac{1}{\text{rank}_i} \tag{68}$$

$$\text{rank}_i = \begin{cases} \frac{1}{\text{the rank of the first correct answer}} \\ \text{(if a correct answer exists)} \\ 0. \\ \text{(if no carrect answer)} \end{cases}$$

The second metric is the MAP. MAP is the mean of the average of the precisions at each rank when a correct answer is detected (Diefenbach et al., 2018). Precision@k is the number of correct answers among the first k retrieved answers divided by k. Precision indicates how many of the answers are correct and is calculated as follows:

$$\text{MAP} = \frac{1}{Q}\sum_{q=1}^{Q} AP(q) \tag{69}$$

$$\text{AP(q)} = \frac{1}{k} * \sum_{k=1}^{K} p@k$$

$$\text{Precision@k} = \frac{\text{number of correct answers among first } k \text{ results}}{k}$$

where k is the rank of correctly retrieved answers.

Accuracy is used for evaluating datasets whose questions have just one correct answer or one label. Accuracy indicates how many of the questions are answered correctly.

$$\text{Accuracy} = \frac{\text{number of correct answered questions}}{\text{number of questions}} \tag{70}$$

7 | AVAILABLE RESULTS FROM THE LITERATURE

As mentioned in Section 5, WikiQA (Yang et al., 2015), TREC-QA (Wang et al., 2007), MovieQA (Tapaswi et al., 2016), InsuranceQA (Feng et al., 2015), and Yahoo! (Wan et al., 2016b,a) are the main datasets that have been used for evaluating text-based QA systems. In this section, we report the results of different models on the mentioned datasets to have a comparison of the quality of the state-of-the-art models. It has to be mentioned that considering a large number of models reviewed in this paper, it is not possible to reimplement all of them to have a comprehensive comparison and we only report the results based upon their availability.

TABLE 7 Results of Different Models on the WikiQA Dataset.

Model	Setting	MAP	MRR
Yih et al. (2013)	LCLR	0.5993	0.6086
Le and Mikolov (2014)	PV	0.5110	0.5160
Yang et al. (2015)	Word Count	0.5707	0.6266
	Wgt Word Count	0.5961	0.6515
	PV-Cnt	0.5976	0.6058
	CNN-Cnt	0.6520	0.6652
He and Golub (2016)		0.6930	0.7090
Miao et al. (2016)		0.689	0.707
Wang et al. (2016b)		0.706	0.723
Rao et al. (2016)		0.701	0.718
R: Yu et al. (2014)	CNN	0.6190	0.6281
R: Yin et al. (2016)	BCNN, one-conv	0.6629	0.6813
	BCNN, two-conv	0.6593	0.6738

I: Garg et al.	TANDA (BERT-b)	0.893	0.903
	TANDA (BERT-L)	0.903	0.912
	TANDA (RoBERTa-B)	0.889	0.901
	TANDA (RoBERTa-L)	**0.920**	**0.933**
H: Wang et al. (2016a)	IARNN-word	0.7098	0.7234
	IARNN-Occam(word)	0.7121	0.7318
	IARNN-context	0.7182	0.7339
	IARNN-Occam(context)	0.7341	0.7418
	IABRNN-GATE	0.7258	0.7394
	GRU	0.6581	0.6691
	OARNN	0.6881	0.7013
H: Yin et al. (2016)	ABCNN-1, one-conv	0.6810	0.6979
	ABCNN-1, two-conv	0.6855	0.7023
	ABCNN-2, one-conv	0.6885	0.7054
	ABCNN-2, two-conv	0.6879	0.7068
	ABCNN-3, one-conv	0.6914	0.7127
	ABCNN-3, two-conv	0.6921	0.7108
H: He and Lin (2016)		0.7090	0.7234
H: Bian et al. (2017)	listwise	0.746	0.759
	with k-max	0.754	0.764
	with k-threshold	0.753	0.764
H: Wang et al. (2017)	BiMPM	0.718	0.731
H: Wang and Jiang (2017)	NN	0.7102	0.7224
	NTN	0.7349	0.7456
	EucCos	0.6740	0.6882
	Sub	0.7019	0.7151
	Mult	0.7433	0.7545
	SUBMULT+NN	0.7332	0.7477
H: Yoon et al. (2019)	Comp-Clip	0.714	0.732
	Comp-Clip + LM	0.746	0.762
	Comp-Clip + LM + LC	0.764	0.784
	Comp-Clip + LM + LC +TL	0.834	0.848
H: Yang et al. (2019)	RE2	0.7452	0.7618

Table 7 reports the results of different representation-based (**R**), interaction-based (**I**), and hybrid (**H**) methods on the WikiQA dataset and compares it with other baseline models. As can be seen TANDA (Garg et al.) achieved the best results over all methods on the WikiQA dataset in MRR and MAP metrics. Although this interaction-based technique is the best state-of-the-art model in the field, comparing the rest of models, we can see that the general performance of hybrid models is better than interaction-based models and the next best results are all from the hybrid models.

Table 8 reports the results of different models which are described in Section 6 as well as their baseline methods on the TREC-QA dataset. According to this table, the TANDA (Garg et al.) model achieved the best result in the MAP.

TABLE 8 Results of Different Models on TREC-QA Dataset.

Model	Setting	MAP	MRR
Cui et al. (2005)		0.4271	0.5259
Wang et al. (2007)		0.6029	0.6852
Heilman and Smith (2010)		0.6091	0.6917
Wang and Manning (2010)		0.5951	0.6951
Yao et al. (2013)		0.6307	0.7477
Feng et al. (2015)	Architecture-II	0.711	0.800
Rao et al. (2016)		0.801	0.877
R:Yih et al. (2013)	LR	0.6818	0.7616
	BDT	0.6940	0.7894
	LCLR	0.7092	0.7700
R:Yu et al. (2014)	TRAIN bigram + count	0.7058	0.7800
	TRAIN-ALL bigram + count	0.7113	0.7846
	TRAIN unigram + count	0.6889	0.7727
	TRAIN-ALL unigram + count	0.6934	0.7677
	TRAIN unigram	0.5387	0.6284
	TRAIN-ALL unigram	0.5470	0.6329
	TRAIN bigram	0.5476	0.5476
	TRAIN-ALL bigram	0.5693	0.6613
R:Yang et al. (2015)	Word Count	0.5707	0.6266
	Wgt Word Count	0.5961	0.6515
R:Severyn and Moschitti (2015)	TRAIN	0.7329	0.7962
	TRAIN-ALL	0.7459	0.8078
R:Wang and Nyberg (2015)	BM25	0.6370	0.7076
	Single-Layer LSTM	0.5302	0.5956
	Single-Layer BiLSTM	0.5636	0.6304
	Three-Layer BiLSTM	0.5928	0.6721
	Three-Layer BiLSTM + BM25	0.7134	0.7913
R:Tan et al. (2016)	QA-CNN	0.714	0.807
	QA-LSTM (max-pooling)	0.733	0.819
	Conv-pooling LSTM	0.742	0.819
	Conv-based LSTM	0.737	0.827
	HD-LSTM TRAIN	0.7520	0.8146
	HD-LSTM TRAIN-ALL	0.7499	0.8153
I:Yang et al. (2016)	aNMM, TRAIN-ALL	0.7495	0.8109
I: Garg et al.	TANDA (BERT-b)	0.912	0.951
	TANDA (BERT-L)	0.912	0.967
	TANDA (RoBERTa-B)	0.914	0.952
	TANDA (RoBERTa-L)	**0.943**	**0.974**
H:He and Lin (2016)		0.7588	0.8219
H:Tan et al. (2016)	Attentive LSTM	0.753	0.830
H:Wang et al. (2016a)	IARNN-word	0.7098	0.7757
	IARNN-Occam(word)	0.7162	0.7916
	IARNN-context	0.7232	0.8069
	IARNN-Occam(context)	0.7272	0.8191

	IABRNN-GATE	0.7369	0.8208
	GRU	0.6487	0.6991
	OARNN	0.6887	0.7491
H:Bian et al. (2017)	listwise	0.810	0.889
	with k-max	0.817	0.895
	with k-threshold	0.821	0.899
H:Wang et al. (2017)	BiMPM	0.802	0.875
H:Tay et al. (2018)	MCAN(SM)	0.827	0.880
	MCAN(NN)	0.827	0.890
	MCAN(FM)	0.838	0.904
H: Yoon et al. (2019)	Comp-Clip	0.835	0.877
	Comp-Clip + LM	0.850	0.898
	Comp-Clip + LM + LC	0.868	0.928
	Comp-Clip + LM + LC +TL	0.875	0.940

Performance of different models from Section 6 and their baselines on Yahoo! dataset is reported in Table 9. One representation-based model (Tay et al., 2017), one interaction-based models (Wan et al., 2016b) , and one hybrid model (Wan et al., 2016a) are evaluated on the Yahoo! dataset. As can be seen, Bi-Match-SRNN, proposed by Wan et al. (2016b) outperforms other models in P@1 and MRR metrics. This is the only dataset in which an interaction-based model, namely Bi-Match-SRNN, performs better than the hybrid model.

Table 10 reports the results of different models on the Insurance-QA dataset. Among three categories of deep models (representation-based, interaction-based, and hybrid models) hybrid models achieves the best accuracy on the Insurance-QA dataset too. The proposed model by Wang and Jiang (2017) with SUBMULT+NN comparison function on TEST1, and with Mult comparison function on TEST2 achieves the best accuracy.

8 | DISCUSSION

Using deep neural networks in QA eliminated the manual feature engineering. In representation-based models, contextual representation of each sentence is modeled separately and then they are compared.

Yu et al. (2014) and Wang and Nyberg (2015) used distributional semantic model for generating a contextualized representation for text. Yu et al. (2014) applied CNN for capturing local dependencies among phrases of a sentence. Average pooling over CNN helps to create a representation of meaning the whole sentence. CNNs are capable of capturing complex semantics of a sentence in the given window. Wang and Nyberg (2015) utilized BiLSTM for creating a representation aware of the longer dependencies.

Matching proper nouns and cardinal numbers in question and answer sentences is an important issue. That means if the proper nouns in two sentences do not match, it will be enough reason for rejecting the answer. But, using pre-trained word embeddings not only does not distinguish these names but also considers a close representation for them. For mitigating this problem, Yu et al. (2014) used count of co-occurring words as a feature, and Wang and Nyberg (2015) used GBDT for exact matching of question and answer sentences.

Yin et al. (2016) and Severyn and Moschitti (2015) used CNNs in their architecture. Severyn and Moschitti (2015) used CNN for embedding the input text and proposed an architecture capable of adding other additional features. Yin et al. (2016) used wide convolution. Tay et al. (2017) used holographic composition for better modeling of question and answer relation. They enriched the aggregation of two embeddings by using a circular correlation. Tan et al. (2016) used

TABLE 9 Results of Different Models on Yahoo! Dataset

Model	Setting	P@1	MRR
Random Guess		0.2	0.4570
Okapi BM-25		0.2250	0.4927
CNN		0.4125	0.6323
CNTN		0.4654	0.6687
LSTM		0.4875	0.6829
NTN-LSTM		0.5448	0.7309
and Walker et al. (1995)	BM25	0.579	0.726
Socher et al. (2011)	RAE	0.398	0.652
Hu et al. (2014)	ARC-1	0.581	0.756
	ARC-2	0.766	0.869
	Deep Match	0.452	0.679
Qiu and Huang (2015)	CNTN	0.626	0.781
Yin and Schütze (2015)	MultiGranCNN	0.725	0.840
Palangi et al. (2016)	LSTM-RNN	0.690	0.822
Pang et al. (2016)	MatchPyramid-Tensor	0.764	0.867
R:Tay et al. (2017)	HD-LSTM	0.5569	0.7347
I:Wan et al. (2016b)	Match-SRNN	0.785	**0.879**
	Bi-Match-SRNN	**0.790**	0.882
H:Wan et al. (2016a)	MV-LSTM-Cosine	0.739	0.852
	MV-LSTM-Bilinear	0.751	0.860
	MV-LSTM-Tensor	0.766	0.869

different combinations of CNN and BiLSTM in different models. They used CNN on the top of BiLSTM for capturing richer information from the text and used BiLSTM on the top of CNN for capturing the long-range dependencies from local n-grams extracted by CNN. Using a combination of CNN and BiLSTM helps to mitigate the weakness of CNNs in capturing similarity between long dependencies in given texts.

Representation-based models build an embedding for each sentence separately across a distinct component. Although they are simple and usually use shared parameters across two separate components, they are not able to capture the matching between each pair of tokens from question and answer sentences. Interaction-based models solve this problem by direct interaction between each term of given sentences. Yang et al. (2016) generated a matching matrix by comparing each token of the question with each token of the answer and used attention for specifying the importance level of each question term. Wan et al. (2016b) proposed a special recursive model for modeling the interaction between two sequences by using GRU.

TABLE 10 Results of Different Models on Insurance-QA Dataset

Model	Setting	TEST1	TEST2
Bag-of-word		0.321	0.322
Metzler-Bendersky IR model		0.551	0.508
Feng et al. (2015)	CNN	0.628	0.592
	CNN with GESD	0.653	0.610
R:Tan et al. (2016)	QA-LSTM (head/tail)	0.536	0.510
	QA-LSTM (avg pooling,k=50)	0.557	0.524
	QA-LSTM (max pooling,k=1)	0.631	0.580
	QA-LSTM (max pooling,k=50)	0.666	0.637
	Conv-pooling LSTM (c=4000,k=1)	0.646	0.622
	Conv-pooling LSTM (c=200,k=1)	0.674	0.635
	Conv-pooling LSTM (c=400,k=50)	0.675	0.644
	Conv-based LSTM ($\|h\|$=200,k=50)	0.661	0.630
	Conv-based LSTM ($\|h\|$=400,k=50)	0.676	0.644
	QA-CNN (max-pooling, k = 3)	0.622	0.579
H: Tan et al. (2016)	Attentive CNN (max-pooling, k = 3)	0.633	0.602
	Attentive LSTM (avg-pooling k=1)	0.681	0.622
	Attentive LSTM (avg-pooling k=50)	0.678	0.632
	Attentive LSTM (max-pooling k=50)	0.690	0.648
H:Wang et al. (2016a)	IARNN-word	0.671	0.616
	IARNN-Occam (word)	0.696	0.637
	IARNN-context	0.667	0.631
	IARNN-Occam (context)	0.689	0.651
	IABRNN-GATE	0.701	0.628
	GRU	0.532	0.581
	OARNN	0.661	0.602
H: Wang and Jiang (2017)	NN	0.749	0.724
	NTN	0.750	0.725
	EucCos	0.702	0.679
	Sub	0.713	0.682
	Mult	0.752	**0.734**
	SUBMULT+NN	**0.756**	0.723

Using pre-trained language models like Peters et al. (2017), Radford et al. (2018), Devlin et al. (2019) has been recently attracted the researchers and made significance advances in many downstream NLP tasks. Pre-trained language models have also been applied to the QA task including the current state-of-the-art models. TANDA is one of the text-based QA models which use Bert and Roberta for modeling the contextual relation between question and answer sentence. As transformers are used in these pre-trained models for modeling the contextual representation of each given token by using the multi-head attention mechanism, we classify these models as interaction-based models.

Wan et al. (2016a) and He and Lin (2016) used BiLSTM for creating representation of each sentence and then built matching matrix on the top the new representation for each token. He and Lin (2016) generated the interaction matrix by concatenating the different similarity measures. They utilized CNN for finding strong interactions among a pair of words. In their model, the interaction matrix is built by using three different similarity metrics for comparing each pair of tokens, and similarity focus are used for assigning weight to interactions based on their level of importance. Wan et al. (2016a) captured rich contextualized local information of each token by using a BiLSTM and used three distinct similarity metrics for building an interaction tensor.

Hybrid models combine both interaction and representation models. They consist of a representation component that combines a sequence of words into a fixed d-dimensional representation and an interaction component. The attention mechanism is used in most of the hybrid models for generating a richer representation for answer or question sentences by attending to the other sentence.

We consider the attention mechanism as a type of interaction component because the attention weight for each token is calculated based on interacting with the two sentences. Tan et al. (2016) used attention mechanism for generating a question aware representation for answer sentence in their proposed Attentive-LSTM model. They used output of LSTM for question sentence, for weighting each of the answer sentence tokens. The last hidden state of RNNs carries more information about the sentence and this biases the attention toward the later hidden states. Wang et al. (2016a) utilized the attention mechanism in the word, the context, and the gate level of an RNN. Yin et al. (2016) proposed the first model which incorporated the attention mechanism in CNN.

Models proposed by Bian et al. (2017), Wang et al. (2017), Wang and Jiang (2017), Yoon et al. (2019), and Yang et al. (2019) follow a compare-aggregate architecture. Usually, there is a fewer number of relevant words in question and answer sentences, and summing the small attention weight of these irrelevant tokens affects the impact of relevant tokens. For mitigating this problem, Bian et al. (2017) introduced a new attention mechanism and adopted list-wise ranking instead of point-wise which better fits the nature of ranking.

Wang et al. (2017) matches the question and the answer sentences in two directions and from multiple perspectives. Wang and Jiang (2017) used different comparison functions for matching the answer representation with an attentional representation of answer from question perspective.

Yoon et al. (2019) leveraged the performance of CompClip by using transfer learning. They also used the ELMo language model for capturing more meaningful contextual information from question and answer sentences. Tay et al. (2018) generates a new embedding for each word and used the concatenation of the new embedding and the main word embedding. The new word embedding is built by casting the co-attention multiple times. On the other words, it casts attention instead of using it as a pooling strategy. RE2 (Yang et al., 2019) extracts more informative features from two sentences by using a simple attention mechanism for aligning two sentences and using augmented residual connections. RE2 has a very simple and fast architecture by using the minimum number of parameters and does not use RNNs due to its slow speed.

We discussed the main features of each model and found that currently, the state-of-the-art techniques include models that have used pre-trained language models like BERT. At the same time, representation models had the simplest architecture, and interaction models could extract richer information from the semantic relation of two sentences.

Hybrid models, which are a combination of these two types of models, usually use the attention mechanism. The attention mechanism helps to create a better representation based on another sentence. Most of these models followed the compare-aggregate architecture. The attention mechanism has had a significant impact on QA. In the architecture of BERT also the multi-head attention is used. Using this mechanism and pre-training it with a large size dataset provides a powerful model for extracting rich semantic representation from the text.

9 | CONCLUSION

In this paper, we provided a comprehensive review of the state-of-the-art methods on text-based QA systems. We first introduced the general architecture of QA systems, and then proposed a categorization for existing publications in the field. In the first step, publications are divided into two classes: information retrieval-based techniques, and deep learning-based techniques. We reviewed the main methods from both categories and highlighted deep learning-based approaches in more detail by following the well-know categorization for neural text matching, namely representation-based, interaction-based, and hybrid models. The existing publications with the deep learning perspective are categorized in these classes. We also reviewed available datasets that are widely used for training, validating, and testing text-based QA methods. The available results from different techniques on these datasets are presented in the paper to have a naive comparison of the techniques.

REFERENCES

Berger, A. and Lafferty, J. (1999) Information retrieval as statistical translation. In *Proceedings of the 22Nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '99, 222–229. New York, NY, USA: ACM.

Bian, W., Li, S., Yang, Z., Chen, G. and Lin, Z. (2017) A compare-aggregate model with dynamic-clip attention for answer selection. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, CIKM '17, 1987–1990. New York, NY, USA: ACM.

Bollacker, K., Evans, C., Paritosh, P., Sturge, T. and Taylor, J. (2008) Freebase: A collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD '08, 1247–1250. New York, NY, USA: ACM.

Bromley, J., Guyon, I., LeCun, Y., Säckinger, E. and Shah, R. (1993) Signature verification using a "siamese" time delay neural network. In *Proceedings of the 6th International Conference on Neural Information Processing Systems*, NIPS'93, 737–744. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Chang, M.-W., Goldwasser, D., Roth, D. and Srikumar, V. (2010) Discriminative learning over constrained latent representations. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, 429–437. Stroudsburg, PA, USA: Association for Computational Linguistics.

Cui, H., Sun, R., Li, K., Kan, M.-Y. and Chua, T.-S. (2005) Question answering passage retrieval using dependency relations. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '05, 400–407. New York, NY, USA: ACM.

Devlin, J., Chang, M.-W., Lee, K. and Toutanova, K. (2019) Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*.

Diefenbach, D., Lopez, V., Singh, K. and Maret, P. (2018) Core techniques of question answering systems over knowledge bases: a survey. *Knowledge and Information Systems*, **55**, 529–569.

Dimitrakis, E., Sgontzos, K. and Tzitzikas, Y. (2019) A survey on question answering systems over linked data and documents. *Journal of Intelligent Information Systems*, 1–27.

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

Garg, S., Vu, T. and Moschitti, A. () Tanda: Transfer and adapt pre-trained transformer models for answer sentence selection. *Thirty-Fourth AAAI Conference on Artificial Intelligence*.

Graves, A., rahman Mohamed, A. and Hinton, G. E. (2013) Speech recognition with deep recurrent neural networks. *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, 6645–6649.

Guo, J., Fan, Y., Ai, Q. and Croft, W. B. (2016) A deep relevance matching model for ad-hoc retrieval. In *CIKM*.

He, H. and Lin, J. (2016) Pairwise word interaction modeling with deep neural networks for semantic similarity measurement. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 937–948. San Diego, California: Association for Computational Linguistics.

He, X. and Golub, D. (2016) Character-level question answering with attention. In *EMNLP*.

Heilman, M. and Smith, N. A. (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*, HLT '10, 1011–1019. Stroudsburg, PA, USA: Association for Computational Linguistics.

Hochreiter, S. and Schmidhuber, J. (1997) Long short-term memory.

Hu, B., Lu, Z., Li, H. and Chen, Q. (2014) Convolutional neural network architectures for matching natural language sentences. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, 2042–2050. Cambridge, MA, USA: MIT Press.

Jurafsky, D. and Martin, J. H. (2009) *Speech and Language Processing (2Nd Edition)*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc.

Kodra, L. (2017) A review on neural network question answering systems. *International Journal of Artificial Intelligence & Applications*, **8**, 59–74.

Lai, T. M., Bui, T. and Li, S. (2018) A review on deep learning techniques applied to answer selection. In *Proceedings of the 27th International Conference on Computational Linguistics*, 2132–2144. Santa Fe, New Mexico, USA: Association for Computational Linguistics. URL: `https://www.aclweb.org/anthology/C18-1181`.

Le, Q. and Mikolov, T. (2014) Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning* (eds. E. P. Xing and T. Jebara), no. 2 in Proceedings of Machine Learning Research, 1188–1196. Bejing, China: PMLR.

Miao, Y., Yu, L. and Blunsom, P. (2016) Neural variational inference for text processing. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, 1727–1736. JMLR.org.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. and Dean, J. (2013) Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, 3111–3119. USA: Curran Associates Inc.

Momtazi, S. and Klakow, D. (2009) A Word Clustering Approach for Language Model-based Sentence Retrieval in Question Answering Systems. In *Proceedings of the Annual International ACM Conference on Information and Knowledge Management (CIKM)*, 1911–1914. ACM.

— (2011) Trained Trigger Language Model for Sentence Retrieval in QA: Bridging the Vocabulary Gap. In *Proceedings of the Annual International ACM Conference on Information and Knowledge Management (CIKM)*.

— (2015) Bridging the vocabulary gap between questions and answer sentences. *Inf. Process. Manage.*, **51**, 595–615.

Murdock, V. and Croft, W. B. (2004) Simple translation models for sentence retrieval in factoid question answering. In *SIGIR 2004*.

Palangi, H., Deng, L., Shen, Y., Gao, J., He, X., Chen, J., Song, X. and Ward, R. (2016) Deep sentence embedding using long short-term memory networks: Analysis and application to information retrieval. *IEEE/ACM Trans. Audio, Speech and Lang. Proc.*, **24**, 694–707.

Pang, L., Lan, Y., Guo, J., Xu, J., Wan, S. and Cheng, X. (2016) Text matching as image recognition. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, 2793–2799. AAAI Press.

Pennington, J., Socher, R. and Manning, C. (2014) Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 1532–1543. Doha, Qatar: Association for Computational Linguistics.

Peters, M., Ammar, W., Bhagavatula, C. and Power, R. (2017) Semi-supervised sequence tagging with bidirectional language models. *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Peters, M., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K. and Zettlemoyer, L. (2018) Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, 2227–2237. New Orleans, Louisiana: Association for Computational Linguistics.

Qiu, X. and Huang, X. (2015) Convolutional neural tensor network architecture for community-based question answering. In *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, 1305–1311. AAAI Press.

Radford, A., Narasimhan, K., Salimans, T. and Sutskever, I. (2018) Improving language understanding by generative pre-training. *URL https://s3-us-west-2. amazonaws. com/openai-assets/researchcovers/languageunsupervised/language understanding paper. pdf*.

Rao, J., He, H. and Lin, J. (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*, CIKM '16, 1913–1916. New York, NY, USA: ACM.

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

Soares, M. A. C. and Parreiras, F. S. (2018) A literature review on question answering techniques, paradigms and systems. *Journal of King Saud University-Computer and Information Sciences*.

Socher, R., Huang, E. H., Pennington, J., Ng, A. Y. and Manning, C. D. (2011) Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Proceedings of the 24th International Conference on Neural Information Processing Systems*, NIPS'11, 801–809. USA: Curran Associates Inc.

Tan, M., dos Santos, C., Xiang, B. and Zhou, B. (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. Berlin, Germany: Association for Computational Linguistics.

Tapaswi, M., Zhu, Y., Stiefelhagen, R., Torralba, A., Urtasun, R. and Fidler, S. (2016) Movieqa: Understanding stories in movies through question-answering. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 4631–4640.

Tay, Y., Phan, M. C., Tuan, L. A. and Hui, S. C. (2017) Learning to rank question answer pairs with holographic dual lstm architecture. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '17, 695–704. New York, NY, USA: ACM.

Tay, Y., Tuan, L. A. and Hui, S. C. (2018) Multi-cast attention networks. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '18, 2299–2308. New York, NY, USA: Association for Computing Machinery.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u. and Polosukhin, I. (2017) Attention is all you need. In *Advances in Neural Information Processing Systems 30* (eds. I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan and R. Garnett), 5998–6008. Curran Associates, Inc.

and Walker, S., Jones, S., Hancock-Beaulieu, M. M. and Gatford, M. (1995) Okapi at trec-3. In *Overview of the Third Text REtrieval Conference (TREC)*, 109–126. Gaithersburg, MD: NIST.

Wan, S., Lan, Y., Guo, J., Xu, J., Pang, L. and Cheng, X. (2016a) A deep architecture for semantic matching with multiple positional sentence representations. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, 2835–2841. AAAI Press.

Wan, S., Lan, Y., Xu, J., Guo, J., Pang, L. and Cheng, X. (2016b) Match-srnn: Modeling the recursive matching structure with spatial rnn. In *IJCAI*.

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O. and Bowman, S. (2018) GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, 353–355. Brussels, Belgium: Association for Computational Linguistics.

Wang, B., Liu, K. and Zhao, J. (2016a) 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. Berlin, Germany: Association for Computational Linguistics.

Wang, D. and Nyberg, E. (2015) A long short-term memory model for answer sentence selection in question answering. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, 707–712. Beijing, China: Association for Computational Linguistics.

Wang, M. and Manning, C. D. (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 '10, 1164–1172. Stroudsburg, PA, USA: Association for Computational Linguistics.

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

Wang, S. and Jiang, J. (2017) A compare-aggregate model for matching text sequences. International Conference on Learning Representations (ICLR).

Wang, Z., Hamza, W. and Florian, R. (2017) Bilateral multi-perspective matching for natural language sentences. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, 4144–4150.

Wang, Z., Mi, H. and Ittycheriah, A. (2016b) Sentence similarity learning by lexical decomposition and composition. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, 1340–1349. Osaka, Japan: The COLING 2016 Organizing Committee.

Wu, P., Zhang, X. and Feng, Z. (2019) A survey of question answering over knowledge base. In *Knowledge Graph and Semantic Computing: Knowledge Computing and Language Understanding* (eds. X. Zhu, B. Qin, X. Zhu, M. Liu and L. Qian), 86–97. Singapore: Springer Singapore.

Yadav, V., Sharp, R. and Surdeanu, M. (2018) Sanity check: A strong alignment and information retrieval baseline for question answering. In *SIGIR*.

Yang, L., Ai, Q., Guo, J. and Croft, W. B. (2016) anmm: Ranking short answer texts with attention-based neural matching model. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, CIKM '16, 287–296. New York, NY, USA: ACM.

Yang, R., Zhang, J., Gao, X., Ji, F. and Chen, H. (2019) Simple and effective text matching with richer alignment features. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 4699–4709. Association for Computational Linguistics.

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

Yao, X., Van Durme, B., Callison-Burch, C. and Clark, P. (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. Atlanta, Georgia: Association for Computational Linguistics.

Yih, W.-t., Chang, M.-W., Meek, C. and Pastusiak, A. (2013) Question answering using enhanced lexical semantic models. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 1744–1753. Sofia, Bulgaria: Association for Computational Linguistics.

Yin, W. and Schütze, H. (2015) Multigrancnn: An architecture for general matching of text chunks on multiple levels of granularity. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 63–73. Beijing, China: Association for Computational Linguistics.

Yin, W., Schütze, H., Xiang, B. and Zhou, B. (2016) Abcnn: Attention-based convolutional neural network for modeling sentence pairs. *Transactions of the Association for Computational Linguistics*, **4**, 259–272.

Yoon, S., Dernoncourt, F., Kim, D. S., Bui, T. and Jung, K. (2019) A compare-aggregate model with latent clustering for answer selection. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, 2093–2096.

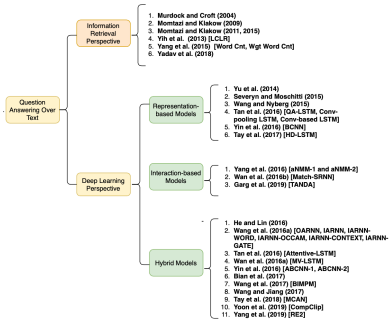
Yu, L., Hermann, K. M., Blunsom, P. and Pulman, S. G. (2014) Deep learning for answer sentence selection. In *Deep Learning and Representation Learning Workshop: NIPS 2014*, vol. abs/1412.1632.

Yu, M., Yin, W., Hasan, K. S., dos Santos, C., Xiang, B. and Zhou, B. (2017) Improved neural relation detection for knowledge base question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 571–581. Vancouver, Canada: Association for Computational Linguistics.

S. MOMTAZI is currently an assistant professor at the Amirkabir University of Technology, Iran. She completed her BSc and MSc education at the Sharif University of Technology, Iran. She received a Ph.D. degree in Artificial Intelligence from Saarland University, Germany. As part of her Ph.D., she was a visiting researcher at the Center of Language and Speech Processing at Johns Hopkins University, US. After finishing the Ph.D., she worked at the Hasso-Plattner Institute (HPI) at Potsdam University, Germany and the German Institute for International Educational Research (DIPF), Germany as a post-doctoral researcher. Natural language processing with a focus on question answering systems is her main research focus. She has worked in this area of research for more than 14 years.

Z. ABBASIANTAEB is a graduate student of Artificial Intelligence at the Amirkabir University of Technology, Iran. She received her B.Sc. degree in Software Engineering from the Amirkabir University of Technology, Iran. Natural language processing and information retrieval are her main research interests. Currently, she is working as a research assistant at NLP Lab under the supervision of Dr. Saeedeh Momtazi.

GRAPHICAL ABSTRACT



Text-based Question Answering (QA) has been widely studied in Information Retrieval (IR) communities. By the advent of Deep Learning (DL) techniques, various DL-based methods have been used for this task which have not been studied and compared well. In this paper, we provide a comprehensive overview of different models proposed for QA, including both traditional IR perspective, and more recent DL perspective.