From Question to Text: Question-Oriented Feature Attention for Answer Selection

HEYAN HUANG and XIAOCHI WEI, Beijing Institute of Technology, China
LIQIANG NIE, Shandong University, China
XIANLING MAO, Beijing Institute of Technology, China
XIN-SHUN XU, Shandong University, China

Understanding unstructured texts is an essential skill for human beings as it enables knowledge acquisition. Although understanding unstructured texts is easy for we human beings with good education, it is a great challenge for machines. Recently, with the rapid development of artificial intelligence techniques, researchers put efforts to teach machines to understand texts and justify the educated machines by letting them solve the questions upon the given unstructured texts, inspired by the reading comprehension test as we humans do. However, feature effectiveness with respect to different questions significantly hinders the performance of answer selection, because different questions may focus on various aspects of the given text and answer candidates. To solve this problem, we propose a question-oriented feature attention (QFA) mechanism, which learns to weight different engineering features according to the given question, so that important features with respect to the specific question is emphasized accordingly. Experiments on MCTest dataset have well-validated the effectiveness of the proposed method. Additionally, the proposed QFA is applicable to various IR tasks, such as question answering and answer selection. We have verified the applicability on a crawled community-based question-answering dataset.

CCS Concepts: • **Information systems → Question answering**;

Additional Key Words and Phrases: Question answering, answer selection, attention method

ACM Reference format:
Heyan Huang, Xiaochi Wei, Liqiang Nie, Xianling Mao, and Xin-Shun Xu. 2018. From Question to Text: Question-Oriented Feature Attention for Answer Selection. *ACM Trans. Inf. Syst.* 37, 1, Article 6 (October 2018), 33 pages.
https://doi.org/10.1145/3233771

This work is supported by the National High Technology Research and Development Program of China (863 Program, No. 2015AA015404), the National Natural Science Foundation of China (61751201, 61702310, 61702300, and 61702302), the National Basic Research Program of China (973 Program, No. 2015CB352502), the Project of Thousand Youth Talents 2016, and the Tencent AI Lab Rhino-Bird Joint Research Program (No. JR201805).
Authors' addresses: H. Huang and X. Wei (corresponding author), Beijing Engineering Research Center of High Volume Language Information Processing and Cloud Computing Applications, School of Computer, Beijing Institute of Technology, No.5 Zhongguancun South Street, Haidian 100081, Beijing, China; emails: hhy63@bit.edu.cn, wxchi@bit.edu.cn; L. Nie, School of Computer Science, Shandong University, 72 Binhai Road, Jimo Qingdao, 266237, Shandong Province, China; email: nieliqiang@gmail.com; X. Mao, Beijing Engineering Research Center of High Volume Language Information Processing and Cloud Computing Applications, School of Computer, Beijing Institute of Technology, No.5 Zhongguancun South Street, Haidian 100081, Beijing, China; email: maoxl@bit.edu.cn; X.-S. Xu, School of Computer Science, Shandong University, 72 Binhai Road, Jimo Qingdao, 266237, Shandong Province, China; email: xuxinshun@sdu.edu.cn.

https://doi.org/10.1145/3233771

1 INTRODUCTION

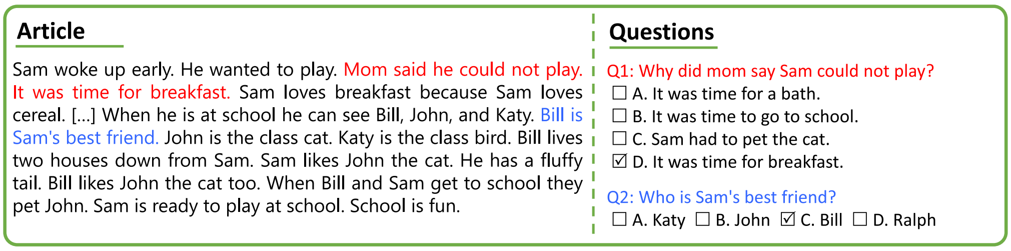
Languages are important tools for human beings to record their mind, life, and culture. Knowledge accumulated in these records, often in the form of unstructured texts, is spread worldwide. Reading comprehension skills enable readers to gain the deeper insights into and absorb these involved knowledge. Yet, even for the same record, the amount and quality of knowledge perceived by different readers varies remarkably. The more passage content they understand, the more knowledge they acquire. On the contrary, a smattering of knowledge may seriously confuse and even mislead readers. Despite the importance of reading comprehension, it is a laborious and time-consuming process. The average reading speed is 200 words per minute according to the statistic of dailymail,[1] which means that it requires about 3h to read through the book of *Hamlet* without stopping.

To evaluate and improve the ability of reading comprehension, in most language examinations, such as International English Language Testing System (IELTS), Test of English as a Foreign Language (TOEFL), and Graduate Record Examination (GRE), reading comprehension is a compulsory testing subject. Currently, most reading comprehension examinations are in the form of multiple-choice question answering. Figure 1 displays a typical example. In this kind of examination, a text is first presented to the examinees. They then read the text to comprehend its meaning. After that, several multiple-choice questions follow up, and only one answer candidate is correct for each question. The examinees are required to choose the correct answers based on their understanding of the text. Different from traditional question answering (QA) [25, 28, 64], questions in reading comprehension are hard to be correctly answered without knowing the context of the given texts. Common sense is less helpful here, because answer cues are hidden in the original text. Considering questions Q1 and Q2 illustrated in Figure 1 as examples, we can never know *who Sam is*, *what Sam is going to do*, or *who Bill is* without reading the short text. In light of this, people believe that this kind of QA can well test the ability of reading comprehension.

Recently, lots of work has been done on machine reading, which focuses on teaching machines to understand unstructured text, and let them answer related multiple-choice questions to judge their reading comprehension ability [34, 41, 46, 56]. Most of them regard the task of answer selection as a classification or ranking problem with feature engineering methods. Technically speaking, the question and each answer candidate can be combined together to reconstruct a statement sentence. For example, considering the questions in Figure 1, question Q1 combined with its answer candidate "A" can be restated as the statement "*Mom said Sam could not play because it was time for a bath,*" and question Q2 combined with its answer candidate "A" can be restated as "*Bill is Sam's best friend.*" In this way, the answer selection task is successfully converted into the problem of finding the correct statement according to the given text. Prior work extracted various engineering features from the text-statement pairs, such as lexical features [46], dependency features [56], and coreference features [42], and these engineering features are directly fed into regression or classification models to infer the correct answer.

However, these existing methods may be suboptimal, because the extracted engineering features can hardly be efficiently utilized according to the given question. The structures of natural language sentences are very complex and sophisticated. Even a simple sentence contains multi-aspect information, e.g., the person, the time, and the reason. On the contrary, questions are usually very simple and straightforward, and it is on only one or a few points of related sentences. An example is illustrated in Figure 2. In this example, there is a sentence and it contains rich information, such as people (Steve Jobs and Steve Wozniak), organization (Apple Inc.), and time (1976). Questions Q1 and Q2 are two different questions about this sentence, but Q1 only focuses on "when" and Q2 is on

[1]The statistic is available at http://tinyurl.com/ydfcs59d.



Article

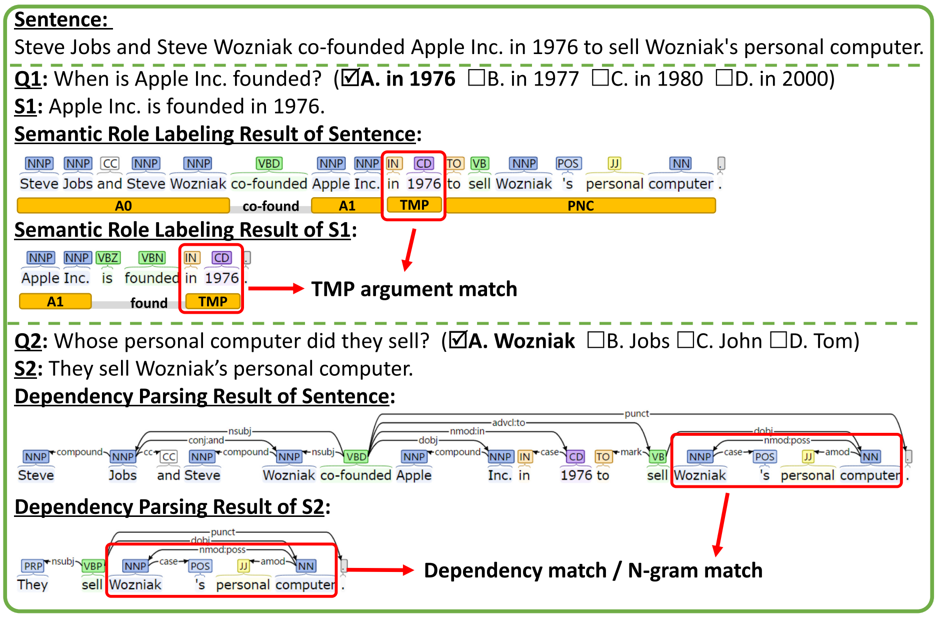
Sam woke up early. He wanted to play. Mom said he could not play. It was time for breakfast. Sam loves breakfast because Sam loves cereal. [...] When he is at school he can see Bill, John, and Katy. Bill is Sam's best friend. John is the class cat. Katy is the class bird. Bill lives two houses down from Sam. Sam likes John the cat. He has a fluffy tail. Bill likes John the cat too. When Bill and Sam get to school they pet John. Sam is ready to play at school. School is fun.

Questions

Q1: Why did mom say Sam could not play?
☐ A. It was time for a bath.
☐ B. It was time to go to school.
☐ C. Sam had to pet the cat.
☑ D. It was time for breakfast.

Q2: Who is Sam's best friend?
☐ A. Katy ☐ B. John ☑ C. Bill ☐ D. Ralph

Fig. 1. Illustration of reading comprehension examination. To save some space, we deliberately omit some irrelevant content of the text. The correct answers of Q1 and Q2 can be inferred from the red and blue parts, respectively.



Sentence:
Steve Jobs and Steve Wozniak co-founded Apple Inc. in 1976 to sell Wozniak's personal computer.

Q1: When is Apple Inc. founded? (☑**A. in 1976** ☐B. in 1977 ☐C. in 1980 ☐D. in 2000)
S1: Apple Inc. is founded in 1976.

Semantic Role Labeling Result of Sentence:

Semantic Role Labeling Result of S1:

Q2: Whose personal computer did they sell? (☑**A. Wozniak** ☐B. Jobs ☐C. John ☐D. Tom)
S2: They sell Wozniak's personal computer.

Dependency Parsing Result of Sentence:

Dependency Parsing Result of S2:

NNP NNP CC NNP NNP VBD NNP NNP IN CD TO VB NNP POS JJ NN
Steve Jobs and Steve Wozniak co-founded Apple Inc. in 1976 to sell Wozniak 's personal computer .
A0 co-found A1 TMP PNC
NNP NNP VBZ VBN IN CD
Apple Inc. is founded in 1976 .
A1 found TMP
TMP argument match
compound cc conj:and nsubj compound nsubj dobj compound advcl:to nmod:in case mark punct dobj nmod:poss case amod
nsubj dobj nmod:poss case amod punct
PRP VBP NNP POS JJ NN
They sell Wozniak 's personal computer .
Dependency match / N-gram match

Fig. 2. Illustration of the challenge that different questions may focus on different features. Question Q1 and Q2 ask something about the same given sentence. The illustrated statement S1 and S2 are generated according to the questions and their corresponding correct answers.

"whose." Researchers have dedicated lots of feature engineering efforts to capture different kinds of clues in the sentence, including but not limited to, part of speech matching [20, 30, 35], named entity matching [37, 39, 59], and semantic matching [14, 32, 72]. Nevertheless, the effectiveness of these features varies a lot when resolving different questions, and only a part of features are useful to a given question. What is more, some useless features may even bring in noises and hence impact the answer selection performance. For example, as shown in Figure 2, question Q1 focuses on the time when the Apple Inc. was founded. According to the semantic role labeling result[2] of the

[2]The semantic role labeling is implemented with the help of mate-tools NLP. The demo is available at http://homepages.inf.ed.ac.uk/mroth/demo.html.

original sentence and the statement[3] S1, we find that the "TMP"[4] argument matching feature exactly indicates the correctness of the statement. In contrast, other features, e.g., A1 argument matching, have less help. Different from Q1, Q2 focuses on the owner of the personal computer, so the efficient features for these two questions are quite different. By analyzing the result of dependency parsing,[5] it is clear that the dependency of "nmod:poss"[6] from "computer" to "Wozniak" indicates the correct answer, and the n-gram matching of the sequence "Wozniak's personal computer" also provides the evidence, while other features, such as the "TMP" argument matching used in Q1, may mislead the final decision. Whereas, in most existing work, these features are equally treated among different questions and the importance of these features are not distinguished accordingly. Therefore, how to efficiently utilize these engineering features according the given question is a vital problem to solve.

To solve this problem, we use the attention mechanism to automatically weight the extracted engineering features according to the given question, and we propose a novel **Q**uestion-oriented **F**eature **A**ttention (QFA) mechanism. In particular, it learns a weight distribution of engineering features by deeply analyzing the given question, so that the importance of different features with respect to different questions can be distinguished. To demonstrate the effectiveness of the QFA, we propose a QFA-based answer selection model, QFAReader. Specifically, It first pairs the question and each answer candidate together to establish a statement candidate. After that, each sentence in the text and the statement candidate are combined together to generate a sentence-statement pair and for each of them, a rich set of correctness-oriented engineering features are extracted. These extracted engineering features are then weighted with the QFA to distinguish their importance and then merged together with a cross-sentence max pooling for the final decision. It is noted that the proposed QFA and QFAReader is applicable to other IR tasks, such as answering selection and search result ranking. By considering the content of the given question/query, the importance of engineering features extracted from candidates can be effectively distinguished for follow up answer selection and result ranking.

The main contributions of this article are threefold:

- As far as we know, this is the first work on feature-level attention mechanism. Different from previous work on attention mechanisms that determine which sentence/word is more important [15, 24, 29], our QFA automatically prioritizes different features according to the given question/query, so that feature importance in different questions can be well distinguished.
- We successfully apply the proposed QFA mechanism into the task of reading comprehension, and propose a QFA-based reading comprehension model, QFAReader, accordingly. It first utilizes QFA to weight different engineering features according to the given question, then uses the weighted feature to judge the answer correctness. In this way, features extracted for different questions can be more efficiently utilized in the answer selection model.
- Via extensive experiments on the MCTest dataset, we have well validated our QFA mechanism and the QFAReader. In addition, to demonstrate their applicability, we have applied the QFAReader to the task of answer selection, and the experiment is conducted on a real word community-based question answering (cQA) dataset. As a byproduct, we have released our code and data to facilitate other researchers.[7]

[3]The statement is constructed according to the question and the answer candidate A.
[4]"TMP" denotes the argument of time in semantic role labeling.
[5]The dependency parsing is implemented with Stanford CoreNLP. The online demo is available at http://corenlp.run/.
[6]"nmod:poss" denotes the relation of nominal modifier in dependency parsing.
[7]Our code and data are available at https://datapublication.wixsite.com/qfareader.

The rest of this article is organized as follows. Section 2 describes the related work about QA, reading comprehension, and attention mechanisms. Section 3 defines the reading comprehension problem and details our extracted features and the proposed model. In Section 4, we present the experimental results and analysis, and in Section 5, we applied the QFAReader to the task of cQA answer selection, followed by conclusion and future work in Section 6.

2 RELATED WORK

Our work is related to QA systems, reading comprehension, as well as attention mechanisms.

2.1 QA Systems

QA systems alleviate information overload by providing users simple and accurate answers. A great many QA systems were developed for this purpose, by utilizing external sources to obtain the correct answer. These systems can be roughly divided into three categories, according to the type of external sources they use, i.e., document-based QA, knowledge-based QA, and Community-based QA.

Document-based QA mainly focuses on open-domain questions, and it retrieves relevant sentences or phrases from unstructed document corpus to answer the question. The TREC QA track [54], initially from 1999, is a milestone. Most QA systems accomplish this task via pipeline frameworks [25, 28, 64]. The questions are first translated into keyword queries to retrieve relevant passages from the corpus. Then linguistic and statistical methods are utilized to extract answer candidates from relevant passages, and these candidates are finally ranked based on aggregated evidences and features to obtain the correct answer.

Different from document-based QA, the knowledge-based one retrieves answers from structured knowledge bases, e.g., Freebase, DBPedia, and Yago. Berant et al. [3] constructed the early system. It represents the question with logical form and then retrieves the answer entity with the generated logical form from the knowledge base. Following this, researchers explored various methods to represent questions. For example, Bao et al. [2] represented the question with a incomplete-triple form, i.e., (entity, relation, ?), where "?"denotes the missing entity, by a machine translation model, then directly matched the answer entity in the knowledge base. Bordes et al. [6] introduced the idea of representation learning with the knowledge-based QA. Following this idea, Yang et al. [65], Bordes et al. [5], and Wei et al. [60] jointly represented question and knowledge base with vectors, and inferred the correct answer by comparing the distance in the latent space. Dong et al. [18] employed neural network structure to involve relation path, subgraph, and answer category into the answer representation. Yao et al. [68] leverage external free text to reinforce the learning of question embedding and knowledge embedding.

Community-based QA mainly focuses on selecting the most similar user-generated QA pairs to the given question. Most researchers in this field explore the effectiveness of different features. Surdeanu et al. [49] developed a set of NLP features, including answer content, similarity, and translation probability. Cao et al. [8] utilized question category information to enhance the retrieval performance. Considering the property of community-based QA, user activities, user profiles, and social network features were exploited by Bian et al. [4], Zhou et al. [71], and Molino et al. [33], respectively. Embedding features generated by neural networks have been explored by Shen et al. [44] and Yin et al. [69] in recent years. Shah et al. [43] and Dalip et al. [16] summarized these commonly used features and analyzed their effectiveness.

The problem of reading comprehension can be regarded as an extension of document-based QA. The traditional document-based QA answers questions via retrieving answers from large corpus, while the questions of reading comprehension only focus on the given passages. Therefore, the traditional QA method can hardly be directly applied to this problem.

2.2 Reading Comprehension

Studies on reading comprehension are still at the infant stage. According to the type of datasets they utilized, existing methods can be divided into two categories, i.e., methods for multiple-choice questions and for cloze-test questions.

The most typical dataset for multiple-choice questions is MCTest [41]. This dataset is small, and most methods on this dataset used fixed and engineered features to represent the question-answer pairs. They then employed pair-wise ranking models to select the correct answer. Smith et al. [46] utilized simple lexical matching, and obtained good performance. Wang et al. [56] developed a rich set of features extracted from various natural language processing results, e.g., frame semantic parsing, dependency parsing, and coreference resolution. Karthik et al. [34] explored the effectiveness of discourse relations in the reading comprehension task. Sachan et al. [42] regarded the reading comprehension as the textual entailment, and extracted a great many features from the constructed answer-entailing structure. Word embeddings were also explored by Trischler et al. [51], and they demonstrated that word embeddings contain rich useful information to comprehend the given text. Additionally, Wang et al. [55] noticed the data deficiency problem in the MCTest dataset, and incorporated external sources to enhance the performance.

Different from multiple-choice questions, large-scale cloze-test questions, e.g., CNN and Dailly Mail [21], CBT [22], and People Daily [15], can be automatically generated easily by randomly removing entities from texts. Therefore, most existing work on this kind of questions relies on deep neural networks as the large-scale of the dataset. Hermann et al. [21] proposed two deep models, i.e., Attentive Reader and Impatient Reader, inspired by the Memory Networks [61]. Attentive Reader represents the text and the question with bidirectional LSTM, and applies a word token level attention mechanism to emphasize the importance of different words. Impatient Reader goes one step further by enabling the model to re-read the text as each question token is observed. Chen et al. [10] simplified the Attentive Reader by incorporating the question embedding into the passage embedding, and they obtained a significant improvement. Kadlec et al. [24] employed pointer networks [53] to directly pick the answer from the text as opposed to computing the answer using a blended representation of words. Dhingra et al. [17] proposed a Gated Attention Reader, which utilizes the attention mechanism to achieve the multiplicative interaction between the query and the document. Motivated by the generative adversarial architecture, Yang et al. [67] use auto-generated questions to reinforce the training of question answering model.

Most of these discussed reading comprehension methods can hardly dynamically weight features according to the given question. When the answer selection model is well trained, the weights of different features are fixed. Question content in these models are usually used as engineering features to evaluate answer correctness. So when it comes to answering new questions, different features can hardly be distinguished accordingly. Distinct from these existing methods, our proposed QFAReader deeply analyzes the content of questions and extract a feature attention to distinguish feature, so that it is capable of emphasizing the useful features and understating useless ones appropriately according to the given question.

2.3 Attention Mechanisms

Attention mechanisms are initially found in the human visual systems [11, 40]. Researchers found that the salient region of human views dynamically comes to the forefront as we need, instead of compressing entire image into a statistic representation. Inspired by this, attention mechanisms are successfully applied to machine translation [1, 29], image caption [31, 63], image QA [45, 66], and reading comprehension [10, 21]. In particular, Bahdanau et al. [1] applied the attention mechanisms to the RNN decoder in machine translation. It emphasizes the corresponding source language word, when generating the target language word. Luong et al. [29] followed the work of Reference [1] and extended the traditional attention mechanism. To reduce the time complexity of attention mechanism, Luong et al. [29] proposed a local attention, which adds a sliding window and only considers the words in the window. Xu et al. [63] proposed an end-to-end image caption model with attention mechanism. It encodes the image into a set of feature vectors with a CNN, and each vector is a representation corresponding to a part of the image. The attention mechanism is then utilized to decide which part of the image weighs more when generating each word of the caption. Yang et al. [66] added the attention mechanism into image question answering. They encoded each part of the image into a feature vector with CNN, and encoded the question with an LSTM. The representations of image and questions are fed into a single-layer neural network to generate the attention distribution, which determines the useful part of the image for answering the question. Shih et al. [45] proposed another attention-based image question answering model. The image regions and the question are first represented into the same semantic space, then the dot product between the question and the image region representation is used as the attention, which selects the relevant part of the image. For reading comprehension, Hermann et al. [21] used the attention mechanism to select relevant words in the text, according to the given question. Chen et al. [10] compared the embedding of the question and each word to obtain the attention score. The weighted combination of all word embeddings is then utilized to infer the correct answer. Besides, attention mechanisms have been successfully applied in search engines to emphasize relevant search items [26, 27, 47].

Attention mechanisms have the ability of selecting important information from context. However, most existing attention mechanisms focus on selecting specific pieces, i.e., the relevant part of images, some important words, and relevant sentences. Different from the previous work, our proposed question-oriented feature attention is used to select useful features according to the given question. As far as we know, this is the first work on feature-level attention mechanisms.

3 PROPOSED METHOD

In this section, we first formalize the reading comprehension problem and then summarize the features we used. After that, we detail our proposed QFA mechanism. At last, we describe the QFAReader model.

3.1 Problem Definition

We first declare some notations. In particular, we use bold capital letters (e.g., $\mathbf{X}$) and bold lowercase letters (e.g., $\mathbf{x}$) to denote matrices and vectors, respectively. We employ non-bold letters (e.g., x) to represent scalars and curlicue letters (e.g., $\mathcal{X}$) as sets. If not clarified, then all vectors are in column forms.

Reading comprehension requires machine to answer a series of questions according to the given unstructed text. It can be treated as an extension of traditional QA systems. To successfully solve this problem, reading comprehension requires not only the question and answer candidates but also a text that the question is based on. This problem can be well formalized as follows. For each question q, let $\mathcal{A} = \{a_1, a_2, \ldots, a_M\}$ be the set of given answer candidates to the question and $\mathcal{T} = \{t_1, t_2, \ldots, t_N\}$ be the corresponding unstructured text, where a_i indicates the ith answer candidate for q and t_j indicates the jth sentence in the text $\mathcal{T}$. The reading comprehension task is then simplified as selecting the best answer $a_k \in \mathcal{A}$ with the highest evidence given q, $\mathcal{T}$, and $\mathcal{A}$, i.e., $\arg\max_k P(a_k|q, \mathcal{T}, \mathcal{A})$.

Table 1. Examples of Rewriting Rules for Generating the Question-Answer Candidate Statements

Question Pattern	Answer Type	Rewritten Expression
Who V<(is|was|are|were)> (.+)	Person	$Ans $1 $2
What V<(is|are|was|were)> the (\S+) (of|for) (.+)	$2	$2 $3 $4 $1 $Ans
When V<(is|was|are|were)> (.+)	Year	$2 $1 in $Ans

3.2 Statement Construction

In QFAReader, the question and its answer candidate are first reconstructed as a statement sentence. For example, if the question is "*Who is Sam's best friend?*" and the answer candidate is "*Katy*," then the statement "*Katy is Sam's best friend*" is constructed accordingly. The statement encodes information from both the question and the answer candidate. In this way, solving the question is switched to select the correct statement according to the given text.

To accomplish this goal, the rule-based method described in Reference [13] is utilized in this article.[8] Its pre-defined 198 manually constructed rewriting rules for different kinds of questions are designed according to their prefix (e.g., when, where, and what). The rules are represented as the form of regular expression, and some examples are listed in Table 1. In this table, \$1, \$2, \$3, and \$4 denote parameters in question patterns and \$Ans is the answer candidate for the question. According to the first rule in the table, the question "*Who is Sam's best friend?*" and its corresponding answer candidate "*Katy*" is able to construct the statement "*Katy is Sam's best friend.*" Similarly, the question "*What is the name of the dog?*" and its corresponding answer candidate "*Max*" can be rewritten as the statement "*Name of the dog is Max*" according to the second rule in this table. In the construction process, these rewriting rules are sorted by question prefix, and for each prefix, the particular rules are listed higher than general ones. When a new question is observed, the system checks the rule in the list in order until a match is founded.

With these constructed statements, the one sharing the most similar description to the text is more likely to be correct. For example, in the given text displayed in Figure 1, the question Q2 and its four answer candidates can be converted into four statements, and the QFAReader should select the statement "*Bill is Sam's best friend*" as the most likely one.

3.3 Extracted Features

In this section, we describe the features extracted for the reading comprehension task. In QFAReader, the answer candidates are judged by measuring the correctness of its corresponding statement, we hence extract a rich set of linguistic features, and they can be roughly divided into eight categories, namely, word-matching features, part of speech (POS) features, dependency features, constituency features, named entity (NE) features, semantic role (SR) features, semantic features, and coreference features.

To incorporate information from all sentences of the given text, these features are extracted from every single sentence, instead of the whole text. However, the coreference severely impacts the quality of features. There are a great many pronouns (e.g., he, she, and they), and the sentences are hard to understand without indicating the reference of the pronouns, if only the current sentence is considered. For example, when comparing the sentence "*She made a big cake and hung up some balloons*" and the statement "*Jessie got ready for the party, made cake and hung balloons,*" it is impossible to conclude that the statement is correct, as the original sentence does not tell us

[8] We utilized MCTest as the dataset in our experiments. The dataset provides the constructed statements for each question-answer candidate pair, and these statements are constructed with this method.

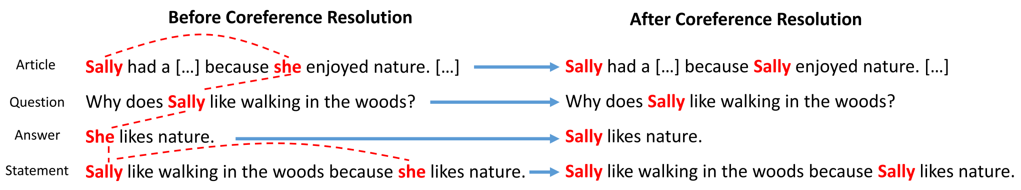

Before Coreference Resolution
After Coreference Resolution
Article
Sally had a […] because she enjoyed nature. […]
Sally had a […] because Sally enjoyed nature. […]
Question
Why does Sally like walking in the woods?
Why does Sally like walking in the woods?
Answer
She likes nature.
Sally likes nature.
Statement
Sally like walking in the woods because she likes nature.
Sally like walking in the woods because Sally likes nature.

Fig. 3. The coreference resolution process before feature extraction. Some irrelevant contents of the text are omitted. The red dot lines indicate the coreference relations between words.

whether "she" denotes "Jessie" or not. Therefore, we run a coreference resolution system for the text, question, answer, and statement before the feature extraction process. We first concatenate the given text, the questions, the corresponding answers, and the generated statements together. Then, with the help of Stanford CorefAnnotator,[9] the same coreference entities are found out and then replaced by a common display name. Figure 3 illustrates an example of the coreference resolution process. In this figure, the red words denote the same person named "Sally," so all of them are replaced into "Sally." After coreference resolution, sentences can be better processed with various NLP tools, hence the extracted features are more reliable to judge the statement.

We now detail our extracted features. It is worth noting that all these features are extracted from sentence-answer candidate pairs or sentence-statement candidate pairs.

3.3.1 Word-Matching Features. Word-matching features are widely used in traditional QA systems to evaluate the quality of the answers [16, 28, 49]. Richardson et al. [41] introduced them into the reading comprehension task. Word-matching features measure the correctness of answer candidates with the assumption that if there are more overlapping between the statement candidate and the question, the corresponding answer candidate is more likely to be correct. We hence utilize the number of matched words and the ratio of matched words in the statement candidate as two features. Sliding window has been demonstrated effective in measuring the correctness of the answer candidate in previous work [41]. It measures the correctness of answer candidates by matching a word sequence of several words instead of one word only, we hence add the number of matched bi-grams and tri-grams between the statement candidate and the sentence into the feature set. Considering the phenomenon of inflection in English (e.g., apple→apples, go→going, and do→done), the sentence and the statement candidate are lemmatized,[10] and the number of matched words between the lemmatized sentence and the lemmatized statement candidate is added into the feature set. Since the content of the answer is decisive in determining the correctness, we hence extracte the aforementioned five features, i.e., the ratio of word matching, the number of word matching, bi-gram matching, tri-gram matching, and lemmatized word matching, from the sentence-answer candidate again. We thus have ten dimension word-matching features, and they are listed in Table 2.

3.3.2 POS Features. POS can categorize words according to their grammatical properties. For example, by analysing verbs in a sentence, we are able to get the action, occurrence, or state of being the sentence described, and nouns denote the abstract or concrete entities in the sentence. Therefore, words with different POS tags may hold different kind of information. Consequently, in the reading comprehension task, classifying words by their POS tags can extract useful information based on different questions. For example, some what-questions may focus on nouns, whereas

[9]The tool is available at https://stanfordnlp.github.io/CoreNLP/coref.html.
[10]Lemmatisation is used to identify the word's dictionary form, e.g., apples→apple. The lemmatization is implemented with the help of Stanford CoreNLP, which is available at https://stanfordnlp.github.io/CoreNLP/lemma.html.

Table 2. Summary of Our Extracted Word-Matching Features

Features	Description
# Statement Word Match	The number of words occurred in both the statement candidate and the sentence.
% Statement Word Match	The ratio of words in the statement candidate that occurred in both the statement candidate and the sentence.
# Statement Bi-gram Match	The number of continuous two words occurred in both the statement candidate and the sentence.
# Statement Tri-gram Match	The number of continuous three words occurred in both the statement candidate and the sentence.
# Statement Lemma Match	The number of lemmatized word occurred in both the statement candidate and the sentence.
# Answer Word Match	The number of words occurred in both the answer candidate and the sentence.
% Answer Word Match	The ratio of words in the answer candidate that occurred in both the answer candidate and the sentence.
# Answer Bi-gram Match	The number of continuous two words occurred in both the answer candidate and the sentence.
# Answer Tri-gram Match	The number of continuous three words occurred in both the answer candidate and the sentence.
# Answer Lemma Match	The number of lemmatized word occurred in both the answer candidate and the sentence.

some how-question may focus on verbs. Previous work on answer selection [16, 49] in QA systems has demonstrated the effectiveness of POS features. Among all POS categories, four of them contain the most important information of sentences, i.e., nouns, verbs, adjectives, and adverbs. We hence extract POS features from sentence-statement candidate pairs by counting the number of nouns, verbs, adjectives, and adverbs occurred in both the sentence and the statement candidate, respectively. Newly added words have been used as a effective feature in previous studies of reading comprehension task [34]. Motivated by this idea, the numbers of new nouns and new verbs that appear in the statement but absent in the sentence are added into the feature set. Aforementioned matching features measure the coverage of the information in the statement, while these newly added words tell us how much information is not mentioned in the current sentence and may be hidden somewhere else. Similar to word-matching features, we also extract the POS features from sentence-answer candidate pairs. Table 3 lists all POS features we extract and our POS tag is annotated with the help of Stanford POS Tagger.[11]

3.3.3 Dependency Features. Dependency parsing aims at finding dependency relations among words in the sentences, and it constructs a dependency tree for each sentence to represent its syntactic structure. Figure 4 illustrates two examples of the dependency parsing. In the dependency trees, verbs are taken as the center of the structure, and other words are either directly or indirectly connected to the verb in terms of the directed links. It is hence convenient to analyze sentences with free word order. As the examples illustrated in the figure, these two sentences have the same meaning but with different descriptions. According to their dependency trees, the most

[11]Stanford CoreNLP POS Tagger is available at https://stanfordnlp.github.io/CoreNLP/pos.html.

Table 3. Summary of Our Extracted POS Features

Features	Description
# statement n. Match	The number of nouns occurred in both the statement candidate and the sentence.
# statement v. Match	The number of verbs occurred in both the statement candidate and the sentence.
# Statement adj. Match	The number of adjectives occurred in both the statement candidate and the sentence.
# Statement adv. Match	The number of adverbs occurred in both the statement candidate and the sentence.
# Statement New n.	The number of nouns that occurred in the statement candidate but missed in the sentence.
# Statement New v.	The number of verbs that occurred in the statement candidate but missed in the sentence.
# Answer n. Match	The number of nouns occurred in both the answer candidate and the sentence.
# Answer v. Match	The number of verbs occurred in both the answer candidate and the sentence.
# Answer adj. Match	The number of adjacents occurred in both the answer candidate and the sentence.
# Answer adv. Match	The number of adverbs occurred in both the answer candidate and the sentence.
# Answer New n.	The number of nouns that occurred in the answer candidate but missed in the sentence.
# Answer New v.	The number of verbs that occurred in the answer candidate but missed in the sentence.

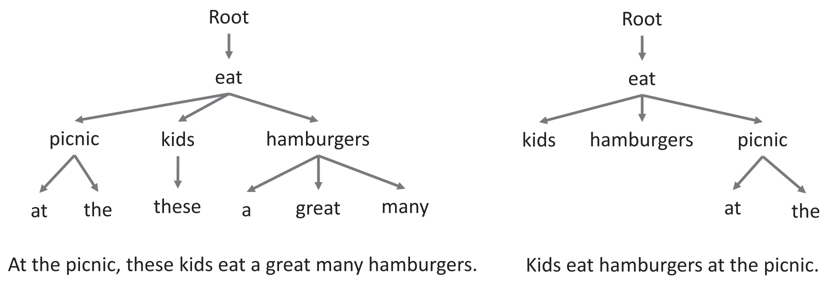

Root
eat
picnic
kids
hamburgers
at
the
these
a
great
many
At the picnic, these kids eat a great many hamburgers.
Root
eat
kids
hamburgers
picnic
at
the
Kids eat hamburgers at the picnic.

Fig. 4. Illustration of dependency parsing results of two sentences.

important information of these two sentences, i.e., eat, picnic, kids, and hamburgers, can be easily matched from the trees. As a result, the number of matched dependencies between the sentence and the statement candidate are added into the feature set. It is worth noting that the matched dependency means that for two given triple-form dependencies, i.e., (H, R, D), the head word H, the dependent word D, and their dependency relation R are all matched. In dependency trees, different paths from the root to leaf nodes describe different aspect of the sentence. For example, the path "eat→picnic→at" describes the location, while the path "eat→kids→these" denotes the involved people. To fetch this kind of information in the sentences, the number of matched relation

Table 4. Summary of Our Extracted Dependency Features

Features	Description
# Dependency Match	The number of dependencies exist in both the statement candidate and the sentence.
# Path Match	The number of dependency relation paths exist in both the statement candidate and the sentence.
Longest Path Match	The number of nodes in the longest matched dependency relation path.
Root Match	Whether the root node of the statement candidate and the sentence is matched. The value is 1 if they are matched, and the value is 0 other wise.

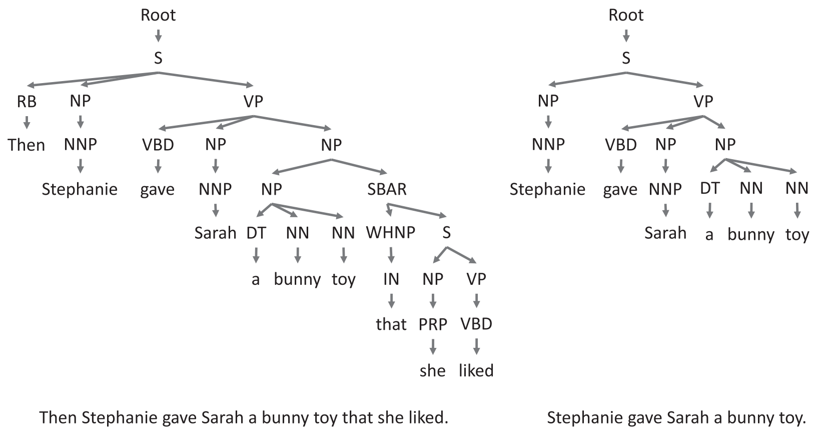


Fig. 5. Illustration of constituency parsing results of two sentences.

paths and the number of involved nodes in the longest matched relation path are utilized as two dependency features. Relation paths matching means all dependencies in the relation path are matched. Since the root node of the tree acts as the structural center of the sentence, it contains vital information of the sentence. Consequently, we compare the root nodes of the sentence and the statement candidate, and added the root node matching into the dependency feature set. All extracted dependency features are listed in Table 4, and all these dependency features are extracted with the help of the Stanford dependency parser.[12]

3.3.4 Constituency Features. Different from dependency parsing, constituency parsing focuses on organizing sentence words into a hierarchical structure according to the phrase structure grammars. As the examples illustrated in Figure 5, the sentence words that can construct a phrase are organized under the same subtree. Therefore, by analysing the structure of the constituency parsing tree, we are able to get the substructure of the sentence. Intuitively, if the statement is judged as correct according to the given sentence, they share more similar phrases (substructures). We hence use the number of matched sub-trees between the answer candidate and the sentence as a constituency feature. The sub-tree matching means that both the roots and the children are

[12]Stanford dependency parser is available https://stanfordnlp.github.io/CoreNLP/depparse.html.

Table 5. Summary of Our Extracted Constituency Features

Features	Description
# Sub-tree Match	The number of sub-trees exist in both the statement candidate and the sentence.
# Node Match	The number of nodes in the largest matched matched sub-trees between the statement candidate and the sentence.
Answer Sub-tree match	Whether the words in the answer candidate are all leaves of a sub-tree in the sentence. If it is true, then the feature value is 1, else the value is 0.

matched. Since a sub-tree in the constituency parsing usually represents a phrase in the sentence, a larger sub-tree with more nodes means that more information is involved. Hence, if a larger tree is matched, the statement is more likely to share the same meaning with the sentence. For example, in Figure 5, the sub-tree constructed with three words "a bunny toy" is matched between the two given sentences, so they are more likely to describe the same content than the situation when only a small sub-tree constructed with one word "toy" is matched. So, the number of nodes in the largest matched sub-tree is added into the constituency feature set. As we discussed in word-matching features, the answer content is decisive in determining the correctness, we hence assume that if the words in the answer are matched with a phrase in the sentence, the answer is more likely to be correct. As a result, the match between answer candidates and sub-tree leaf nodes in the sentence is added into the feature set. We list all the extracted constituency features in Table 5. In this article, the constituency parsing is implemented with the help of Stanford constituency parser[13].

3.3.5 NE Features. Named entities usually denote the names of people, locations, and organizations. They play vital roles in measuring the correctness of the given statement, because the sentence usually discusses something about the involved entities. Therefore, if the same named entity appeared in both the sentence and the statement candidate, they are more likely to talk about the same thing. So the number of matched named entities are used as the NE features. Considering the MCTest dataset we utilize, the texts are children's stories and they rarely contain organizations, so only the number of matched names of people and locations are used in this work. Beyond the name of people and locations, there are many statements and sentences talking about numbers and datetimes, such as the sentence "*Lisa is excited, because on Saturday, Whiskers turns two years old.*" To this end, whether there are matched numbers and matched datetimes are taken into consideration as our NE features. The extracted NE features are listed in Table 6 and it is noted that the recognition of named entity, number and datetime is implemented with the help of Stanford NER classifier.[14]

3.3.6 SR Features. Semantic role labeling (SRL) is a traditional natural language processing task aiming at detecting the semantic arguments associated with predicates and classifying them into specific roles according to their grammatical functions. It has the ability to simplify the sentence into a formalized structure. In particular, the analysis of a sentence contains two parts, i.e., the predicates and the corresponding semantic arguments. These two parts jointly describe a complete meaning of the sentence. The predicate is usually a verb, and it is the center of the sentence. As a complementary, the semantic arguments describe different properties of the predict, such as the

[13]Stanford constituency parser is available at https://stanfordnlp.github.io/CoreNLP/parse.html.
[14]Stanford NER Classifier is available at https://stanfordnlp.github.io/CoreNLP/ner.html.

Table 6. Summary of Our Extracted NE Features

Features	Description
# People Match	The number of people's names appeared in both the statement candidate and the sentence.
# Location Match	The number of locations' names appeared in both the statement candidate and the sentence.
Number Match	Whether there are matched numbers between the statement candidate and the sentence. If there are any, then the feature value is 1, else the value is 0.
Datetime Match	Whether there is matched datetimes between the statement candidate and the sentence. If there are any, then the feature value is 1, else the value is 0.

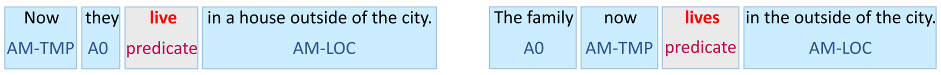


Fig. 6. Illustrate semantic labeling results of two sentences. The red parts indicate the predicates of the sentences and the blue parts indicate the corresponding arguments.

agent, the recipient, and the location. Figure 6 displays two examples of SRL. In these two examples, they all contain one predicate, i.e., "live," but it is notable that it is a common phenomenon that a sentence contains more than one predicate, and different predicts describe different aspects of the sentence. The first sentence describes three semantic arguments of the predicate "live," i.e., the datetime "now," the agent "they," and the location "in a house outside of the city," and the semantic arguments of the predicate "lives" in the second sentence are the agent "The family," the datetime "now" and the location "in the outside of the city." We find that if two sentences have the same meaning, the predicates and their corresponding semantic arguments have more overlapping. As a result, for a given statement candidate and a sentence, the number of overlapped words in the semantic argument of "Agent" (A0), "Recipient" (A1), "Other Objects" (A2-A5), "Datetime" (AM-TMP), "Location" (AM-LOC), "Cause" (AM-CAU), "Purpose" (AM-ADV and AM-PNC), and other arguments, are, respectively, used as SR features, when their predicates are matched. In some cases, the predicates of the statement candidates and the sentences have the same meaning with various words. For instance, in the sentence "*Mandy likes making pictures of flowers*" and the statement candidate "*Mandy likes to paint flowers,*" the predicates "make" and "paint" hold the same meaning. The number of overlapped words in the semantic argument without considering the predicate matching is hence added into the SR feature set. Since there are too many common semantic arguments of A0-A5 for different predicates, the matched words of these arguments are very noisy if the predicates are not considered. So only the overlapped words of "datetime," "location," "cause," and "purpose" are utilized. All SR features utilized in this article are listed in Table 7. These SR features are extracted with the help of mate-tools NLP.[15]

3.3.7 Semantic Features. Word embedding is widely used as semantic feature in various tasks, such as answer selection [60], text classification [57], and retrieval [72]. It represents words with a k-dimensional semantic space, where similar words are located closer to each other. To measure

[15]Mate-tools NLP is available at https://code.google.com/archive/p/mate-tools/.

Table 7. Summary of Our Extracted SR Features

Features	Description
# Pred. A0 Match	The number of matched words in the semantic arguments of A0 with the condition that their predicates are same.
# Pred. A1 Match	The number of matched words in the semantic arguments of A1 with the condition that their predicates are same.
# Pred. A2-A5 Match	The number of matched words in the semantic arguments of A2-A5 with the condition that their predicates are same.
# Pred. Location Match	The number of matched words in the semantic arguments of AM-LOC with the condition that their predicates are same.
# Pred. Datetime Match	The number of matched words in the semantic arguments of AM-TMP with the condition that their predicates are same.
# Pred. Cause Match	The number of matched words in the semantic arguments of AM-CAU with the condition that their predicates are same.
# Pred. Purpose Match	The number of matched words in the semantic arguments of AM-ADV and AM-PNC with the condition that their predicates are same.
# Pred. Other Match	The number of matched words in other semantic arguments with the condition that their predicates are same.
# Location Match	The number of matched words in the semantic arguments of AM-LOC without considering their predicates.
# Datetime Match	The number of matched words in the semantic arguments of AM-TMP without considering their predicates.
# Cause Match	The number of matched words in the semantic arguments of AM-CAU without considering their predicates.
# Purpose Match	The number of matched words in the semantic arguments of AM-ADV and AM-PNC without considering their predicates.

the semantic similarity between the statement candidate and the sentence, the cosine distance between them in the semantic space is calculated as the semantic feature. The representation of the statement candidate and the sentence is obtained by averaging the embeddings of their words. In this work, we use the pre-trained 300-dimensional GloVe word embedding[16] to calculate the cosine distance.

3.3.8 Coreference Features. As we have mentioned that coreference is essential in reading comprehension, and we utilize coreference resolution to replace the pronouns with their referred entities. Besides, the coreference can link different sentences and the statements by referring the their mentions into the same entity. Different from the named entity matching, the coreference is able to recognize other entities besides people's name. For example, in the sentence "*Lauren gives Lulu her lamb to sleep with*" and the statement "*Lauren gives her the lamb and Alan gives her his blanket,*" the mention "lamb" denotes the same one. To fetch this kind of relations between the sentence and the statement candidate, the number of linked coreferences between them is used as a coreference feature. If there are multiple entities sharing the same coreference, then we count them only once. Since the answers are decisive to decide the correctness, the linked references

[16]The word embeddings are available at https://nlp.stanford.edu/projects/glove/.

Table 8. Summary of Our Extracted Coreference Features

Features	Description
# Statement Coref.	The number of linked coreference entities between the statement candidate and the sentence.
# Answer Coref.	The number of linked coreference entities between the answer candidate and the sentence.

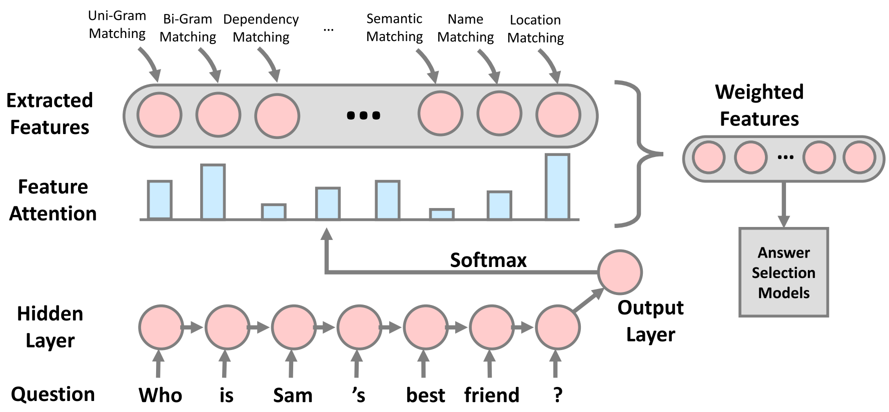


Fig. 7. Schematic illustration of the question-oriented feature attention mechanism.

between the sentence and the answer candidate are also considered as another coreference feature. All our extracted coreference features are list in Table 8.

3.4 Question-Oriented Feature Attention Mechanism

In this article, we assume that different questions may focus on different aspects of information conveyed by the text. Therefore, we propose the QFA mechanism to identify the core information related to the given question, so that the information can be better used.

To solve questions according to the given text, a rich set of features for each sentence-statement pair are extracted. It is well known that different features encode different information. Selecting valuable information is to capture the useful features and omit the others. This is somehow similar to the traditional attention mechanisms as they both automatically select useful information for the specific task. But the main difference between traditional attention mechanism and the proposed QFA mechanism is that the former focuses on selecting important sentences or words with respect to the given task, while the later selects useful features according to the given question.

The overall framework of QFA is illustrated in Figure 7. In particular, the model generates the feature attention according to the given question. The extracted features are then weighted by the feature attention. At last, these weighted features are fed into answer selection models for further inference.

As to infer the feature attention, the Recurrent Neural Network (RNN) is employed in this work. RNN is widely utilized to model sequence data, such as sentences, and it has been demonstrated powerful to capture the semantic meaning of language in verious tasks [24, 50, 52]. It is widely accepted that some extension models, such as long short-term memory network (LSTM) and bidirectional RNNs, is superior to classical RNN in representation, but due to the small dataset we utilize in our experiments (We utilize MCTest in this article, and it will be discussed later.), we simply employ the classical RNN in this article to alleviate the overfitting problem.

Formally, in QFA, a question q with T words is represented as a sequence of word embeddings, i.e., $Q = \{\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_T\}$, where $\mathbf{q}_t$ denotes the k-dimensional word embedding of the tth word in q. In this work, we use the pre-trained word embeddings by GloVe[17] [38]. These embeddings are sequentially fed into the RNN. It is represented as

$$\mathbf{h}_t = \sigma\left(\mathbf{W}_{ih}\mathbf{q}_t + \mathbf{W}_{hh}\mathbf{h}_{t-1} + \mathbf{b}_h\right), \tag{1}$$

where $\mathbf{h}_t$ and $\mathbf{h}_{t-1}$ are, respectively, the output states of the tth and the $(t-1)$th hidden units, $\mathbf{W}_{ih}$ is the weight matrix from the input to the hiden unit, $\mathbf{W}_{hh}$ is the weight matrix between two hidden units, $\mathbf{b}_h$ is the bias vector of the hidden unit, and $\sigma(\cdot)$ is the activation function. Once the last word of the question is fed into the model, the output state of the hidden unit $\mathbf{h}_T$ is sent to the output layer to generate the feature attention $\mathbf{a}$ as follows:

$$\mathbf{a} = softmax\left(\mathbf{W}_{ho}\mathbf{h}_T + \mathbf{b}_o\right), \tag{2}$$

where $\mathbf{W}_{ho}$ is the weight matrix from the hidden unit to output, $\mathbf{b}_o$ is the bias vector of the output layer, and $\mathbf{h}_T$ is the output state of the last hidden layer. We expect that the RNN model is able to automatically learn the pattern of feature distribution according to the semantic information in the question, so that when a new question is observed, it can output a corresponding distribution of the feature importance. It is noted that the softmax function, i.e., $y_j = \frac{e^{z_j}}{\sum_{m=1}^{M} e^{z_m}}$, is employed as the activation function. This is because its output is naturally a distribution, and it is thus reasonable to distinguish the importance of different features with a softmax function.

Suppose we extract M different features from each sentence-statement pair, we thus have a M-dimensional feature vector $\mathbf{x}$ to infer the correct answer. To emphasize the important features, we weight these features by the feature attention as follows:

$$\widehat{\mathbf{x}} = \mathbf{a} \odot \mathbf{x}, \tag{3}$$

where $\odot$ denotes the element-wise multiplication of feature attention $\mathbf{a}$ and feature vector $\mathbf{x}$. The weighted features are then used for further answer inference. By integrating feature attentions, the weighted features are capable of distinguishing the useful features from the useless ones according to the given question. It is noted that the proposed QFA is trained together with answer selection model. Therefore, sentence and statement information is back propagated via gradients. In this way, the well-trained QFA is able to distinguish engineering features according to the given new question.

3.5 The Proposed QFAReader Model

In this section, we detail our proposed reading comprehension model, QFAReader. It utilizes the aforementioned QFA to efficiently harvest the extracted features. Additionally, to judge the answer according to multiple sentences in the given text, we assume that each sentence in the text may contain incomplete yet useful information to select the correct answer. Even though some simple questions can be directly inferred from one single sentence of the text, other sentences may still convey rich information to indicate the correct answer indirectly. In this way, some complex questions, requiring multiple sentence to infer the correct answer, can be solved.

In light of this, the text $\mathcal{T}$ is split into sentences, i.e., $\mathcal{T} = \{t_1, t_2, \ldots, t_N\}$. We regard the statement and every single sentence t_n pair separately, and the final decision is made by jointly considering all these sentences at last. The overall structure of the QFAReader is illustrated in Figure 8.

[17]The word embeddings are available at https://nlp.stanford.edu/projects/glove/.

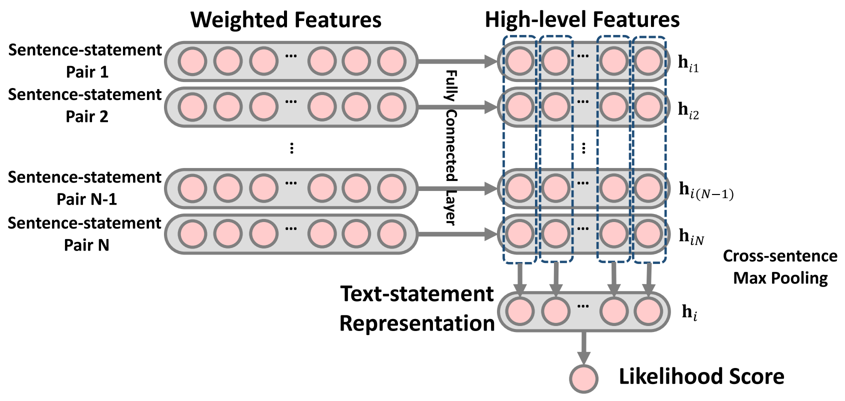


Fig. 8. Framework of the proposed QFAReader. The QFA-weighted features for sentence-statement pairs are first transformed into a latent space with a fully connected layer, and then merged together with cross-sentence max pooling. The likelihood of the statement is finally predicted with a perceptron.

In QFAReader, we extract a rich set of features for each sentence-statement pair to measure the correctness of the statement. These features are then weighted by the proposed QFA as described in Section 3.4. The weighted feature vector for the ith statement candidate and the nth sentence is denoted as $\widehat{\mathbf{x}}_{in}$, and they are transformed into the latent space with a fully connected layer as follows,

$$\mathbf{h}_{in} = \sigma\left(\mathbf{W}_h \widehat{\mathbf{x}}_{in} + \mathbf{b}\right), \tag{4}$$

where $\mathbf{h}_{in} = [h_{in}^1, h_{in}^2, \ldots, h_{in}^D]^T \in \mathbb{R}^D$ is a D-dimensional representation of $\widehat{\mathbf{x}}_{in}$ in the latent space, $\mathbf{W}_h$ is the weight matrix, $\mathbf{b}$ is the bias vector, and $\sigma(\cdot)$ is the activation function. In this work, we utilize ReLU as the activation function due to its fast speed during optimization. Via this transformation, high-level feature patterns are extracted from the original surface features, and they are more efficient to describe inherent information in sentence-statement pairs.

For reading comprehension, we assume that all sentences in the text contain information for solving the given question, yet their importance varies a lot. For example, when solving the question "*Where did the school have the picnic?*" according to two given sentences "*The whole school went outside*" and "*They had a picnic to celebrate Ana winning*," we find that the first sentence contains more information about the "location" of having the picnic, whereas the second one is more likely to tell us the "reason" of having this picnic. Even QFA have the ability to distinguish feature importance, the learned feature attention is assigned to all sentences in the passage. Therefore, for a given question, different sentences in the same passage have the same feature weight distribution and the same features are emphasized. It means that the "location" and "reason" information is simultaneously emphasized for all sentences in the example. Actually, the "reason" in the first sentence and the "location" in the second sentence may be less useful. To fuse these sentence-statement pairs together to infer the correctness of the statement, we construct the latent representation of the text-statement candidate pair $\mathbf{h}_i = [h_i^1, h_i^2, \ldots, h_i^D]^T \in \mathbb{R}^D$ with cross-sentence max pooling, which incorporate the latent representation of all sentence-statement candidate pairs simultaneously, where h_i^k indicates the kth element in $\mathbf{h}_i$. In the cross-sentence max pooling, the max value of all the kth elements among $\{\mathbf{h}_{in} | n = 1, 2, \ldots, N\}$ is selected as h_i^k. It is formulated as

$$h_i^k = max_n\left(\left\{h_{in}^k | n = 1, 2, \ldots, N\right\}\right), \tag{5}$$

where N is the number of sentences in the text. This is somehow similar as the traditional max pooling technique in the convolutional neural network, which filters out the most important information among different regions [66]. By selecting the largest value in the same latent space, the generated representation of the text-statement candidate pair only maintains the most useful information for solving the given question.

In the light of this, all the sentence-statement pairs are encoded into one text-statement pair. And it is used to predict the correctness of the statement candidate with a perceptron model. It is formulated as

$$y_i = \sigma\left(\mathbf{W}_o \mathbf{h}_i + b_o\right), \tag{6}$$

where $\mathbf{W}_o$, b_o, and $\sigma(\cdot)$ are the weight matrix, the bias vector, and the activation function, respectively. The output y_i denotes the correctness of the ith statement candidate, and a larger value indicates the statement is more likely to be correct. Therefore, given the question Q, the extracted features for each sentence-statement candidate $\mathcal{X} = \{\mathbf{x}_{in} | i = 1, 2, \ldots, M; n = 1, 2, \ldots, N\}$, and parameters θ of the model, the index of the predicted correct answer $\widehat{i}$ for a new question q, text $\mathcal{T}$, and answer candidates $\mathcal{A} = \{a_1, a_2, \ldots, a_M\}$ is given by

$$\widehat{i} = \arg\max_i f(Q, \{\mathbf{x}_{in} | n = 1, 2, \ldots, N\} | \theta), \tag{7}$$

where $f(\cdot)$ denotes our proposed QFAReader. Since features for all sentences of the text are fed into the model simultaneously, the sentence index n can be omitted for simplification, and we hence rewrite $f(Q, \{\mathbf{x}_{in} | n = 1, 2, \ldots, N\}$ as $f(Q, \mathbf{X}_i)$.

To optimize QFAReader, given the training samples $\{Q^k, \{\mathbf{x}_{in} | n = 1, 2, \ldots, N\}^k, i^*\}_{k=1}^K$, where i^* is the index of the correct answer, and K is the number of training samples, we minimize the pair-wise hinge loss function L,

$$L = \sum_{k=1}^{K} \left[-f(Q^k, \mathbf{X}_{i^*}^k) + \epsilon + \max_{i \neq i^*} f(Q^k, \mathbf{X}_i^k) \right]_+, \tag{8}$$

where $[x]_+$ denotes the positive part of x and $\epsilon > 0$ is a margin hyperparameter. $\mathbf{X}_{i^*}$ is the features extracted based on the correct answer and the text, and $\mathbf{X}_i$ is the features extracted according to the incorrect answer and the text. In the answer selection task, there are only one correct answer and multiple incorrect ones for the given question. To balance the number of positive and negative samples, we consider the main idea of support vector machine, which determine the decision boundary according to the data points closest to the decision boundary. Therefore, in QFAReader, we use the correct answer as the positive sample and the incorrect one with the highest score as the negative sample in each iteration [42, 51, 56]. It is noted that the negative sample involved in model training may changed in different iterations, as the likelihood scores for all samples are keep changing during the training process.

When it comes to optimization, stochastic gradient descent (SGD) [34] is employed. The parameters we have to learn are the weight matrices and bias vectors in QFA (i.e., $\mathbf{W}_{ih}$, $\mathbf{W}_{hh}$, $\mathbf{W}_{ho}$, $\mathbf{b}_h$, and $\mathbf{b}_o$) and QFAReader (i.e., $\mathbf{W}_h$, $\mathbf{W}_o$, $\mathbf{b}$, and b_o). It is worth noting that to minimize the impact of overfitting problem, dropout [48] is adopted to the RNN model and the weight decay are used for all learnable parameters.

4 EXPERIMENTS AND RESULTS

In this section, we present our experimental results to demonstrate the effectiveness of the proposed feature attention mechanism and QFAReader in the reading comprehension task.

Table 9. Statistics of MC160 and MC500 Datasets

Dataset	# texts				# Questions			
	TRAIN	DEV	TEST	Total	TRAIN	DEV	TEST	Total
MC160	70	30	60	160	280	120	240	640
MC500	300	50	150	500	1,200	200	600	2,000

4.1 Dataset Description

In this article, we focus on selecting the correct answer to a multiple-choice question and MCTest [41] fits the task well. MCTest is a reading comprehension dataset with a set of texts and associated questions. These texts are fictional children's stories so that the answers cannot be obtained without the texts. In this dataset, each text has four multiple-choice questions, each with four answer candidates, and only one answer is correct for each question. There are two subsets in MCTest, i.e., MC160 and MC500, and they contain 160 and 500 texts, respectively. These texts are divided into training set, development set, and testing set as shown in Table 9.

4.2 Evaluation Metric and Baselines

Accuracy is used as the metric in this work. It reports the ratio of questions that are correctly answered by the system. There are two kinds of questions in the dataset, i.e., "*one*" and "*multiple*". The "one" questions are answerable with a single sentence from the text, while the "multiple" questions require the context of multiple sentences. Beside the overall accuracy of all questions, the accuracy of these two kinds of questions are also reported separately in the experiments.

To verify the effectiveness of our proposed QFAReader, we compared it with the following baselines.

- Sliding Window [41]: This method is proposed by MCTest dataset provider. It uses a sliding window over texts to compare the similarity between the text and the question-answer candidate pair. The answer candidate most similar to the text is regarded as the correct answer.
- Sliding Window+Word Distance [41]: This is an extension method of Sliding Window. It subtracted the word distance-based score from the sliding window-based score to arrive at the final score. Similar to the Sliding Window, a larger score indicates the answer candidate is more likely to be correct.
- Sliding Window+Word Distance+RTE [41]: Similar to our QFAReader, this method transforms the question and the answer candidates into statements with a rule-based method. Sliding window and word-distance-based methods are then utilized to measure the similarity between the statement and the text.
- Logistic Regression: This is a point-wise learning to rank method, and it is widely used in traditional answer selection or answer ranking problems [43]. In this method, we use the mean value of the features extracted from different sentence-statement pairs as the feature of the text-statement pair. The classification model tries to separate the correct statement from others.
- RankSVM [23]: This is the most widely used pair-wise learning to rank method. It combines the correct answer candidate and each incorrect one as an ordered pair, and then uses the SVM model to find the correct order. Similar as Logistic Regression, the representation of text-statement pair is constructed by averaging the features from all sentence-statement pairs. RankSVM is implemented with the help of existing tool, SVMRank.[18]

[18]SVMRank is available at http://www.cs.cornell.edu/people/tj/svm_light/svm_rank.html.

- ListNet [9]: This is a list-wise learning to rank method, and it uses a neural network to directly optimize the list-wise ranking loss. The features used in this method are similar as Logistic Regression and RankSVM. ListNet is implemented with the help of existing tool, RankLib.[19]
- Smith et al. [46]: This method extends the sliding window baseline. It scores each answer by making multiple comparisons between the text and the QA pair by ranging the window size from two to 30. The final evaluation score is the sum of the obtained scores.
- Sachan et al. [42]: This method assumes that there are answer-entailing structures representing the correctness of the statement and the answer-entailing structures are composed of the snippets from the text. A latent structural SVM is learned to minimize the distance between answer-entailing structure and the statement.
- Wang et al. [56]: This method compares the similarity between the text and the statement constructed from the question and the answer candidate. A rich set of syntax features, frame semantic features, and word embedding features are extracted and fed into a pair-wise ranking model to select the correct answer.
- Yin et al. [70]: This method uses convolutional neural network to obtain the representation of every sentence in the text. Then it employs the attention mechanism to emphasize important sentences and merges the sentence representation as a overall representation. The overall representation is finally used to select the correct answer.
- Trischler et al. [51]: This method constructs a parallel-hierarchical neural network. Word embeddings of sentences and QA pairs are fed into the model in two different perspectives, i.e., semantic and word-by-word. The word-by-word perspective consists of three views, i.e., sentential, sequential sliding window, and dependency sliding window. These components evaluate the similarity between the text and the QA pairs simultaneously.

4.3 Overall Performance

We first compared our QFAReader with the aforementioned baselines on MC160 and MC500 datasets. The parameters were carefully tuned according to the performance on the development set and the accuracy on the testing set are reported. The overall performance is summarized in Table 10. It is worth noting that the performance of baselines are reported according to the corresponding publications. From the table, we have the following observations: (1) Our proposed QFAReader outperforms all baselines on the MC500 dataset, whereas on the MC160, there is a small gap compared with Sachan et al. (2015) and Wang et al. (2015) in "multiple" questions. The gap is mainly caused by the limited size of the MC160 data. Since the neural network is employed in QFAReader, the model is easy to be overfitting when only few training data are observed. When sufficient training data (i.e., MC500) are utilized, QFAReader outperforms other baselines in both "one" and "multiple" questions. This shows the effectiveness of our proposed QFAReader in the reading comprehension task. (2) Compared the performance on "one" and "multiple" questions, all methods including our QFAReader achieve higher accuracy on "one" questions over both MC160 and MC500 datasets. This is because that most "one" questions can be answered by simply comparing the similarity between the text and the QA pair, while "multiple" questions are much more complex. Beside comparing sentence similarities, reasoning and inference are required to resolve "multiple" questions. (3) When comparing our QFAReader with other baselines in different questions, we find that QFAReader obtains a larger performance enhancement on "one" questions. This is because that most extracted features used in QFAReader are used to measure the similarity between the sentence and the statement

[19]RankLib is available at https://sourceforge.net/p/lemur/wiki/RankLib/.

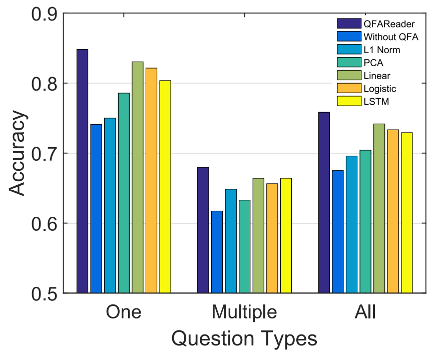
Table 10. Overall Performance Comparison Among Different Baselines

Methods	MC160 Accuracy (%)			MC500 Accuracy (%)		
	One	Multiple	All	One	Multiple	All
Sliding Window	64.73	56.64	60.41	58.21	56.17	57.09
Sliding Window+Word Distance	75.89	60.15	67.50	64.00	57.46	60.43
Sliding Window+Word Distance+RTE	76.78	62.50	69.16	68.01	59.45	63.33
Logistic Regression	71.43	60.94	65.83	68.01	58.84	63.00
RankSVM	73.21	61.72	67.08	68.38	59.45	63.50
ListNet	74.11	60.94	67.09	68.01	59.76	63.50
Smith et al. (2015)	78.79	**70.31**	75.77	69.12	63.34	65.96
Sachan et al. (2015)	—	—	—	67.65	67.99	67.83
Wang et al. (2015)	84.22	67.85	75.27	72.05	67.94	69.96
Trischler et al. (2016)	79.46	**70.31**	74.58	74.26	68.29	71.00
Yin et al. (2016)	63.3	62.9	63.1	54.2	51.7	52.9
QFAReader	**84.82**	67.97	**75.83**	**78.31**	**68.69**	**73.00**

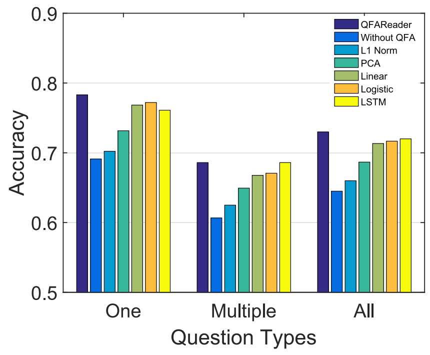
candidate. These features are much more appropriate to solve "one" questions. Therefore, with the help of the proposed feature attention mechanism, these features can be more effectively utilized to enhance the performance. While "multiple" questions require more reasoning and inference, and therefore, more complex features are required. (4) QFAReader and learning to rank methods (i.e., Logistic Regression, RankSVM, and ListNet) utilize the same features to select answers. From the table, we observe that QFAReader outperforms these three learning to rank methods. This is mainly because that they cannot distinguish feature importance according to different questions, and they cannot fuse features from different sentence effectively. (5) Among all baselines, Sliding Window, Sliding Window+Word Distance, and Sliding Window+Word Distance+RTE only utilize simple word matching, while other methods are mostly based on feature engineering. We find that the latter methods outperform the former ones. This illustrates the effectiveness of the features they utilized. It is notable that, most of features used in our QFAReader are designed according to them, so it shows the reasonability of our features in a sense. (6) The main difference between the baseline Sliding Window+Word Distance and the baseline Sliding Window+Word Distance+RTE is that the later utilizes the constructed statement to calculate the answer score, instead of using the concatenated QA pair directly. The better performance of Sliding Window+Word Distance+RTE demonstrates that transforming QA pairs into statements is more effective in the reading comprehension task.

4.4 Feature Attention Analysis

In our QFAReader, the feature attention mechanism is utilized to emphasize important features according to the given question. This can be regarded as a feature selection mechanism by valuing the useful information more. To demonstrate the effectiveness of the proposed feature attention mechanism, we compared the QFAReader with two methods with different feature selection mechanisms, i.e., Without QFA, L1 norm, PCA, linear regression, logistic regression, and LSTM. WithoutQFA is modified from QFAReader by removing the feature attention mechanism. The extracted features are fed into the answer selection model directly. This method does not distinguish the importance of different features. L1 norm is a widely used feature selection method [36]. It automatically finds important features by regularizing the feature weight in the answer selection



(a) Performance comparison on MC160 dataset

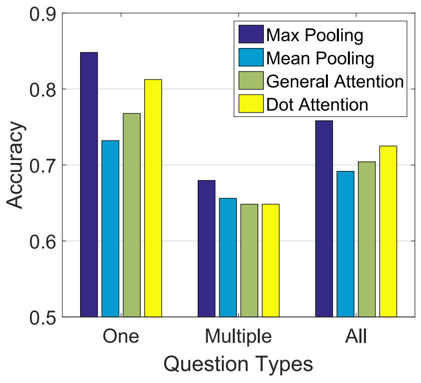


(b) Performance comparison on MC500 dataset

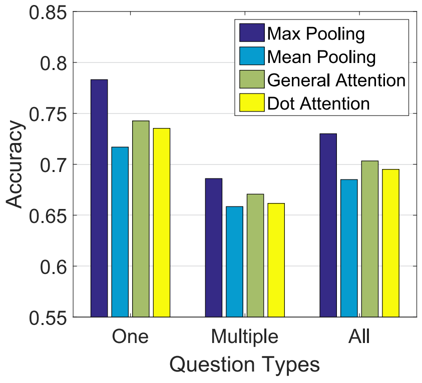
Fig. 9. Performance comparison w.r.t. different feature selection methods on MC160 and MC500 datasets.

model. Principal Component Analysis (PCA) algorithm is a widely used feature selection method in data pre-processing. It selects the most representative features from the feature set and discards the redundant ones. In the PCA method, we first selected the most important features with PCA algorithm,[20] then utilized the pre-processed features to select the correct answer with the answer selection model as WithoutQFA. Linear regression, logistic regression, and LSTM are most widely used regression and classification models for various tasks. We hence replaced the RNN-based QFA with a linear regression, a logistic regression, and an LSTM, respectively. Linear regression and logistic regression use the average value of question word embeddings as the input, and the output is used as the feature attention. The question word embeddings are sequentially fed into LSTM to calculate the feature attention. The linear regression, logistic regression , and LSTM are jointly optimized with answer selection model, similar as QFAReader. The results are displayed in Figure 9. From the figure, we have the following observations: (1) Compared with WithoutQFA, L1 Norm and PCA performs better in both MC160 and MC500 dataset. This demonstrates that distinguishing feature importance plays vital role in reading comprehension task. By using PCA or L1 norm, important information in the feature set is reserved and redundant one is discarded, so it is able to get a better performance. (2) QFAReader, linear regression, logistic regression, and LSTM have better performance than L1 Norm and PCA. Different from traditional feature selection methods (e.g., L1 Norm and PCA), QFA-based methods (e.g., QFAReader, Linear, Logistic, and LSTM) select important features according to the given question. Since different questions may focus on different aspect of the text, traditional answer selection methods can hardly select the real important features, since question contents are ignored. This demonstrates that question contents have important information to find useful features, and distinguishing the feature importance according to the given question is effective to select correct answers. (3) QFA-based methods perform better on both MC160 and MC500 datasets. This demonstrates that the QFA mechanism is useful to enhance the performance even when only few questions are observed. (4) No matter in "one" questions or "multiple" questions, QFA-based methods achieve a significant improvement compared with L1 Norm, PCA, and WithoutQFA. This demonstrates the applicability of the QFA mechanism. It is able to enhance the effectiveness of feature utilization in answering both "one" and "multiple" questions. (5) From the comparison among QFAReader, Linear, Logistic, and LSTM, we find that QFAReader has a slight lead, but the gap is quite narrow. This shows that both neural

[20]The PCA algorithm is implemented with the help scikit learn. It is available at http://scikit-learn.org/.



(a) Performance comparison on MC160 dataset



(b) Performance comparison on MC500 dataset

Fig. 10. Performance comparison w.r.t. different sentence merging methods on MC160 and MC500 datasets.

methods and traditional regression methods are able to distinguish feature importance according to the question content, but RNN is more appropriate to capture the sequence information in the question. (6) QFAReader and LSTM achieve similar performance on MC500 dataset, but their performance gap is much larger on MC160 dataset. RNN and LSTM are all effective in capturing semantic information of complex questions, but LSTM is more likely to overfit on a small dataset.

4.5 Sentence Merging Analysis

In QFAReader, the max pooling cross different sentence-statement pairs is utilized to generate the latent representation of the text-statement pair. In this experiment, we explored three different methods to obtained the latent representation of text-statement pair, i.e., Mean Pooling, Dot Attention, and General Attention. (1) Different from the Max Pooling utilized in QFAReader, Meaning Pooling utilizes the average value of the latent represent representations of all sentence-statement pairs to represent the text-statement pair. (2) Dot Attention is a widely adapted attention mechanism [29, 45]. It uses the dot product between the rectorial representations of the sentence and the statement to calculate the sentence weight, i.e., $\mathbf{h}_t^T \mathbf{h}_s$. Then the weighted sum of the latent representation of all sentence-statement pairs is used as the text-statement pair. It is worth noting that the vectorial representations are calculated by the pre-trained word embeddings. (3) General Attention is another attention mechanism and it is more powerful than Dot Attention [29]. Different from Dot Attention, this method learns a weight matrix $\mathbf{W}_a$ to map the sentence and the statement into the same space and calculates the attention, i.e., $\mathbf{h}_t^T \mathbf{W}_a \mathbf{h}_s$. Similar to Dot Attention, the vectorial representations are obtained by the pre-trained word embeddings.

Figure 10 displays the comparison result. From the figure, we observe the following: (1) Max Pooling performs the best among the others. This demonstrates that Max Pooling is effective in capturing the most important information among different sentences. (2) The performance of Mean Pooling is not as good as others. This is because that not all of the QFA-emphasized information is useful to answer the question as we described in Section 3.5, and most sentences contain noise for the answer inference. Therefore, mean pooling may fuse the features of less importance into the text-statement pair representation. (3) Max Pooling performs better than attention mechanisms, i.e., Dot Attention and General Attention. This is because attention mechanisms select useful sentences and emphasize all information in the selected sentences, while Max Pooling only selects useful information from different sentences. This supports our second assumption that each

Table 11. Performance Comparison w.r.t. Removing Different Features

Feature Set	MC160 Accuracy (%)			MC500 Accuracy (%)		
	One	Multiple	All	One	Multiple	All
All Features	**84.82**	**67.97**	**75.83**	**78.31**	**68.69**	**73.00**
– Word Matching	75.89	64.06	69.58	74.63	63.11	68.33
– POS	81.25	66.41	73.33	75.74	64.02	69.33
– Dependency	74.11	63.28	68.33	71.69	60.06	65.33
– Constituency	79.46	64.84	71.67	78.31	64.94	71.00
– NE	81.25	64.06	72.08	72.06	60.37	65.67
– SR	81.25	68.75	74.58	77.57	66.77	71.67
– Semantic	78.57	67.19	72.50	74.63	64.33	69.00
– Coreference	80.36	65.62	72.50	76.10	67.07	71.17

sentence in the text may provide incomplete yet useful information to infer the correct answer, hence impact the performance. (4) Comparing these two attention mechanisms, we find that Dot Attention performs better on the MC160 dataset and General Attention performs better on MC500. This is because that the number of questions in MC160 is quite limited and General Attention contains many trainable parameters, so it is more likely to be overfitting on MC160 dataset.

4.6 Feature Analysis

In QFAReader, a rich set of features were extracted to measure the correctness of the answer and these features are divided into eight categories. To demonstrate the effectiveness of these features, we separately removed each category from the full feature set and evaluated the performance. The results are displayed in Table 11. It is observed that the model performs the best when all features are involved, and removing any feature category undermines the performance. This reveals that every feature category contains useful information to measure the correctness of the answer candidate. By comparing these feature categories, we find that if the dependency features are removed, the accuracy drops significantly in both MC160 and MC500 dataset. This suggests that the dependency features contribute most among all these eight categories.

4.7 Case Study

To gain insight into our model and reveal further research, we selected three questions from MCTest and visualized their feature attention calculated by QFAReader. The examples are listed in Table 12 and their corresponding feature attentions are shown in Figure 11. From the weight distribution, we find that features of Tri-gram Matching, # Statement Verb Matching, and # Sub-tree Matching have larger feature weights compared with other features in all these three examples. This demonstrates that these three features are relatively important to distinguish answer correctness, and this is somehow consistent with the result of feature analysis in Table 11. In the first example, the question asks about Millie's thought, and feature attention shows that to answer this question, the feature of # Statement Verb Matching plays the most important role. By analyzing its answer candidates, we find that the emphasized features are more likely to separate the correct answer form others. When answering the second question, the feature of # of Statement Coref. is emphasized. This is mostly because there is a pronoun, *"her,"* in the question. In the third example, QFA emphasizes both # Statement Verb Matching and # Pred. A1 Matching. By analyzing the answer candidates, we find that it is reasonable to emphasize these two features, as Verb Matching is able to find "*eating*" and "*watching,*" and Pred. A1 Matching tries to find "*crackers*" and "*TV*".

Table 12. Example Selected from MCTest to Visualize Their Corresponding Question Attentions

Example 1	
Text	[...] She went to the library, and there, she saw a magazine. [...] That way, she would feel better, instead of feeling so hungry! Millie wasn't sure if it would work this time, but the magazine's tips made her want to give it another try!
Question	What did Millie think after reading the magazine?
Answer Candidates	**A. She wasn't sure it would work, but she wanted to give it another try.**
	B. She felt really hungry.
	C. She wanted to go to the library.
	D. She wondered if there was anything she could do to feel better without eating meat.
Example 2	
Text	[...] But at home, her mother had a surprise for her. She told Mary to close her eyes. She thought it might be a toy. When she opened them again, she saw that her mother had baked her something. Her mother opened the oven. Inside was a cake. She jumped up in excitement. [...]
Question	What did Mary's mother surprise her with?
Answer Candidates	**A. A cake she had baked.**
	B. A book.
	C. Candy.
	D. A toy.
Example 3	
Text	It was almost Kira's birthday and Halloween. Kira wanted to paint pumpkins at the circus, but she was sick in bed. [...] Because she was not being careful enough, she was stuck in bed, eating crackers and watching television. [...]
Question	What was Kira doing when she was stuck in bed?
Answer Candidates	**A. Eating crackers and watching TV.**
	B. Drawing.
	C. Watching a movie.
	D. Writing and watching TV.

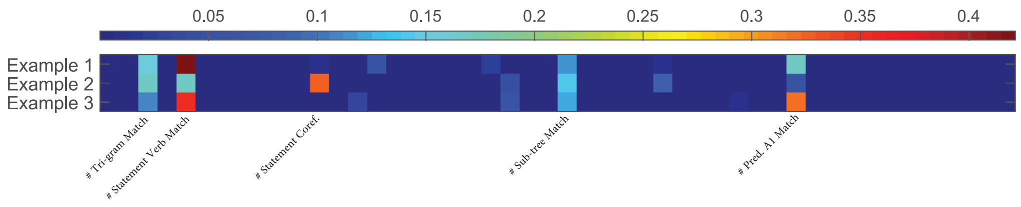


Fig. 11. Attention visualization of the three selected questions. The blue color indicates lower attention values while the red color indicates larger attention value.

In addition, we analyzed error cases made by QFAReader and found that these errors are mainly caused by the following three reasons.

Statement Construction Errors: In QFAReader, the statements are constructed according to the questions and the answer candidates. These statements are used to measure the correctness

Table 13. Example Illustration That Common Sense Is Required to Solve the Questions, and B Is Correct

Text	[...] Stewart laid on his belly and began reading his newspaper. As he was enjoying the sun and the newspaper, he saw a bug crawl across his blanket. [...] When Stewart woke up, he looked for the bug but could not find him. Then, he saw something flying around his house. "Thank you for letting me sleep here last night," said the butterfly. It was the bug! [...]
Question	What was the bug that Stewart found at the beach?
Answer Candidates	A. beetle
	B. caterpillar
	C. ant
	D. butterfly

of the answer candidate. Therefore, the quality of the statements is a vital factor to influence the performance. In this article, the rule-based method is employed to construct the statements,[21] so there are many grammatical errors deteriorating the quality of the extracted features. For example, given the question "*Whose idea is it to use the spoon to dig around the rock?*" and one of its answer candidate "*Billy's,*" the expected statement is "*It is Billy's idea to use the spoon to dig around the rock,*" and according to this correct statement, it is easy to extract the dependency from the word "idea" to "Billy," which is an important clue to indicate the answer correctness. However, according to the pre-defined rules, the statement "*Whose idea is it to use the spoon to dig around the rock Billy's.*" is generated. In this incorrect statement, there is quite a long dependency path between these two words "idea" and "Billy," so the extracted dependency features are unreliable to infer the correct answer. To solve the problem, a more reliable statement generation method is required.

Common Sense Requirement: The texts of MCTest are factional children's stories, so that the texts themselves rather than external knowledge play the pivot role in solving the questions. But sometimes common sense is still required for some questions. As the example illustrated in Table 13, answer B. "caterpillar" is correct, but it does not appear in the text. Without knowing the common sense that butterflies are change from caterpillars, the question can hardly be correctly answered. In the future, information in the knowledge base, such as Freebase, should be involved to solve the problem.

Lacking of Reasoning Ability: The proposed QFAReader measures the correctness of answer candidates according to the extracted features. Even the feature attention is able to strengthen the feature utilization, the reasoning ability is still lacking similar to other feature-based methods. Table 14 illustrates an example to demonstrate the requirement of reasoning ability. Simple NLP-based features can hardly match the correct answer "two" as most of these features are measuring the similarity between the statement and the text. To solve this problem, more complex features or deep neural models with reasoning ability are needed.

We counted the number of error cases caused by these three errors, respectively. According to the statistic on MC500 dataset, we find that these three kinds of errors cover most cases. In particular, about 28% of errors are caused by statement construction, 25% of errors are caused by requiring common sense, about 26% of errors are caused by lacking of reasoning ability, and about 20% of errors are caused by other reasons such as feature extraction errors and coreference errors.

[21] Actually, these statements are provided along with the dataset.

Table 14. Example Illustration That Inference and Reasoning Is Required to Solve the Questions, and A Is the Correct Answer

Text	[...] The cow put his chin in the window. Then the cow put a rock in a window. The cow was done putting things in the window. [...]
Question	How many things did the cow put in a window?
Answer Candidates	**A. two**
	B. three things
	C. five rocks
	D. nothing

5 APPLICATION TO CQA ANSWER SELECTION

In this section, we applied the proposed QFAReader to a real information retrieval task, i.e., cQA answer selection, to show its applicability. QFAReader is only fit for selection-type questions, which means that answer candidates are required, so community-based question answering sites, e.g., Yahoo! Answers, StackExchange, and Quora, are the best-suited data sources. The task of cQA answer selection aims at finding the correct/best answer given a cQA question and its corresponding answer list. Most previous efforts on this task focus on exploring effective features, such as similarity features [49], user features [43], and social features [33]. These manually extracted features represent the quality of answers in different aspects. We assumes that the effectiveness of these features varies a lot with respect to different questions and the proposed QFA is able to distinguish the feature importance according to the given question.

To demonstrate the effectiveness of QFAReader in this task, we randomly crawled 26,752 questions from a cQA dataset, StackExchange. Questions with less than two answers or with less than five user votes are removed. Finally, we obtained 20,278 questions and their corresponding 82,260 answers.[22] These questions are equally divided into five distinct folds for five-fold cross validation, i.e., fourfolds of them are used for training, and the other one fold is used for testing. We extracted more than 200-dimensional features from answers, answerers, and QA pairs to evaluate the quality of answers. These features include, but are not limited to, # words, # characters, # sentences, max sentence length, % capital words, answer sentence embedding, # user tags, # user questions, # user answers, # user best answers, # user edits, overlapping, time span, BM25 similarity, embedding similarity, max matched sequences. It is worth noting that all these features have been explored in previous work of answer selection or answer ranking [12, 13, 16, 49, 71].

Following the experimental setting of Reference [16], we treated the answer with the largest number of votes as the best one, and our purpose is to rank it at a higher position in the answer list. So, we utilize Precision@1 (P@1) and Mean Reciprocal Rank (MRR) [58]. P@1 reports the ratio of questions rank the best answer at the highest position. MRR evaluates the exact rank position of the best answer, and it is calculated as

$$MRR = \frac{1}{|Q|} \sum_{q \in Q} \frac{1}{r_q}, \tag{9}$$

where $|Q|$ denotes the number questions in the test set, and r_q denotes the calculated rank position of the best answer. In our experiment, questions are divided into five disjoint sets equally for five-fold cross validation, where four sets are used for training and the other one is used for testing.

[22]The dataset is available at https://datapublication.wixsite.com/qfareader.

Table 15. Performance Comparison among Different Baselines in cQA Answer Selection Task

Methods	P@1	p-value	MRR	p-value
LR(point-wise)	0.5342	1.7e-3	0.7363	2.4e-3
RankNet(pair-wise)	0.5387	3.7e-3	0.7368	1.9e-3
RankSVM(pair-wise)	0.5545	5.7e-5	0.7402	6.2e-5
ListNet(list-wise)	0.5586	1.3e-4	0.7370	8.3e-5
AdaRank(list-wise)	0.5740	5.7e-4	0.7601	4.2e-3
MART(list-wise)	0.5926	7.2e-4	0.7687	1.8e-3
Without QFA	0.5670	3.1e-5	0.7573	6.2e-5
QFAReader	**0.5992**	—	**0.7704**	—

We compared the QFAReader with state-of-the-art learning to rank methods, include point-wise method (Logistic Regression [43]), pair-wise methods (RankNet [7] and RankSVM [23]), and list-wise methods (ListNet [9], AdaRank [62], and MART [19]). RankSVM is implemented with the help of SVMRank,[23] and other learning to rank methods are implemented with the help of RankLib.[24] In addition, we make a comparison with Without QFA, which removes QFA from QFAReader as described in Section 4.4.

Experimental results are displayed in Table 15. From the table, we have following observations: (1) Compared with existing learning to rank based methods, QFAReader achieves an outstanding performance in both P@1 and MRR metrics. This demonstrates the applicability of the proposed QFAReader in answer selection task. (2) When QFA is removed from QFAReader, the performance of Without QFA decreases and achieves a similar performance compared with other baselines. This indicates that question contents are helpful to distinguish feature importance in answer selection task, and the proposed QFA is effective to learn a reliable feature attention to weight the extracted engineer features. (3) We conducted the significance test with a paired two-sided t-test on both P@1 and MRR metrics. A small p-value indicates a significant improvement of QFAReader. The result shows that the p-values are much smaller than 0.01. This indicates that the performance improvement of QFAReader is statistically significant.

6 CONCLUSION AND FEATURE WORK

This article presents a novel method to select correct answers to multiple-choice questions according to the given unstructured text. It first transforms each question-answer candidate pair into a statement and extracts a rich set of features based on each sentence-statement pair to measure the correctness of the statement. These extracted surface features are transformed into a latent space with a fully connected layer to extract high-level features. Then they are merged together with a cross-sentence max pooling layer, so that the correct answer can be inferred according multiple sentences. Different from traditional feature-based reading comprehension model, a novel feature attention is learned based on the given questions to emphasize important features. Extensive experiments on MCTest demonstrated the effectiveness of the proposed model and the feature attention. Furthermore, we successfully applied the proposed method to select best answer for cQA sites. This demonstrates the applicability of the proposed method.

[23]SVMRank is available at http://www.cs.cornell.edu/people/tj/svm_light/svm_rank.html.
[24]RankLib is available at https://sourceforge.net/p/lemur/wiki/RankLib/.

In the future, we plan to strengthen our model from the following aspects: (1) We will explore more reliable statement generation methods to reduce the noise of features. (2) External knowledge, such as Freebase, Wikipedia, and Yago, will be explored to enhance the performance. (3) Beyond multiple-choice questions, we will try to solve cloze-test questions with deep models. (4) The dataset used in this article is relatively small, so more complex models such as LSTM and memory networks are not fully explored. In the future, the proposed method can be extended by generating feature attentions with other advanced models.

REFERENCES

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv* 1409.0473 (2014), 1–15.

[2] Jun-Wei Bao, Nan Duan, Ming Zhou, and Tiejun Zhao. 2014. Knowledge-based question answering as machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics.* ACL, 967–976.

[3] Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing.* ACL, 1533–1544.

[4] Jiang Bian, Yandong Liu, Eugene Agichtein, and Hongyuan Zha. 2008. Finding the right facts in the crowd: Factoid question answering over social media. In *Proceedings of the 17th International Conference on World Wide Web.* ACM, 467–476.

[5] Antoine Bordes, Sumit Chopra, and Jason Weston. 2014. Question answering with subgraph embeddings. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing.* ACL, 615–620.

[6] Antoine Bordes, Jason Weston, and Nicolas Usunier. 2014. Open question answering with weakly supervised embedding models. In *Proceedings of the 2014 European Conference on Machine Learning and Knowledge Discovery in Databases.* Springer, 165–180.

[7] Christopher J. C. Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Gregory N. Hullender. 2005. Learning to rank using gradient descent. In *Machine Learning, Proceedings of the 22nd International Conference.* 89–96.

[8] Xin Cao, Gao Cong, Bin Cui, Christian S. Jensen, and Quan Yuan. 2012. Approaches to exploring category information for question retrieval in community question-answer archives. *ACM Transactions on Information Systems* 30, 2 (2012), 7:1–7:38.

[9] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. 2007. Learning to rank: From pairwise approach to listwise approach. In *Machine Learning, Proceedings of the 24th International Conference.* 129–136.

[10] Danqi Chen, Jason Bolton, and Christopher D. Manning. 2016. A thorough examination of the CNN/daily mail reading comprehension task. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics.* ACL, 2358–2367.

[11] M. Corbetta and G. L. Shulman. 2002. Control of goal-directed and stimulus-driven attention in the brain. *Nature Rev. Neurosci.* 3, 3 (2002), 201.

[12] Denzil Correa and Ashish Sureka. 2014. Chaff from the wheat: Characterization and modeling of deleted questions on stack overflow. In *Proceedings of the 23rd International World Wide Web Conference.* 631–642.

[13] Silviu Cucerzan and Eugene Agichtein. 2005. Factoid question answering over unstructured and structured web content. In *Proceedings of the 14th Text REtrieval Conference.* NIST, 1–7.

[14] Qing Cui, Bin Gao, Jiang Bian, Siyu Qiu, Hanjun Dai, and Tie-Yan Liu. 2015. KNET: A general framework for learning word embedding using morphological knowledge. *ACM Trans. Info. Syst.* 34, 1 (2015), 4:1–4:25.

[15] Yiming Cui, Ting Liu, Zhipeng Chen, Shijin Wang, and Guoping Hu. 2016. Consensus attention-based neural networks for chinese reading comprehension. In *Proceedings of the 26th International Conference on Computational Linguistics, Proceedings of the Conference.* NIPS, 1777–1786.

[16] Daniel Hasan Dalip, Marcos André Gonçalves, Marco Cristo, and Pável Calado. 2013. Exploiting user feedback to learn to rank answers in q&a forums: A case study with stack overflow. In *The 36th International ACM SIGIR Conference on Research and Development in Information Retrieval.* ACM, 543–552.

[17] Bhuwan Dhingra, Hanxiao Liu, Zhilin Yang, William W. Cohen, and Ruslan Salakhutdinov. 2017. Gated-attention readers for text comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics.* 1832–1846.

[18] Li Dong, Furu Wei, Ming Zhou, and Ke Xu. 2015. Question answering over freebase with multi-column convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics.* 260–269.

[19] Jerome H. Friedman. 2001. Greedy function approximation: A gradient boosting machine. *Ann. Stat.* 29, 5 (2001), 1189–1232.

[20] Jun Hatori, Takuya Matsuzaki, Yusuke Miyao, and Jun'ichi Tsujii. 2012. Incremental joint approach to word segmentation, POS tagging, and dependency parsing in chinese. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics.* ACL, 1045–1053.

[21] Karl Moritz Hermann, Tomás Kociský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Adv. Neural Info. Process. Syst. 28.* NIPS, 1693–1701.

[22] Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. 2015. The goldilocks principle: Reading children's books with explicit memory representations. *arXiv* 1511.02301 (2015), 1–13.

[23] Thorsten Joachims. 2002. Optimizing search engines using clickthrough data. In *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.* 133–142.

[24] Rudolf Kadlec, Martin Schmid, Ondrej Bajgar, and Jan Kleindienst. 2016. Text understanding with the attention sum reader network. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics.* ACL, 908–918.

[25] Cody C. T. Kwok, Oren Etzioni, and Daniel S. Weld. 2001. Scaling question answering to the web. *ACM Trans. Info. Syst.* 19, 3 (2001), 242–262.

[26] Dmitry Lagun and Eugene Agichtein. 2014. Effects of task and domain on searcher attention. In *Proceedings of the 37th International ACM SIGIR Conference on Research and Development in Information Retrieval.* ACM, 1087–1090.

[27] Dmitry Lagun and Eugene Agichtein. 2015. Inferring searcher attention by jointly modeling user interactions and content salience. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval.* ACM, 483–492.

[28] Jimmy J. Lin. 2007. An exploration of the principles underlying redundancy-based factoid question answering. *ACM Trans. Info. Syst.* 25, 2 (2007), 6.

[29] Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing.* ACL, 1412–1421.

[30] Chen Lyu, Yue Zhang, and Donghong Ji. 2016. Joint word segmentation, POS-tagging and syntactic chunking. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence.* AAAI, 3007–3014.

[31] Elman Mansimov, Emilio Parisotto, Lei Jimmy Ba, and Ruslan Salakhutdinov. 2015. Generating images from captions with attention. *arXiv* 1511.02793 (2015).

[32] Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Adv. Neural Info. Process. Syst. 26.* ACM, 3111–3119.

[33] Piero Molino, Luca Maria Aiello, and Pasquale Lops. 2016. Social question answering: Textual, user, and network features for best answer prediction. *ACM Trans. Info. Syst.* 35, 1 (2016), 4:1–4:40.

[34] Karthik Narasimhan and Regina Barzilay. 2015. Machine comprehension with discourse relations. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics.* ACL, 1253–1262.

[35] Liqiang Nie, Meng Wang, Zheng-Jun Zha, Guangda Li, and Tat-Seng Chua. 2011. Multimedia answering: Enriching text QA with media information. In *Proceeding of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval.* ACM, 695–704.

[36] Liqiang Nie, Xiaochi Wei, Dongxiang Zhang, Xiang Wang, Zhipeng Gao, and Yi Yang. 2017. Data-driven answer selection in community QA systems. *IEEE Trans. Knowl. Data Eng.* 29, 6 (2017), 1186–1198.

[37] Sinno Jialin Pan, Zhiqiang Toh, and Jian Su. 2013. Transfer joint embedding for cross-domain named entity recognition. *ACM Trans. Info. Syst.* 31, 2 (2013), 7.

[38] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing.* ACL, 1532–1543.

[39] Hadas Raviv, Oren Kurland, and David Carmel. 2016. Document retrieval using entity-based language models. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval.* ACM, 65–74.

[40] Ronald A. Rensink. 2000. The dynamic representation of scenes. *Visual Cogn.* 7, 1–3 (2000), 17–42.

[41] Matthew Richardson, Christopher J. C. Burges, and Erin Renshaw. 2013. MCTest: A challenge dataset for the open-domain machine comprehension of text. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing.* ACL, 193–203.

[42] Mrinmaya Sachan, Kumar Avinava Dubey, Eric P. Xing, and Matthew Richardson. 2015. Learning answer-entailing structures for machine comprehension. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics.* ACL, 239–249.

[43] Chirag Shah and Jeffrey Pomerantz. 2010. Evaluating and predicting answer quality in community QA. In *Proceeding of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval.* ACM, 411–418.

[44] Yikang Shen, Wenge Rong, Zhiwei Sun, Yuanxin Ouyang, and Zhang Xiong. 2015. Question/answer matching for CQA system via combining lexical and sequential information. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence.* AAAI, 275–281.

[45] Kevin J. Shih, Saurabh Singh, and Derek Hoiem. 2016. Where to look: Focus regions for visual question answering. In *Proceedings of 2016 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 4613–4621.

[46] Ellery Smith, Nicola Greco, Matko Bosnjak, and Andreas Vlachos. 2015. A strong lexical matching method for the machine comprehension test. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. ACL, 1693–1698.

[47] Mark D. Smucker, Xiaoyu Sunny Guo, and Andrew Toulis. 2014. Mouse movement during relevance judging: Implications for determining user attention. In *The 37th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 979–982.

[48] Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.* 15, 1 (2014), 1929–1958.

[49] Mihai Surdeanu, Massimiliano Ciaramita, and Hugo Zaragoza. 2008. Learning to rank answers on large online QA collections. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics*. ACL, 719–727.

[50] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Adv. Neural Info. Process. Syst. 27*. NIPS, 3104–3112.

[51] Adam Trischler, Zheng Ye, Xingdi Yuan, Jing He, and Philip Bachman. 2016. A parallel-hierarchical model for machine comprehension on sparse data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*. ACL, 432–442.

[52] Ferhan Türe and Oliver Jojic. 2016. Ask your TV: Real-time question answering with recurrent neural networks. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 457–458.

[53] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Adv. Neural Info. Process. Syst. 28*. NIPS, 2692–2700.

[54] Ellen M. Voorhees. 1999. The TREC-8 question answering track report. In *Proceedings of The 8th Text REtrieval Conference*. NIST, 1–6.

[55] Bingning Wang, Shangmin Guo, Kang Liu, Shizhu He, and Jun Zhao. 2016. Employing external rich knowledge for machine comprehension. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence*. AAAI, 2929–2925.

[56] Hai Wang, Mohit Bansal, Kevin Gimpel, and David A. McAllester. 2015. Machine comprehension with syntax, frames, and semantics. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*. ACL, 700–706.

[57] Linlin Wang, Zhu Cao, Gerard de Melo, and Zhiyuan Liu. 2016. Relation classification via multi-level attention CNNs. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*. ACL, 1298–1307.

[58] Xin-Jing Wang, Xudong Tu, Dan Feng, and Lei Zhang. 2009. Ranking community answers by modeling question-answer relationships via analogical reasoning. In *Proceedings of the 32nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. 179–186.

[59] Zhiguo Wang and Nianwen Xue. 2014. Joint POS tagging and transition-based constituent parsing in chinese with non-local features. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*. ACL, 733–742.

[60] Xiaochi Wei, Heyan Huang, Liqiang Nie, Hanwang Zhang, Xianling Mao, and Tat-Seng Chua. 2017. I know what you want to express: Sentence element inference by incorporating external knowledge base. *IEEE Trans. Knowl. Data Eng.* 29, 2 (2017), 344–358.

[61] Jason Weston, Sumit Chopra, and Antoine Bordes. 2014. Memory networks. *arXiv* 1410.3916 (2014), 1–15.

[62] Jun Xu and Hang Li. 2007. AdaRank: A boosting algorithm for information retrieval. In *SIGIR 2007: Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. 391–398.

[63] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C. Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of the 32nd International Conference on Machine Learning*. IMLS, 2048–2057.

[64] Hui Yang, Tat-Seng Chua, Shuguang Wang, and Chun-Keat Koh. 2003. Structured use of external knowledge for event-based open domain question answering. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'03)*. ACM, 33–40.

[65] Min-Chul Yang, Nan Duan, Ming Zhou, and Hae-Chang Rim. 2014. Joint relational embeddings for knowledge-based question answering. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*. ACL, 645–650.

[66] Zichao Yang, Xiaodong He, Jianfeng Gao, Li Deng, and Alexander J. Smola. 2016. Stacked attention networks for image question answering. In *Proceedings of 2016 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 21–29.

[67] Zhilin Yang, Junjie Hu, Ruslan Salakhutdinov, and William W. Cohen. 2017. Semi-supervised QA with generative domain-adaptive nets. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*. 1040–1050.

[68] Xuchen Yao and Benjamin Van Durme. 2014. Information extraction over structured data: Question answering with freebase. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*. 956–966.

[69] Jun Yin, Xin Jiang, Zhengdong Lu, Lifeng Shang, Hang Li, and Xiaoming Li. 2016. Neural generative question answering. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence*. AAAI, 2972–2978.

[70] Wenpeng Yin, Sebastian Ebert, and Hinrich Schütze. 2016. Attention-based convolutional neural network for machine comprehension. *arXiv* 1602.04341 (2016), 1–7.

[71] Zhi-Min Zhou, Man Lan, Zheng-Yu Niu, and Yue Lu. 2012. Exploiting user profile information for answer ranking in cQA. In *Proceedings of the 21st International Conference on World Wide Web*. ACM, 767–774.

[72] Stefan Zwicklbauer, Christin Seifert, and Michael Granitzer. 2016. Robust and collective entity disambiguation through semantic embeddings. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 425–434.

Received July 2017; revised March 2018; accepted June 2018