Learning to Attend On Essential Terms: An Enhanced Retriever-Reader Model for Open-domain Question Answering

Jianmo Ni[1]*, **Chenguang Zhu**[2], **Weizhu Chen**[3], **Julian McAuley**[1]
[1] University of California, San Diego
[2] Microsoft Speech and Dialogue Research Group
[3] Microsoft Dynamics 365 AI
{jin018,jmcauley}@ucsd.edu, {chezhu,wzchen}@microsoft.com

Abstract

Open-domain question answering remains a challenging task as it requires models that are capable of understanding questions and answers, collecting useful information, and reasoning over evidence. Previous work typically formulates this task as a reading comprehension or entailment problem given evidence retrieved from search engines. However, existing techniques struggle to retrieve indirectly related evidence when no directly related evidence is provided, especially for complex questions where it is hard to parse precisely what the question asks. In this paper we propose a retriever-reader model that learns to attend on essential terms during the question answering process. We build (1) an essential term selector which first identifies the most important words in a question, then reformulates the query and searches for related evidence; and (2) an enhanced reader that distinguishes between essential terms and distracting words to predict the answer. We evaluate our model on multiple open-domain multiple-choice QA datasets, notably performing at the level of the state-of-the-art on the AI2 Reasoning Challenge (ARC) dataset.

1 Introduction

Open-domain question answering (QA) has been extensively studied in recent years. Many existing works have followed the 'search-and-answer' strategy and achieved strong performance (Chen et al., 2017; Kwon et al., 2018; Wang et al., 2018b) spanning multiple QA datasets such as TriviaQA (Joshi et al., 2017), SQuAD (Rajpurkar et al., 2016), MS-Macro (Nguyen et al., 2016), ARC (Clark et al., 2018) among others.

However, open-domain QA tasks become inherently more difficult when (1) dealing with questions with little available evidence; (2) solving questions where the answer type is free-form text (e.g. multiple-choice) rather than a span among existing passages (i.e., 'answer span'); or when (3) the need arises to understand long and complex questions and reason over multiple passages, rather than simple text matching. As a result, it is essential to incorporate commonsense knowledge or to improve retrieval capability to better capture partially related evidence (Chen et al., 2017).

As shown in Table 1, the TriviaQA, SQuAD, and MS-Macro datasets all provide passages within which the correct answer is guaranteed to exist. However, this assumption ignores the difficulty of retrieving question-related evidence from a large volume of open-domain resources, especially when considering complex questions which require reasoning or commonsense knowledge. On the other hand, ARC does not provide passages known to contain the correct answer. Instead, the task of identifying relevant passages is left to the solver. However, questions in ARC have multiple answer choices that provide indirect information that can help solve the question. As such an effective model needs to account for relations among passages, questions, and answer choices. Real-world datasets such as Amazon-QA (a corpus of user queries from Amazon) (McAuley and Yang, 2016) also exhibit the same challenge, i.e., the need to surface related evidence from which to extract or summarize an answer.

Figure 1 shows an example of a question in the ARC dataset and demonstrates the difficulties in retrieval and reading comprehension. As shown for Choice 1 (C1), a simple concatenation of the

*Most of the work was done during internship at Microsoft, Redmond.

[1]For SQuAD and TriviaQA, since the questions are paired with span-type answers, it is convenient to obtain ranking supervision where retrieved passages are relevant via distant supervision; however free-form questions in ARC and Amazon-QA result in a lack of supervision which makes the problem more difficult. For MS-Macro, the dataset is designed to annotate relevant passages though it has free-form answers.

Dataset	# of questions	Open-domain	Multiple choice	Passage retrieval	No ranking supervision[1]
ARC (Clark et al., 2018)	≈ 7K	✓	✓	✓	✓
Amazon-QA (McAuley and Yang, 2016)	≈ 1.48M	✓			✓
SQuAD (Rajpurkar et al., 2016)	≈ 100K	✓			
TriviaQA (Joshi et al., 2017)	≈ 650K	✓			
MS-Macro (Nguyen et al., 2016)	≈ 1M	✓			

Table 1: Differences among popular QA datasets.

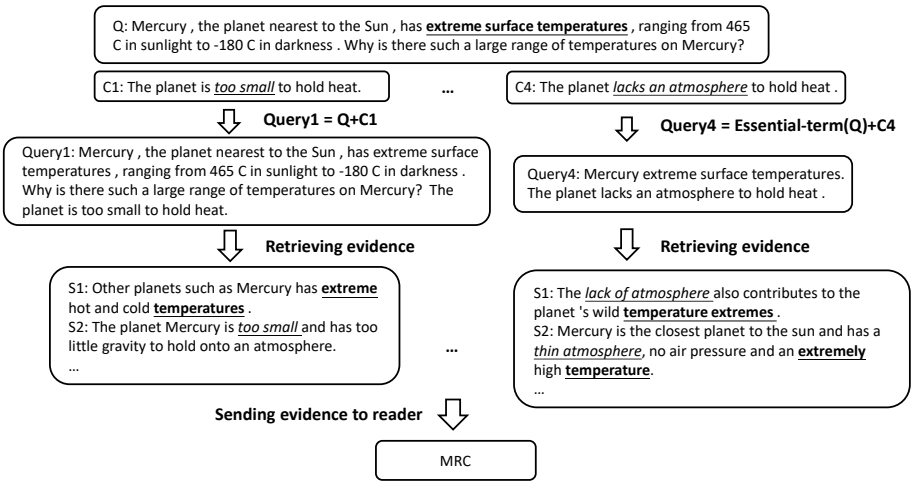

Figure 1: Example of the retrieve-and-read process to solve open-domain questions. Words related with the question are in **bold**; and words related with C1 and C4 are in *italics*.

question and the answer choice is not a reliable query and is of little help when trying to find supporting evidence to answer the question (e.g. we might retrieve sentences similar to the question or the answer choice, but would struggle to find evidence explaining *why* the answer choice is correct). On the other hand, a reformulated query consisting of essential terms in the question and Choice 4 can help retrieve evidence explaining why Choice 4 is a correct answer. To achieve this, the model needs to (1) ensure that the retrieved evidence supports the fact mentioned in both the question and the answer choices and (2) capture this information and predict the correct answer.

To address these difficulties, we propose an essential-term-aware Retriever-Reader (ET-RR) model that learns to attend on essential terms during retrieval and reading. Specifically, we develop a two-stage method with an essential term selector followed by an attention-enhanced reader.

Essential term selector. ET-Net is a recurrent neural network that seeks to understand the question and select essential terms, i.e., key words, from the question. We frame this problem as a classification task for each word in the question. These essential terms are then concatenated with each answer choice and fed into a retrieval engine to obtain related evidence.

Attention-Enhanced Reader. Our neural reader takes the triples (question, answer choice, retrieved passage) as input. The reader consists of a sequence of language understanding layers: an input layer, attention layer, sequence modeling layer, fusion layer, and an output layer. The attention and fusion layers help the model to obtain a refined representation of one text sequence based on the understanding of another, e.g. a passage representation based on an understanding of the question. We further add a choice-interaction module to handle the semantic relations and differences between answer choices. Experiments show that this can further improve the model's accuracy.

We evaluate our model on the ARC Challenge dataset, where our model achieves an accuracy of 36.61% on the test set, and outperformed all leaderboard solutions at the time of writing (Sep. 2018). To compare with other benchmark datasets, we adapt RACE (Lai et al., 2017) and MCScript (Ostermann et al., 2018) to the open domain setting by removing their super-

vision in the form of relevant passages. We also consider a large-scale real-world open-domain dataset, Amazon-QA, to evaluate our model's scalability and to compare against standard benchmarks designed for the open-domain setting. Experiments on these three datasets show that ET-RR outperforms baseline models by a large margin. We conduct multiple ablation studies to show the effectiveness of each component of our model. Finally, we perform in-depth error analysis to explore the model's limitations.

2 Related Work

There has recently been growing interest in building better retrievers for open-domain QA. Wang et al. (2018b) proposed a Reinforced Ranker-Reader model that ranks retrieved evidence and assigns different weights to evidence prior to processing by the reader. Min et al. (2018) demonstrated that for several popular MRC datasets (e.g. SQuAD, TriviaQA) most questions can be answered using only a few sentences rather than the entire document. Motivated by this observation, they built a sentence selector to gather this potential evidence for use by the reader model. Nishida et al. (2018) developed a multi-task learning (MTL) method for a retriever and reader in order to obtain a strong retriever that considers certain passages including the answer text as positive samples during training. The proposed MTL framework is still limited to scenarios where it is feasible to discover whether the passages contain the answer span. Although these works have achieved progress on open-domain QA by improving the ranking or selection of given evidence, few have focused on the scenario where the model needs to start by searching for the evidence itself.

Scientific Question Answering (SQA) is a representative open-domain task that requires capability in both retrieval and reading comprehension. In this paper, we study question answering on the AI2 Reasoning Challenge (ARC) scientific QA dataset (Clark et al., 2018). This dataset contains multiple-choice scientific questions from 3rd to 9th grade standardized tests and a large corpus of relevant information gathered from search engines. The dataset is partitioned into "Challenge" and "Easy" sets. The challenge set consists of questions that cannot be answered correctly by either of the solvers based on Pointwise Mutual Information (PMI) or Information Retrieval (IR). Existing models tend to achieve only slightly better and sometimes even worse performance than random guessing, which shows that existing models are not well suited to this kind of QA task.

Jansen et al. (2017) first developed a rule-based focus word extractor to identify essential terms in the question and answer candidates. The extracted terms are used to aggregate a list of potential answer justifications for each answer candidate. Experiments shown that focus words are beneficial for SQA on a subset of the ARC dataset. Khashabi et al. (2017) also worked on the problem of finding essential terms in a question for solving SQA problems. They published a dataset containing over 2,200 science questions annotated with essential terms and train multiple classifiers on it. Similarly, we leverage this dataset to build an essential term selector using a neural network-based algorithm. More recently, Boratko et al. (2018) developed a labeling interface to obtain high quality labels for the ARC dataset. One finding is that human annotators tend to retrieve better evidence after they reformulate the search queries which are originally constructed by a simple concatenation of question and answer choice. By feeding the evidence obtained by human-reformulated queries into a pre-trained MRC model (i.e., DrQA (Chen et al., 2017)) they achieved an accuracy increase of 42% on a subset of 47 questions. This shows the potential for a "human-like" retriever to boost performance on this task.

Query reformulation has been shown to be effective in information retrieval (Lavrenko and Croft, 2001). Nogueira and Cho (2017) modeled the query reformulation task as a binary term selection problem (i.e., whether to choose the term in the original query and the documents retrieved using the original query). The selected terms are then concatenated to form the new query. Instead of choosing relevant words, Buck et al. (2018) proposed a sequence-to-sequence model to generate new queries. Das et al. (2019) proposed Multi-step Retriever-Reader which explores an iterative retrieve-and-read strategy for open-domain question answering. It formulates the query reformulation problem in the embedding space where the vector representation of the question is changed to improve the performance. Since there is no supervision for training the query reformulator, all these methods using reinforcement learning to maximize the task-specific metrics (e.g. Recall for para-

graph ranking, F1 and Exact Matching for span-based MRC). Different from these works, we train the query reformulator using an annotated dataset as supervision and then apply the output to a separate reader model. We leave the exploration of training our model end-to-end using reinforcement learning as future work.

3 Approach

In this section, we introduce the essential-term-aware retriever-reader model (ET-RR). As shown in Figure 2, we build a term selector to discover which terms are essential in a question. The selected terms are then used to formulate a more efficient query enabling the retriever to obtain related evidence. The retrieved evidence is then fed to the reader to predict the final answer.

For a question with q words $\mathbf{Q} = \{w_t^Q\}_{t=1}^q$ along with its N answer choices $\mathbf{C} = \{\mathbf{C}_n\}_{n=1}^N$ where $\mathbf{C}_n = \{w_t^C\}_{t=1}^c$, the essential-term selector chooses a subset of essential terms $\mathbf{E} \subset \mathbf{Q}$, which are then concatenated with each $\mathbf{C}_n$ to formulate a query. The query for each answer choice, $\mathbf{E} + \mathbf{C}_n$, is sent to the retriever (e.g. Elastic Search[2]), and the top K retrieved sentences based on the scores returned by the retriever are then concatenated into the evidence passage $\mathbf{P}_n = \{w_t^P\}_{t=1}^K$.

Next, given these text sequences $\mathbf{Q}$, $\mathbf{C}$, and $\mathbf{P} = \{\mathbf{P}_n\}_{n=1}^N$, the reader will determine a matching score for each triple $\{\mathbf{Q}, \mathbf{C}_n, \mathbf{P}_n\}$. The answer choice $\mathbf{C}_{n^*}$ with the highest score is selected.

We first introduce the reader model in Section 3.1 and then the essential term selector in Section 3.2.

3.1 Reader Model

3.1.1 Input Layer

To simplify notation, we ignore the subscript n denoting the answer choice until the final output layer. In the input layer, all text inputs—the question, answer choices, and passages, i.e., retrieved evidence—are converted into embedded representations. Similar to Wang (2018), we consider the following components for each word:

Word Embedding. Pre-trained GloVe word embedding with dimensionality $d_w = 300$.

Part-of-Speech Embedding and Named-Entity Embedding. The part-of-speech tags and named entities for each word are mapped to embeddings with dimension 16.

[2]https://www.elastic.co/products/elasticsearch

Relation Embedding. A relation between each word in $\mathbf{P}$ and any word in $\mathbf{Q}$ or $\mathbf{C}$ is mapped to an embedding with dimension 10. In the case that multiple relations exist, we select one uniformly at random. The relation is obtained by querying ConceptNet (Speer et al., 2017).

Feature Embeddings. Three handcrafted features are used to enhance the word representations: (1) Word Match; if a word or its lemma of $\mathbf{P}$ exists in $\mathbf{Q}$ or $\mathbf{C}$, then this feature is 1 (0 otherwise). (2) Word Frequency; a logarithmic term frequency is calculated for each word. (3) Essential Term; for the i-th word in $\mathbf{Q}$, this feature, denoted as w_{e_i}, is 1 if the word is an essential term (0 otherwise). Let $\mathbf{w}_e = [w_{e_1}, w_{e_2}, \ldots, w_{e_q}]$ denote the essential term vector.

For $\mathbf{Q}, \mathbf{C}, \mathbf{P}$, all of these components are concatenated to obtain the final word representations $\mathbf{W}_Q \in \mathbb{R}^{q \times d_Q}, \mathbf{W}_C \in \mathbb{R}^{c \times d_C}, \mathbf{W}_P \in \mathbb{R}^{p \times d_P}$, where d_Q, d_C, d_P are the final word dimensions of $\mathbf{Q}, \mathbf{C}$, and $\mathbf{P}$.

3.1.2 Attention Layer

As shown in Figure 2, after obtaining word-level embeddings, attention is added to enhance word representations. Given two word embedding sequences $\mathbf{W}_U, \mathbf{W}_V$, word-level attention is calculated as:

$$\begin{aligned} \mathbf{M}'_{UV} &= \mathbf{W}_U\mathbf{U} \cdot (\mathbf{W}_V\mathbf{V})^\top \\ \mathbf{M}_{UV} &= \text{softmax}(\mathbf{M}'_{UV}) \\ \mathbf{W}_U^V &= \mathbf{M}_{UV} \cdot (\mathbf{W}_V\mathbf{V}), \end{aligned} \tag{1}$$

where $\mathbf{U} \in \mathbb{R}^{d_U \times d_w}$ and $\mathbf{V} \in \mathbb{R}^{d_V \times d_w}$ are two matrices that convert word embedding sequences to dimension d_w, and $\mathbf{M}'_{UV}$ contains dot products between each word in $\mathbf{W}_U$ and $\mathbf{W}_V$, and softmax is applied on $\mathbf{M}'_{UV}$ row-wise. Three types of attention are calculated using Equation (1): (1) question-aware passage representation $\mathbf{W}_P^Q \in \mathbb{R}^{p \times d_w}$; (2) question-aware choice representation $\mathbf{W}_C^Q \in \mathbb{R}^{c \times d_w}$; and (3) passage-aware choice representation $\mathbf{W}_C^P \in \mathbb{R}^{c \times d_w}$.

3.1.3 Sequence Modeling Layer

To model the contextual dependency of each text sequence, we use BiLSTMs to process the word representations obtained from the input layer and attention layer:

$$\begin{aligned} \mathbf{H}^q &= \mathbf{BiLSTM}[\mathbf{W}_Q] \\ \mathbf{H}^c &= \mathbf{BiLSTM}[\mathbf{W}_C; \mathbf{W}_C^P; \mathbf{W}_C^Q] \\ \mathbf{H}^p &= \mathbf{BiLSTM}[\mathbf{W}_P; \mathbf{W}_P^Q], \end{aligned} \tag{2}$$

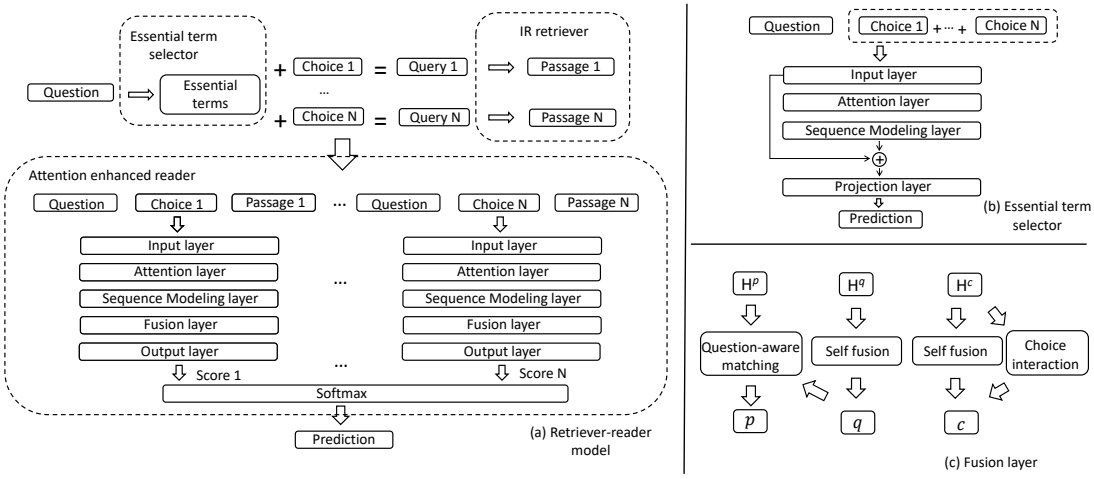

Figure 2: Model structure for our essential-term-aware retriever-reader model.

where $\mathbf{H}^q \in \mathbb{R}^{q\times l}, \mathbf{H}^c \in \mathbb{R}^{c\times l}$, and $\mathbf{H}^p \in \mathbb{R}^{p\times l}$ are the hidden states of the BiLSTMs, ';' is feature-wise concatenation, and l is the size of the hidden states.

3.1.4 Fusion Layer

We further convert each question and answer choice into a single vector: $\mathbf{q} \in \mathbb{R}^l$ and $\mathbf{c} \in \mathbb{R}^l$:

$$\begin{aligned}\boldsymbol{\alpha}_q &= \mathrm{softmax}([\mathbf{H}^q; \mathbf{w}_e] \cdot \mathbf{w}_{sq}^\top),\ \mathbf{q} = \mathbf{H}^{q\top}\boldsymbol{\alpha}_q \\ \boldsymbol{\alpha}_c &= \mathrm{softmax}(\mathbf{H}^c \cdot \mathbf{w}_{sc}^\top),\ \mathbf{c} = \mathbf{H}^{c\top}\boldsymbol{\alpha}_c,\end{aligned} \tag{3}$$

where the essential-term feature $\mathbf{w}_e$ from Section 3.1.1 is concatenated with $\mathbf{H}^q$, and $\mathbf{w}_{sq}$ and $\mathbf{w}_{sc}$ are learned parameters.

Finally, a bilinear sequence matching is calculated between $\mathbf{H}^p$ and $\mathbf{q}$ to obtain a question-aware passage representation, which is used as the final passage representation:

$$\boldsymbol{\alpha}_p = \mathrm{softmax}(\mathbf{H}^p \cdot \mathbf{q}); \quad \mathbf{p} = \mathbf{H}^{p\top}\boldsymbol{\alpha}_p. \tag{4}$$

3.1.5 Choice Interaction

When a QA task provides multiple choices for selection, the relationship between the choices can provide useful information to answer the question. Therefore, we integrate a choice interaction layer to handle the semantic correlation between multiple answer choices. Given the hidden state $\mathbf{H}^{c_n}$ of choice c_n and $\mathbf{H}^{c_i}$ of other choices $c_i, \forall i \neq n$, we calculate the differences between the hidden states and apply max-pooling over the differences:

$$\mathbf{c}_{inter} = \mathrm{Maxpool}(\mathbf{H}^{c_n} - \frac{1}{N-1}\sum_{i\neq n}\mathbf{H}^{c_i}), \tag{5}$$

where N is the total number of answer choices. Here, $\mathbf{c}_{inter}$ characterizes the differences between an answer choice c_n and other answer choices. The final representation of an answer choice is updated by concatenating the self-attentive answer choice vector and inter-choice representation as $\mathbf{c}^{final} = [\mathbf{c}; \mathbf{c}_{inter}]$.

3.1.6 Output Layer

For each tuple $\{\mathbf{q}, \mathbf{p}_n, \mathbf{c}_n\}_{n=1}^N$, two scores are calculated by matching (1) the passage and answer choice and (2) question and answer choice. We use a bilinear form for both matchings. Finally, a softmax function is applied over N answer choices to determine the best answer choice:

$$\begin{aligned}s_n^{pc} = \mathbf{p}_n\mathbf{W}^{pc}\mathbf{c}_n^{final};\quad s_n^{qc} = \mathbf{q}\mathbf{W}^{qc}\mathbf{c}_n^{final} \\ \mathbf{s} = \mathrm{softmax}(\mathbf{s}^{pc}) + \mathrm{softmax}(\mathbf{s}^{qc}),\end{aligned} \tag{6}$$

where s_n^{pc}, s_n^{qc} are the scores for answer choice $1 \leq n \leq N$; $\mathbf{s}^{pc}, \mathbf{s}^{qc}$ are score vectors for all N choices; and $\mathbf{s}$ contains the final scores for each answer choice. During training, we use a cross-entropy loss.

3.2 Essential Term Selector

Essential terms are key words in a question that are crucial in helping a retriever obtain related evidence. Given a question $\mathbf{Q}$ and N answer choices $\mathbf{C}_1, \ldots, \mathbf{C}_N$, the goal is to predict a binary variable y_i for each word Q_i in the question $\mathbf{Q}$, where $y_i = 1$ if Q_i is an essential term and 0 otherwise. To address this problem, we build a neural model, ET-Net, which has the same design as the reader model for the input layer, attention layer, and sequence modeling layer to obtain the hidden state $\mathbf{H}^q$ for question $\mathbf{Q}$.

In detail, we take the question $\mathbf{Q}$ and the concatenation $\mathbf{C}$ of all N answer choices as input to

Question	If an object is attracted to a magnet, the object is most likely made of (A) wood (B) plastic (C) cardboard (D) metal
# annotators	5
Annotation	If,0; an,0; object,3; is,0; attracted,5; to,0; a,0; magnet,,5; the,0; object,1; is,0; most,0; likely,0; made,2; of,0

Table 2: Example of essential term data.

Model	Precision	Recall	F1
MaxPMI	0.88	0.65	0.75
SumPMI	0.88	0.65	0.75
PropSurf	0.68	0.64	0.66
PropLem	0.76	0.64	0.69
ET Classifier	0.91	0.71	0.80
ET-Net	0.74	**0.90**	**0.81**

Table 3: Performance of different selectors.

ET-Net. $\mathbf{Q}$ and $\mathbf{C}$ first go through an input layer to convert to the embedded word representation, and then word-level attention is calculated to obtain a choice-aware question representation $\mathbf{W}_Q^C$ as in Equation (1). We concatenate the word representation and word-level attention representation of the question and feed it into the sequence modeling layer:

$$\mathbf{H}^q = \mathbf{BiLSTM}[\mathbf{W}_Q; \mathbf{W}_Q^C]. \quad (7)$$

As shown in Figure 2, the hidden states obtained from the attention layer are then concatenated with the embedded representations of $\mathbf{Q}$ and fed into a projection layer to obtain the prediction vector $\mathbf{y} \in \mathbb{R}^q$ for all words in the question:

$$\mathbf{y} = [\mathbf{H}^q; \mathbf{W}_Q^f] \cdot \mathbf{w}^s, \quad (8)$$

where $\mathbf{w}^s$ contains the learned parameters, and $\mathbf{W}_Q^f$ is the concatenation of the POS embedding, NER embedding, relation embedding, and feature embedding from Section 3.1.1.

After obtaining the prediction for each word, we use a binary cross-entropy loss to train the model. During evaluation, we take words with y_i greater than 0.5 as essential terms.

4 Experiments

In this section, we first discuss the performance of the essential term selector, ET-Net, on a public dataset. We then discuss the performance of the whole retriever-reader pipeline, ET-RR, on multiple open-domain datasets.For both the ET-Net and ET-RR models, we use 96-dimensional hidden states and 1-layer BiLSTMs in the sequence modeling layer. A dropout rate of 0.4 is applied for the embedding layer and the BiLSTMs' output layer. We use adamax (Kingma and Ba, 2014) with a learning rate of 0.02 and batch size of 32. The model is trained for 100 epochs. Our code is released at `https://github.com/nijianmo/arc-etrr-code`.

4.1 Performance on Essential Term Selection

We use the public dataset from Khashabi et al. (2017) which contains 2,223 annotated questions, each accompanied by four answer choices. Table 2 gives an example of an annotated question. As shown, the dataset is annotated for binary classification. For each word in the question, the data measures whether the word is an "essential" term according to 5 annotators. We then split the dataset into training, development, and test sets using an 8:1:1 ratio and select the model that performs best on the development set.

Table 3 shows the performance of our essential term selector and baseline models from Khashabi et al. (2017). The second best model (ET Classifier) is an SVM-based model from Khashabi et al. (2017) requiring over 100 handcrafted features. As shown, our ET-Net achieves a comparable result with ET Classifier in terms of the F1 Score.

Table 4 shows example predictions made by ET-Net. As shown, ET-Net is capable of selecting most ground-truth essential terms. It rejects certain words such as "organisms" which have a high TF-IDF in the corpus but are not relevant to answering a particular question. This shows its ability to discover essential terms according to the context of the question.

4.2 Performance on Open-domain Multiple-choice QA

We train and evaluate our proposed pipeline method ET-RR on four open-domain multiple-choice QA datasets. All datasets are associated with a sentence-level corpus. Detailed statistics are shown in Table 5.

- ARC (Clark et al., 2018): We consider the 'Challenge' set in the ARC dataset and use the provided corpus during retrieval.
- RACE-Open: We adapted the RACE dataset (Lai et al., 2017) to the open-domain setting. Originally, each question in RACE comes

Example questions
Which **unit** of **measurement** can be used to describe the **length** of a **desk** ?
One way **animal** usually **respond** to a sudden **drop** in **temperature** is by
Organisms require energy to survive. Which of the following **processes provides energy** to the body ?

Table 4: Examples of essential term prediction (in questions) by ET-Net. True positives are marked bold and underlined while false positives are only underlined.

Dataset	Train	Dev	Test	Corpus
ARC	1,119	299	1,172	1.46M
RACE-Open	9,531	473	528	0.52M
MCScript-Open	1,036	156	319	24.2K
Amazon-Patio	36,587	4,531	4,515	2.55M
Amazon-Auto	49,643	6,205	6,206	7.32M
Amazon-Cell	40,842	5,105	5,106	1.86M

Table 5: Statistics on ARC, RACE-Open, MCScript-Open and Amazon-QA. Corpus size is the number of sentences.

Dataset	Example questions
ARC	The best way to separate salt from water is with the use of Which geologic process most likely caused the formation of the Mount St. Helens Volcano?
RACE-Open	According to the article, what does the band Four Square hope to do in the future? According to the article we know it is _ to prevent the forests from slowly disappearing.
Amazon-QA	For anyone with small ears, do these fit comfortably or do they feel like they are always going to fall out, not in correctly, etc. Does it remove easily and does it leave any sticky residue behind? thanks in advance.

Table 6: Example of predictions on ARC, RACE-Open and Amazon-QA. Predicted terms are underlined.

with a specific passage. To enable passage retrieval, we concatenate all passages into a corpus with sentence deduplication.[3]

- MCScript-Open: The MCScript (Ostermann et al., 2018) dataset is also adapted to the open-domain setting. Again we concatenate all passages to build the corpus.[4]

- Amazon-QA: The Amazon-QA dataset (McAuley and Yang, 2016) is an open-domain QA dataset covering over one million questions across multiple product categories. Each question is associated with a free-form answer. We adapt it into a 2-way multiple-choice setting by randomly sampling an answer from other questions as an answer distractor. We split all product reviews at the sentence-level to build the corpus. We consider three categories from the complete dataset in our experiments.

In the experiments, ET-RR uses ET-Net to choose essential terms in the question. Table 6 shows example predictions on these target datasets. Then it generates a query for each of the N answer choices by concatenating essential terms and the answer choice. For each query, ET-RR obtains the top K sentences returned by the retriever and considers these sentences as a passage for the reader. We set $K = 10$ for all experiments.

We compare ET-RR with existing retrieve-and-read methods on both datasets. As shown in Table 7, on the ARC dataset, ET-RR outperforms all previous models without using pre-trained models and achieves a relative 8.1% improvement over the second best BiLSTM Max-out method (Mihaylov et al., 2018). Recently, finetuning on pre-trained models has shown great improvement over a wide range of NLP tasks. Sun et al. (2019) proposed a 'Reading Strategies' method to finetune the pre-trained model OpenAI GPT, a language model trained on the BookCorpus dataset (Radford, 2018). They trained Reading Strategies on the RACE dataset to obtain more auxiliary knowledge and then finetune that model on the ARC corpus. Table 8 demonstrates the performance comparison of ET-RR and Reading Strategies on ARC. As shown, though Reading Strategies trained on both ARC and RACE dataset outperforms ET-RR, ET-RR outperforms Reading Strategies using only the ARC dataset at training time.

On the RACE-Open and MCScript-Open datasets, ET-RR achieves a relative improvement of 24.6% and 10.5% on the test set compared with the second best method IR solver respectively. We also evaluate on multiple categories of

[3] As short questions might not contain any words which can relate the question to any specific passage or sentence, we only keep questions with more than 15 words.

[4] We keep questions with more than 10 words rather than 15 words to ensure that there is sufficient data.

Model	ARC Test	RACE-Open Test	MCScript-Open Test
IR solver	20.26	30.70	60.46
Random	25.02	25.01	50.02
BiDAF	26.54	26.89	50.81
BiLSTM Max-out	33.87	/	/
ET-RR (Concat)	35.33	36.87	66.46
ET-RR	**36.61**	**38.61**	**67.71**

Table 7: Accuracy on multiple-choice selection on ARC, RACE-Open and MCScript-Open.

Model	Amazon -Patio	Amazon -Auto	Aamzon -Cell
IR solver	72.80	73.60	70.50
Moqa	84.80	86.30	88.60
ET-RR (Concat)	96.19	95.21	93.26
ET-RR	**96.61**	**95.96**	**93.81**

Table 9: Accuracy on multiple-choice selection on three product categoris of Amazon-QA.

Pre-trained	model	RACE
✓	Reading Strategies	63.8
✓	OpenAI GPT	59.0
	ET-RR (reader)	**52.3**
	Bi-attn (MRU)	50.4
	Hier. Co-Matching	50.4

Table 10: Experimental results for reader on RACE.

the Amazon-QA dataset. As shown in Table 9, ET-RR increases the accuracy by 10.33% on average compared to the state-of-the-art model Moqa (McAuley and Yang, 2016). We also compare ET-RR with ET-RR (Concat), which is a variant of our proposed model that concatenates the question and choice as a query for each choice. Among all datasets, ET-RR outperforms ET-RR (concat) consistently which shows the effectiveness of our essential-term-aware retriever.

4.3 Ablation study

We investigate how each component contributes to model performance.

Performance of reader. Our reader alone can be applied on MRC tasks using the given passages. Here, we evaluate our reader on the original RACE dataset to compare with other MRC models as shown in Table 10. As shown, the recently proposed Reading Strategies and OpenAI GPT models, that finetune generative pre-trained models achieve the highest scores. Among non-pre-trained models, our reader outperforms other

Training Corpus	model	ARC
ARC	Reading Strategies	35.0
	ET-RR	36.6
ARC+RACE	Reading Strategies	40.7

Table 8: Comparisons of ET-RR and Reading Strategies on ARC.

Model	Test
ET-RR	**36.61**
– inter-choice	36.36
– passage-choice	35.41
– question-choice	34.47
– passage-question	34.05

Table 11: Ablation test on attention components of ET-RR on ARC. '–' denotes the ablated feature.

baselines: Bi-attn (MRU) (Tay et al., 2018) and Hierarchical Co-Matching (Wang et al., 2018a) by a relative improvement of 3.8%.

Attention components. Table 11 demonstrates how the attention components contribute to the performance of ET-RR. As shown, ET-RR with all attention components performs the best on the ARC test set. The performance of ET-RR without passage-question attention drops the most significantly out of all the components. It is worth noting that the choice interaction layer gives a further 0.24% boost on test accuracy.

Essential term selection. To understand the contribution of our essential-term selector, we compare ET-RR with two variants: (1) ET-RR (Concat) and (2) ET-RR (TF-IDF). For ET-RR (TF-IDF), we calculate the TF-IDF scores and take words with the top 30% of TF-IDF scores in the question to concatenate with each answer choice as a query.[5]

Table 12 shows an ablation study comparing different query formulation methods and amounts of retrieved evidence K. As shown, with the essential term selector ET-Net, the model consistently outperforms other baselines, given different numbers of retrievals K. Performance for all models is best when $K = 10$. Furthermore, only using TF-IDF to select essential terms in a question is not effective. When $K = 10$, the ET-RR (TF-IDF)

[5]According to the annotated dataset, around 30% of the terms in each question are labelled as essential.

Model	ET-RR (Concat)		ET-RR (TF-IDF)		ET-RR	
Top K	Dev	Test	Dev	Test	Dev	Test
5	39.26	33.36	39.93	34.73	39.93	35.59
10	38.93	35.33	39.43	35.24	**43.96**	**36.61**
20	41.28	34.56	38.59	33.88	42.28	35.67

Table 12: Comparison of query formulation methods and amounts of retrieved evidence (i.e., top K) on the ARC dataset, in terms of percentage accuracy.

method performs even worse than ET-RR (Concat). This illustrates the challenges in understanding what is essential in a question.

Though ET-RR consistently outperforms ET-RR (TF-IDF), the improvement is relatively modest on the Test set (around 1.4%). A similar outcome has been reported in Jansen et al. (2017); Khashabi et al. (2017) where essential term extraction methods have shown around 2%-4% gain compared with TF-IDF models and struggle to obtain further improvement on SQA tasks. This consensus might show the discrepancy of essential terms between human and machine (i.e., the essential terms obtained using a human annotated dataset might not be helpful in a machine inference model). Another reason might be the current retrieval method does not effectively use these essential terms and the performance highly depends on the dataset. Note that the ET-RR outperforms ET-RR (TF-IDF) by around 4% on the Dev set. Therefore, how to develop well-formed single or even multi-hop queries using these terms are worth studying in the future.

4.4 Error Analysis

Table 13 shows two major types of error, where the correct answer choice is in **bold** and the predicted answer choice is in *italics*.

Retrieved supporting evidence but failed to reason over it. For the first question, there exists evidence that can justify the answer candidate (C). However, the model chooses (D) which has more words overlapping with its evidence. This shows that the model still lacks the reasoning capability to solve complex questions.

Failed to retrieve supporting evidence. For the second question, the retrieved evidence of both the correct answer (D) and the prediction (B) is not helpful to solve the question. Queries such as 'what determines the year of a planet' are needed to acquire the knowledge for solving this question.

The elements carbon, hydrogen, and oxygen are parts of many different compounds. Which explains why these three elements can make so many different compounds?
(A) They can be solids, liquids, or gases.
(B) They come in different sizes and shapes.
(C) They combine in different numbers and ratios.
* There are many different types of compounds because atoms of elements combine in many different ways (and in different whole number ratios) to form different compounds.
(D) They can be a proton, a neutron, or an electron.
* Atoms of different elements have a different number of protons, neutrons, and electrons.
Which planet in the solar system has the longest year?
(A) The planet closest to the Sun.
(B) The planet with the longest day.
* The planet with the longest day is Venus; a day on Venus takes 243 Earth days.
(C) The planet with the most moons.
(D) The planet farthest from the Sun.
* The last planet discovered in our solar system is farthest away from the sun.

Table 13: Examples where ET-RR fails on ARC. The retrieved evidence for each answer candidate is marked by *.

This poses further challenges to design a retriever that can rewrite such queries.

5 Conclusion

We present a new retriever-reader model (ET-RR) for open-domain QA. Our pipeline has the following contributions: (1) we built an essential term selector (ET-Net) which helps the model understand which words are essential in a question leading to more effective search queries when retrieving related evidence; (2) we developed an attention-enhanced reader with attention and fusion among passages, questions, and candidate answers. Experimental results show that ET-RR outperforms existing QA models on open-domain multiple-choice datasets as ARC Challenge, RACE-Open, MCScript-Open and Amazon-QA. We also perform in-depth error analysis to show the limitations of current work. For future work, we plan to explore the directions of (1) constructing multi-hop query and (2) developing end-to-end retriever-reader model via reinforcement learning.

Acknowledgements. We thank Jade Huang for proofreading the paper, Liang Wang and Daniel Khashabi for sharing code and the annotated dataset with us. This work is partly supported by NSF #1750063. We thank all the reviewers for their constructive suggestions.

References

Michael Boratko, Harshit Padigela, Divyendra Mikkilineni, Pritish Yuvraj, Rajarshi Das, Andrew McCallum, Maria Chang, Achille Fokoue, Pavan Kapanipathi, Nicholas Mattei, Ryan Musa, Kartik Talamadupula, and Michael Witbrock. 2018. A systematic classification of knowledge, reasoning, and context within the arc dataset. In *1st Workshop on Machine Reading for Question Answering*.

Christian Buck, Jannis Bulian, Massimiliano Ciaramita, Andrea Gesmundo, Neil Houlsby, Wojciech Gajewski, and Wei Wang. 2018. Ask the right questions: Active question reformulation with reinforcement learning. In *ICLR*.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading wikipedia to answer open-domain questions. In *ACL*.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *CoRR*, abs/1803.05457.

Rajarshi Das, Shehzaad Dhuliawala, Manzil Zaheer, and Andrew McCallum. 2019. Multi-step retriever-reader interaction for scalable open-domain question answering. In *ICLR*.

Peter Jansen, Rebecca Sharp, Mihai Surdeanu, and Peter Clark. 2017. Framing qa as building and ranking intersentence answer justifications. *Computational Linguistics*, 43:407–449.

Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke S. Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *ACL*.

Daniel Khashabi, Tushar Khot, Ashish Sabharwal, and Dan Roth. 2017. Learning what is essential in questions. In *CoNLL*.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Heeyoung Kwon, Harsh Trivedi, Peter Jansen, Mihai Surdeanu, and Niranjan Balasubramanian. 2018. Controlling information aggregation for complex question answering. In *ECIR*.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard H. Hovy. 2017. Race: Large-scale reading comprehension dataset from examinations. In *EMNLP*.

Victor Lavrenko and W. Bruce Croft. 2001. Relevance-based language models. *SIGIR Forum*, 51:260–267.

Julian McAuley and Alex Yang. 2016. Addressing complex and subjective product-related queries with customer reviews. In *WWW*.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *EMNLP*.

Sewon Min, Victor Zhong, Richard Socher, and Caiming Xiong. 2018. Efficient and robust question answering from minimal context over documents. In *ACL*.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. Ms marco: A human generated machine reading comprehension dataset. *CoRR*, abs/1611.09268.

Kyosuke Nishida, Itsumi Saito, Atsushi Otsuka, Hisako Asano, and Junji Tomita. 2018. Retrieve-and-read: Multi-task learning of information retrieval and reading comprehension. In *CIKM*.

Rodrigo Nogueira and Kyunghyun Cho. 2017. Task-oriented query reformulation with reinforcement learning. In *EMNLP*.

Simon Ostermann, Michael Roth, Ashutosh Modi, Stefan Thater, and Manfred Pinkal. 2018. Semeval-2018 task 11: Machine comprehension using commonsense knowledge. In *SemEval@NAACL-HLT*.

Alec Radford. 2018. Improving language understanding by generative pre-training. In *Technical report*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100, 000+ questions for machine comprehension of text. In *EMNLP*.

Robert Speer, Joshua Chin, and Catherine Havasi. 2017. Conceptnet 5.5: An open multilingual graph of general knowledge. In *AAAI*.

Kai Sun, Dian Yu, Dong Yu, and Claire Cardie. 2019. Improving machine reading comprehension with general reading strategies. In *NAACL*.

Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. 2018. Multi-range reasoning for machine comprehension. *CoRR*, abs/1803.09074.

Liang Wang. 2018. Yuanfudao at semeval-2018 task 11: Three-way attention and relational knowledge for commonsense machine comprehension. In *SemEval@NAACL-HLT*.

Shuohang Wang, Mo Yu, Shiyu Chang, and Jing Jiang. 2018a. A co-matching model for multi-choice reading comprehension. In *ACL*.

Shuohang Wang, Mo Yu, Xiaoxiao Guo, Zhiguo Wang, Tim Klinger, Wei Zhang, Shiyu Chang, Gerald Tesauro, Bowen Zhou, and Jing Jiang. 2018b. R3: Reinforced ranker-reader for open-domain question answering. In *AAAI*.