Unsupervised Open-Domain Question Answering with Higher Answerability

Anonymous ACL submission

Abstract

Open-domain Question Answering (ODQA) has achieved significant results in terms of supervised learning manner. However, data annotation cannot also be irresistible for its huge demand in an open domain. Though unsupervised QA or unsupervised Machine Reading Comprehension (MRC) has been tried more or less, unsupervised ODQA has not been touched according to our best knowledge. This paper thus pioneers the work of unsupervised ODQA by formally introducing the task and proposing a series of key data construction methods. Our exploration in this work inspiringly shows unsupervised ODQA can reach up to 86% performance of supervised ones.

1 Introduction

Open-domain Question Answering (ODQA) is the task of answering questions based on information from a very large collection of documents which has a variety of topics (Chen and Yih, 2020). Unlike Machine Reading Comprehension (MRC) task where a passage containing evidences and answers is provided for each question, ODQA is more challenging as there is no such supporting passage beforehand. ODQA systems need to go through a large collection of passages such as the whole Wikipedia to find the correct answer.

While tremendous progress on ODQA have been made based on pretrained language models such as BERT (Devlin et al., 2019), ELECTRA (Clark et al., 2020), and T5 (Raffel et al., 2020), fine-tuning these language models requires large-scale labeled data, i.e., passage-question-answer triples (Lewis et al., 2019). Apparently, it is costly and practically infeasible to manually create a dataset for every new domain.

Though previous studies which have made attempts in unsupervised MRC like (Lewis et al., 2019; Li et al., 2020; Fabbri et al., 2020; Hong et al., 2020; Perez et al., 2020), as to our best knowledge, no such manner of attempts have been made in terms of ODQA. Thus in this paper, for the first time, we tackle the ODQA setting without human-annotated data, which we term Unsupervised ODQA (UODQA). Concretely, our setting is: starting from an automatically generated question or question-like sentence, we employ a lexical-based retriever like BM25 to retrieve positive passages that contain the answer and negative passages without the answer, from the Wikipedia corpus. Together with these, we can effectively train a question answering model which can handle multiple passages.

Unlike UQA which the supporting passage is certain for each question, UODQA needs to construct more than one passages through retrieval-based method and solve a multi-passage MRC problem.

As the first attempt to tackle UODQA, we propose a series of methods about how to synthesize data from a set of selected natural sentences and compare end-to-end performance, and finally our proposed method outperforms unsupervised method GPT-3 (zero-shot) (Brown et al., 2020) by a large margin and achieves up to 86% performance of previous popular supervised method DPR (Karpukhin et al., 2020) on three ODQA benchmarks.

2 Task Definition

For UODQA task, there is no limitation to use or construct data for training, only development and test sets from ODQA benchmark have to be used for evaluation and fair comparison. Therefore we will focus on data construction hereafter.

Based on a specific corpus $C$, several $<Q, P^+, P^-, A>$ triples are constructed. For each constructed example, $Q$ denotes the question, $A$ denotes the answer, $P^+$ denotes multiple positive passages that contains the answer supporting the question to solve, $P^-$ denotes multiple negative passages that do not contain the answer, and help
make the model learn to distinguish distracting information. To train a reader model, these data are leveraged to learn a function \( F(Q, P^+, P^-) = A \).

3 Method

3.1 Data Construction

The procedure is shown in Figure 1. The purpose is to automatically construct \(< Q, P^+, P^-, A >\) triples for model training. Obviously, the quality of constructed data decides whether a model can be trained well.

Firstly, based on some specific corpus \( C \), we select a set of sentences to construct \(< Q, A >\). In ODQA, most of the questions are factoid. Many works show that knowing Named Entities (NEs) may help construct \(< Q, A >\) pairs (Glass et al., 2020; Guu et al., 2020), thus a good practice is to select NEs as \( A \) in the constructed data. Meanwhile the sentence where the NE is from is \( Q \) after the selected NE is masked. The constructed \( Q \) is thus a pseudo-question, or conceptually defined as Information Request. In fact, both pseudo-question and real question may be used for effective model training to learn a question answering manner. Many works adopt question generation methods for better training, however, they also inevitably introduce noises.

When selecting sentences to generate \( Q \) from corpus \( C \), previous works on UQA do not set constraints (Lewis et al., 2019; Li et al., 2020; Hong et al., 2020), which brings no guarantee the constructed information request is reasonable or answerable. Such none-guarantee will become much more severe in UODQA. Basically, the source sentences selected need to have complete information. For example, “It was instead produced by Norro Wilson, although the album still had a distinguishable country pop sound.” is ambiguous because of too many coreferences. Moreover, when selecting phrases as \( A \), it needs to be answerable based on the constructed information request. For example, in sentence “Yao Ming played for the Houston Rockets of the National Basketball Association (NBA),” if the phrase “Yao Ming” is selected as \( A \), the constructed “___ played for the Houston Rockets of the National Basketball Association (NBA),” is not certain and answerable because there is not a short answer or named entity (which ODQA focuses on) that can be uniquely determined without ambiguity according to the context.

To obtain \(< Q, A >\) pairs with higher quality, we use sentences from the dataset in (Elsahar et al., 2019), which is an alignment corpus for WikiData and natural language. Each sentence is aligned with a Subject-Predicate-Object triple, and we select the object as \( A \). The dataset does not involve human labeling and is automatically created through a pipeline in terms of toolkits, a set of rules and several distant supervision assumptions.

To obtain \( P^+ \) and \( P^- \), our model retrieves documents from knowledge source \( S \), and selects the documents containing \( A \) as positive \( P^+ \) otherwise negative. This heuristic can not assure enough evidences but still make the model learn reasoning. To filter the trivial cases of \( P^+ \) that the context text surrounding the answer has too much overlap with that in the \( Q \), so the answer can be simply generated based on shortcuts, we set a window size \( n \) and check the left and right \( n \)-gram of the selected \( A \). Thus, \(< Q, P^+, P^-, A >\) triples are constructed.

3.2 Model Training

Following previous common practice in ODQA, we adopt retriever-reader architecture to perform UODQA. BM25 serves as retrieval metric in an unsupervised manner. After retrieving top \( K \) passages, a reader receives the questions and passages as input to output an answer. Following (Izacard and Grave, 2020b), we adopt a generative reader based on T5 (Raffel et al., 2020).

4 Experiments and Analysis

4.1 Evaluation Settings

The evaluation metrics are Exact Match (EM). For EM, if generated answer hits any one of the labeled list of possible golden answer, the sample is positive. The accuracy of EM is calculated as \( EM = N^+ / N \) where \( N^+ \) is number of positive samples and \( N \) is number of all evaluated samples.

We evaluate our model on three ODQA benchmarks, Natural Questions (Kwiatkowski et al., 2019), WebQuestions (Berant et al., 2013) and TriviaQA (Joshi et al., 2017). Statistics are shown in Table 1. The train/dev/test split follows (Lee et al., 2019). As this is an unsupervised ODQA task, we discard training set and only adopt de-

<table>
<thead>
<tr>
<th>Dataset</th>
<th>train</th>
<th>dev</th>
<th>test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Natural Questions</td>
<td>79,106</td>
<td>8,757</td>
<td>3,610</td>
</tr>
<tr>
<td>WebQuestions</td>
<td>3,417</td>
<td>361</td>
<td>2,032</td>
</tr>
<tr>
<td>TriviaQA</td>
<td>78,785</td>
<td>8,837</td>
<td>11,313</td>
</tr>
</tbody>
</table>

Table 1: Data statistics of three datasets.
development and test sets for evaluation. Natural Questions (NQ) is commonly used ODQA benchmark which was constructed according to real Google search engine queries. The answers are short phrases from Wikipedia articles containing various NEs. WebQuestions (WQ) contains questions collected from Google Suggest API, and the answers are all entities from the structured knowledge base (Freebase). TriviaQA (TQA) consists of trivia questions from online collection.

4.2 Implementation Details

We adopt dataset from (Elsahar et al., 2019) and select sentences that have only one object to construct question-answer pairs. Sentences containing character number more than 250 or less than 50 are discarded. The object is used as answer and we use the token [MASK] to replace the answer in the sentence as the question. Following (Karpukhin et al., 2020), the version of Wikipedia corpus we use is Dec. 20, 2018 dump and we split the whole corpus into 100-word segments as units of retrieval. For retrieving documents, we use Apache Lucene 1 to build index and perform BM25 retrieval. To filter $P^+$ using $n$-gram, we use $n$ as 3. We first retrieve top 100 documents, and select the top 40 documents to construct the input for reader. If none of top 40 documents contains the answer, we further find top 41-100 documents that contains the answer, and replace the 40th document with it, otherwise this sample is discarded. Finally, we obtain 844,100 samples to train for 60 hours using 8 nVidia V100s.

We implement the reader following (Izacard and Grave, 2020b) and perform training using learning rate of 1e-4, batch size of 256 and the number of concatenated passages each sample is 40. The model size setting we use is T5-base. We save and evaluate the model checkpoint every 500 training steps and stop training if the performance does not increase any more in 5 evaluations, and the checkpoint of best EM score is selected.

4.3 Results

In Table 2, the subscripts 10 and 50 denote number of passages to input the reader. As shown in this table, we perform experiments based on four kinds of settings, to study to what extent the quality of constructed training data affects performance. The difference among four settings RandSent, RandEnt, QuesGen and OurMethod is how to sample the question and answer. OurMethod refers to what we discuss in Section 3.1 and it is to use the alignment corpus to help sampling. RandSent means we select random sentences from Wikipedia articles and NEs to construct question-answer pairs. RandEnt means we use data from (Elsahar et al., 2019) and select a random NE from each sentence as answer. This expands the scope of types of answers and makes the model learn more diversified knowledge. QuesGen means we perform a question generation step after obtaining the constructed data based on our method. This makes the expression of the pseudo-question more close to the real question and makes the model learn a question answering manner better, but it may hurts the reasonability of constructed questions because of the noise introduced by question generation methods. Some examples are shown in Table 3.

![Figure 1: Our proposed method for synthesizing data and training.](image-url)
There are three main factors of the difference among different settings, reasonability, answerability and strategy to select answer span. Reasonability indicates to what extent the question conforms to the expression of natural language, answerability means whether the sentence describes an fact with accurate meanings and has enough information for deducing answer, and strategy to select answer span determines what knowledge the model learns.

For the setting of RandSent, because random sentences are usually ambiguous and lack enough evidence to infer corresponding answer, the answerability is very weak. For the setting of RandEnt, though the original sentence contains complete information and expresses accurate fact, the randomly masked NE may be too difficult to deduce. Compared with this, our strategy that only selects the object as answer is better, because in the structure of Subject-Predicate-Object, the object usually can be accurately deduced. QuesGen attempts to reform the expression of question to make it more like a real question, however, it also introduces noise to do harm to the performance. For the purpose of implementing unsupervised manner, we only adopt simple rule-based question-generation method, which applies semantic role labeling on the original question and selects one of the parsed argument as answer, and converts the order and tense of the sentence to reform it as a question expression. It indicates that if the question generation method introduces too much noise and hurts the reasonability of sentences too much, it is even worse than doing nothing and maintaining the statement expression of original constructed sentences.

In this paper, we exploit the possibility of completely discarding the training set to train an ODQA model, however, our results may be further improved when the training set can be adopted for further training or refining the reader like a course-to-fine process.

We note that many sampling methods for ODQA were raised in recent years such as ICT (Lee et al., 2019) and batch negative (Karpukhin et al., 2020), and they aim at training a better dense retriever and need to further fine-tune on the training data. However, our paper focuses on exploring what extent the ODQA system can achieve without high-quality human-annotated QA data for the first time. Besides, many works on Unsupervised Question Answering are proposed in recent years, though we aim at solving more challenging ODQA problem where no passage is given, the approaches of UQA may be adopted to further improve our method. We will leave this for future work.

5 Conclusion

In this paper, we first propose the task of Unsupervised Open-domain Question Answering, and explore to what extent it can perform based on our suggested data construction methods. We compare several strategies for synthesizing better data, as a result achieve up to 86% performance of previous supervised method. We hope this work inspires a new line of ODQA in the future and helps build more practical readers for real use.
References


Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. Pre-training transformers as energy-based cloze models. In EMNLP.


A Appendix

A.1 Related Work

A.1.1 Open-Domain Question Answering

Open-domain Question Answering (ODQA) needs to find answers from tremendous open domain information such as Wikipedia or web pages. Traditional methods usually adopt retriever-reader architecture (Karpukhin et al., 2020), which is to first retrieve relevant documents and then generate answers based on these retrieved documents, which is the main focus of our paper. Besides, there is also end-to-end method (Guu et al., 2020), but it costs too much computation resources to be widely applied. The improvements of retriever (Izacard and Grave, 2020a) and reader (Izacard and Grave, 2020b) are both critical for the overall performance, and there is still huge room for improvements.

A.1.2 Unsupervised Question Answering

Unsupervised Question Answering (UQA) is to alleviate the problem of huge cost of data annotation. Generally speaking, the key issue of UQA aims at automatically generating context-question-answer triples from publicly available data. (Lewis et al., 2019) uses an Unsupervised Neural Machine Translation method to generate questions. (Fabbri et al., 2020) proposes to retrieve relevant sentence that contains the answer and reform the sentence with template-based rules to generate questions. (Li et al., 2020) proposes an iterative process to refine the generated questions turn by turn. (Hong et al., 2020) proposes paraphrasing and trimming methods to respectively solve the problem of word overlap and unanswerable generated questions.