Self-Prompting Large Language Models for Open-Domain QA

Junlong Li[1,2] , **Zhuosheng Zhang**[1,2], **Hai Zhao**[1,2*]
[1]Department of Computer Science and Engineering, Shanghai Jiao Tong University
[2]Key Laboratory of Shanghai Education Commission for Intelligent Interaction and Cognitive Engineering, Shanghai Jiao Tong University
{lockonn, zhangzs}@sjtu.edu.cn, zhaohai@cs.sjtu.edu.cn

Abstract

Open-Domain Question Answering (ODQA) requires models to answer factoid questions with no context given. The common way for this task is to train models on a large-scale annotated dataset to retrieve related documents and generate answers based on these documents. In this paper, we show that the ODQA architecture can be dramatically simplified by treating Large Language Models (LLMs) as a knowledge corpus and propose a **Self-Prompting** framework for LLMs to perform ODQA so as to eliminate the need for training data and external knowledge corpus. Concretely, we firstly generate multiple pseudo QA pairs with background passages and one-sentence explanations for these QAs by prompting LLMs step by step and then leverage the generated QA pairs for in-context learning. Experimental results show our method surpasses previous state-of-the-art methods by +8.8 EM averagely on three widely-used ODQA datasets, and even achieves comparable performance with several retrieval-augmented fine-tuned models.

1 Introduction

Open-Domain Question Answering (ODQA) is a longstanding task in natural language processing that aims to answer questions about a wide range of world knowledge with no context given (Voorhees et al., 1999; Huang et al., 2020; Zhu et al., 2021; Zhang et al., 2022a). It is challenging even for humans without access to a large external knowledge corpus. The most common and de facto approach for ODQA now is the Retriever-Reader pipeline (Chen et al., 2017): first retrieving the most related documents of the question then applying the reader model to extract or generate the final answer conditioned on these documents (Karpukhin et al., 2020; Lewis et al., 2020; Izacard and Grave, 2021). Although perform well, these methods usually need to index the whole Wikipedia, leading to a huge storing cost, and the pipeline is also quite complicated.

With the emergence of Large Langauge Models (LLMs) like GPT3 (Brown et al., 2020), FLAN (Wei et al., 2022a), OPT (Zhang et al., 2022b), InstructGPT (Ouyang et al., 2022), some searchers start to use them for ODQA tasks. Through large-scale unsupervised pre-training, LLMs have stored sufficient knowledge in their parameters to answer most open-domain questions and can also recall them precisely given a simple query in natural language. Theoretically, they are capable of generating correct answers without any training data and external corpus, but in practice there still exists a clear gap between LLMs and fully fine-tuned models. To stimulate more potential of them, some attempts have been made in previous works, like encouraging models to generate a rationale called *chain-of-thought* before the final answer (Wei et al., 2022b; Kojima et al., 2022), or asking the model to first generate a contextual document and answer the question based on it in a second forward pass (Yu et al., 2022). However, these methods use only a small portion of the capabilities of LLMs and do not fully utilize a large number of other skills that may be helpful for ODQA.

In this paper, the scenario we focus on is ODQA with no training data and external corpus, and we propose **Self-Prompting** the LLM to explicitly activate various different capabilities of LLMs and combine these capabilities to further explore the upper bound of performance. In the preparation stage, the LLM is required to generate a pseudo QA dataset in the following steps: write a short Wikipedia passage, extract named entities in this passage as answers, raise corresponding questions for the answers, and explain each generated QA pair in a short sentence based on the passage. Relying on the strong instruction understanding ability of LLMs, all these sub-tasks can be perfectly con-

*Corresponding author

ducted with simple natural language prompts. We can automatically build a pseudo dataset by repeating these procedures, where each item is a high quality QA pair with a related context passage and a short sentence to explain it. During inference, we propose a novel clustering-based retrieval method to select both similar and diverse examples from this pseudo dataset as the in-context demonstrations for each test sample. These selected QA pairs, along with the passages and explanations, are concatenated with the test question in a specific order to form the final input sequence, which is then fed into the LLM to get the final answer.

We evaluate our methods on three ODQA benchmarks, including WebQ (Berant et al., 2013), NQ (Kwiatkowski et al., 2019) and TriviaQA (Joshi et al., 2017). Experimental results show that our method significantly surpass the plain zero-shot baseline (+16 EM on average), and also the previous SOTA method *GENREAD* (Yu et al., 2022) (+9 EM on average). We also conduct extensive ablation studies, case studies and analysis to discuss the effects of different generated components, the formats of placing them into sequence, ways of demonstration selection from the pseudo dataset, the quality of generated QAs, and many other aspects of our framework.

In general, our contributions can be summarized as follows:

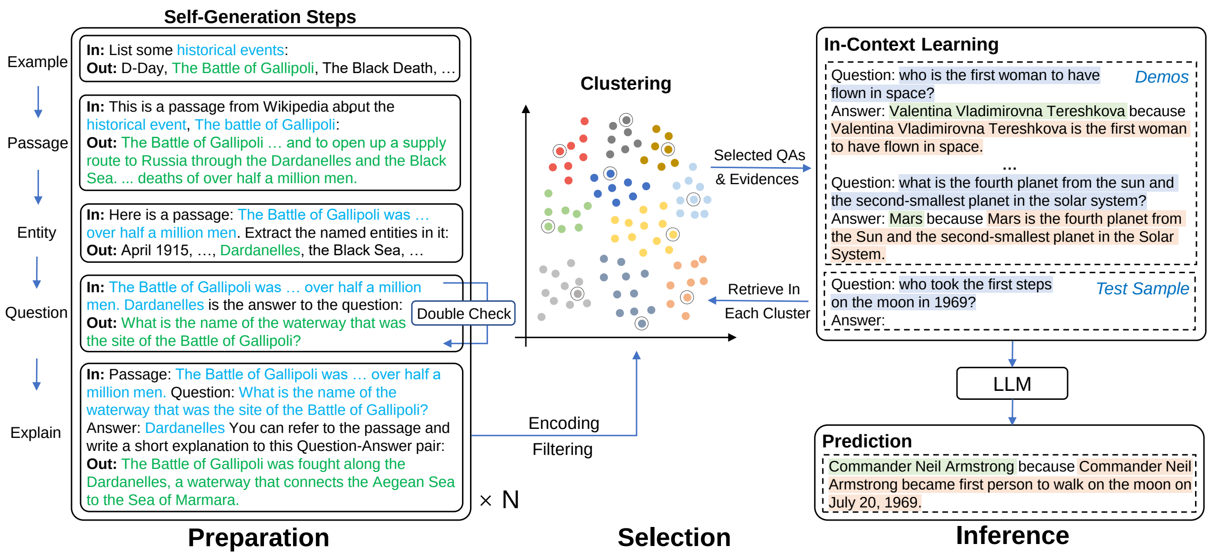
- We propose Self-Prompting to comprehensively combine multiple capabilities of LLMs for zero-shot ODQA. It is able to automatically build a pseudo but high-quality and annotated ODQA dataset in advance, and use it in a in-context learning manner.
- We propose a clustering-based retrieving method to effectively utilize the built pseudo dataset to select both semantically similar and diverse examples for each test sample.
- We conduct extensive experiments to show the effectiveness of Self-Prompting on three ODQA tasks. It outperforms previous SOTA under the zero-shot setting, and is comparable to several fine-tuned Retriever-Reader models.

2 Related Works

Retriever-Reader Models for ODQA The mainstream method to tackle ODQA tasks is the Retriever-Reader architecture. It first leverage a retriever over a large knowledge corpus like Wikipedia to select several related documents that may contain the answer, then a reader is used to process the retrieved documents and predict the final answer. Conventional models use sparse retrieval methods such as TF-IDF or BM25, while recent works choose dense retrieval based on the representation vector encoded by pre-trained language models (Karpukhin et al., 2020; Lewis et al., 2020; Guu et al., 2020; Izacard and Grave, 2021). As for the reader, there are also two different choices: extractive readers like BERT (Devlin et al., 2019) or generative ones like T5 (Raffel et al., 2020). A similar work in this branch is PAQ (Lewis et al., 2021), which first generated 65 million probably-asked questions based on the complete dump of Wikipedia, and directly retrieve these questions instead of documents.

LLM and In-context Learning Generally, Large Language Models (LLMs) refer to the pre-trained models with tens or hundreds of billions of parameters. Some preminent examples are GPT3, FLAN, PaLM, OPT, and InstructGPT (Brown et al., 2020; Wei et al., 2022a; Chowdhery et al., 2022; Zhang et al., 2022b; Ouyang et al., 2022). These models are trained with large-scale unsupervised learning, and are able to perform NLP tasks by converting inputs into natural language queries without further training. The cost of fine-tuning these models is extremely huge, so the usual way of using them is in-context learning, which is to put some input-output pairs as demonstrations in front of the test sample. Some previous works have investigate the calibration (Zhao et al., 2021), example selection (Liu et al., 2022a; Rubin et al., 2022) and ordering (Lu et al., 2022b) of in-context learning, while to the best of our knowledge we are the first to use LLM itself to generate the examples used in in-context learning.

Enhancing Models with LLM generation A recent line of researches aim to use the generation outputs of LLMs to facilitate the training of small models. For example, Ye et al. (2022) use GPT2 (Radford et al., 2019) to generate pseudo data to train tiny language models, and Wang et al. (2022) distill the knowledge of GPT3 into GPT2 for commonsense QAs. Another line of works try directly using contents generated by LLM itself. Some works use LLMs to first generate relevant contexts or background documents then providing



In: List some historical events:
Out: D-Day, The Battle of Gallipoli, The Black Death, …

In: This is a passage from Wikipedia about the historical event, The battle of Gallipoli:
Out: The Battle of Gallipoli … and to open up a supply route to Russia through the Dardanelles and the Black Sea. ... deaths of over half a million men.

In: Here is a passage: The Battle of Gallipoli was … over half a million men. Extract the named entities in it:
Out: April 1915, …, Dardanelles, the Black Sea, …

In: The Battle of Gallipoli was … over half a million men. Dardanelles is the answer to the question:
Out: What is the name of the waterway that was the site of the Battle of Gallipoli?

In: Passage: The Battle of Gallipoli was … over half a million men. Question: What is the name of the waterway that was the site of the Battle of Gallipoli? Answer: Dardanelles You can refer to the passage and write a short explanation to this Question-Answer pair:
Out: The Battle of Gallipoli was fought along the Dardanelles, a waterway that connects the Aegean Sea to the Sea of Marmara.

Question: who is the first woman to have flown in space?
Answer: Valentina Vladimirovna Tereshkova because Valentina Vladimirovna Tereshkova is the first woman to have flown in space.
…
Question: what is the fourth planet from the sun and the second-smallest planet in the solar system?
Answer: Mars because Mars is the fourth planet from the Sun and the second-smallest planet in the Solar System.

Question: who took the first steps on the moon in 1969?
Answer:

Commander Neil Armstrong because Commander Neil Armstrong became first person to walk on the moon on July 20, 1969.

Figure 1: The overall framework for Self-Prompting on ODQA. In the self-generation steps, blue refers to contents generated in previous steps or manually designed topics and green is the newly generated texts. In inference, question, answer, and explanation are question, answer and explanation for both demonstrations and test sample.

them as additional input when answering questions (Liu et al., 2022b; Yu et al., 2022), while others focus on eliciting a series of intermediate reasoning steps referred to as *chain-of-thought* for arithmetic problems (Wei et al., 2022b; Kojima et al., 2022; Zhang et al., 2022c).

3 Approach

We present details of Self-Prompting in this section. It can be divided into two stages: preparation and inference. In the first stage, we require the LLM to automatically build a pseudo ODQA dataset by prompting it to generate QA pairs with context passages and explanations. In the second stage, we dynamically select several examples in the pool through a clustering-based retrieval method as in-context demonstrations to help understanding and answering the given question. An overall framework is shown in Figure 1.

3.1 QA Pool Generation

As a preparation in advance, we first ask the LLM to automatically generate a QA pool as a pseudo dataset. It follows the following steps:

Passage Generation To ensure the diversity of the generated passages, we first manually design some topics (like *countries, books, tourist attractions* etc.) that are likely to appear in ODQA, referring to the dataset statistics in TriviaQA (Joshi et al., 2017). For each topic, the LLM is asked to list some examples with instructions like `"List some {topic}:"`, and this step is repeated until we have collected a certain number of different examples for this topic. Through this, we obtain a huge number of examples covering different categories and they are leverared to generate short Wiki-style passages with the following prompt `"This is a passage from Wikipedia about the {topic}, {example}:"`.

Named Entity Recognition We extract the named entities in these generated passages as the candidate answers. Usually, this NER step is conducted by a fine-tuned small model. In our framework, this is done also by LLM itself. For a given generated passage, the prompt might be `"Here is a passage: {passage} Extract the named entities in it:"`, and we can get the entities in this passage.

Question Generation Named entities (like date, name, location) extracted in the previous step are used as the candidate answers. We then ask the LLM to output a proper question for this answer based on the given passage with prompts like `"{passage} {entity} is the answer to the question:"`. To ensure the correctness of the QA pair, we ask the LLM to do a double-check, i.e., reanswer the question based on the passage to see whether it can recover the entity. In practice, we observe that the conflicts between new predictions and the raw entities are often caused by the failure of generating related questions, while in contrast the new predictions often match the question well.

So we keep the new predictions as final answers.

Explain the QA pair For each QA pair, we ask the LLM to return a piece of one-sentence explanation for it based on the passage. The prompt is like `"Passage: {passage} Question: {question} Answer: {answer} You can refer to the passage and write a short explanation to this Question-Answer pair:"`. In this step, we try to elicit the summarization and induction skills of LLM to provide a fine-grained annotation for the generated QA pairs.

3.2 Dynamic In-context Demonstrations Selection for Inference

It is a open question that how to use the pseudo QA dataset the LLM have generated in the preparation stage. We focus on two aspects, namely selection and format.

Clustering-based Retrieval Some previous works point out using examples with high semantic similarity as in-context demonstrations brings benefits (Liu et al., 2022a), while others claim a fixed set of examples based on clustering is better (Zhang et al., 2022c). We propose to combine these two ways together. First, each QA pair is encoded into a vector representation over the whole QA pool with Sentence-BERT (Reimers and Gurevych, 2019). Suppose k examples are needed for in-context demonstration, the pseudo QAs will be clustered into k categories by the k-means algorithm. For a given question, we also use Sentence-BERT to encode it, and retrieve the most similar example from each cluster with a simple cosine similarity. This selection method has a balance of the similarity and diversity of the demonstrations.

Answer then Explain Finally is the organization format of these selected examples. In the input sequence, we first put these examples sequentially in the format of *Question*$\rightarrow$ *Answer* $\rightarrow$ *Explanation*, and place the test question at the end of the sequence. The specific template is shown in Appendix. By doing so, the LLM is capable of viewing much more information then just switching to a QA mode, and it can also give out a brief explanation for the its answer. This is quite different from the common practice in *chain-of-thought* prompting, i.e., generating a rationale before the answer, but our experiments prove the effectiveness of the former choice.

Datasets	Test	Q Words	A Words	Answers
WebQ	2.0K	6.8	2.5	2.4
NQ	3.6K	9.1	2.2	2.0
TriviaQA	11K	14.0	2.5	14.0

Table 1: Statistics for each dataset, Answers, Q Words and A Words refer to the average number reference answers, words in question, and words in answer for each sample respectively in the Test split.

4 Experiments

4.1 Datasets and Settings

In this paper we conduct experiments on three ODQA benchmarks, including WebQ (Berant et al., 2013), NQ (Kwiatkowski et al., 2019) and TriviaQA (Joshi et al., 2017). The dataset statistics are in Table 1.

We use InstructGPT (Ouyang et al., 2022) (the `text-davinci-002` of GPT3 (Brown et al., 2020)) as the LLM, which keep the same with previous works (Wei et al., 2022b; Kojima et al., 2022; Yu et al., 2022). The exact model we used as Sentence-BERT is `all-mpnet-base-v2`, and the number of demonstrations for in-context learning is 10.

In passage generation, we design 29 topics in advance. Their names and numbers of required examples for each topic are in Appendix. In question generation, we ban some pronouns like *they, he, she* by setting their `logit_bias` as -100 in the API call, to prevent getting ambiguous questions (e.g. *what did he do in 1997?*). After the generation process, we filter the QA pair where the answer has more than 5 words or the LLM output an explanation sentence with no answer span in it. For each passage, the upper limit of generating QA pairs is 10. After filtering duplicated questions, we collect 1,216 passages and 4,883 QA pairs with explanations. The parameters like `max_tokens` or `temperature` for LLM generation in each step are in Appendix. The metrics we evaluate is Exact Match (EM), with the same answer normalization as in Karpukhin et al. (2020). We also observe that in WebQ, if the correct answer contains multiple listed entities, the reference is often given as only one of them, so we perform an additional post-processing on this dataset to extract only the first when the LLM predicts multiple ones (e.g. only return *A* if the raw prediction is *A, B and C*).

The baselines we select include direct prompting: InstructGPT (Ouyang et al., 2022), GENREAD (Yu

Models	# Total Params.	Train Data	External Corpus	WebQ	NQ	TriviaQA	Avg.
**fine-tuned models without retrieval*							
T5-SSM (Roberts et al., 2020)	11B	✓	✗	40.8	34.8	51.0	42.2
**retrieval-augmented fine-tuned models*							
REALM (Guu et al., 2020)	330M	✓	✓	40.7	40.4	55.8	45.6
DPR (Karpukhin et al., 2020)	330M	✓	✓	41.1	41.5	56.8	46.5
RAG (Lewis et al., 2020)	620M	✓	✓	45.2	44.5	56.1	48.6
**retrieval-augmented prompting LLMs (DPR trained on target datasets)*							
Google+InstructGPT	175B	✗	✓	19.9	27.8	58.7	35.5
DPR+InstructGPT	175B	✗	✓	20.1	29.9	55.3	35.1
**directly prompting LLMs*							
InstructGPT	175B	✗	✗	18.6	20.9	52.6	30.7
GENREAD (InstructGPT) (Yu et al., 2022)	175B	✗	✗	24.8	28.2	59.3	37.4
**our method, self prompting by generating pseudo QA for in-context learning*							
Self-Prompting (InstructGPT)	175B	✗	✗	35.6	36.2	66.8	46.2

Table 2: Main results on three ODQA benchmarks, Self-Prompting is free from any training data and external knowledge corpus. # Total Params. is the total number of model paramaters in this system (e.g. RAG use 2 BERT-base with 110M$\times$2 and 1 BART-large with 400M). Train Data refers to whether the system uses training data for training, and External Corpus means whether a external knowledge corpus is used to retrieve documents.

et al., 2022); retrieval-augmented LLM prompting: DPR+InstructGPT, Google+InstructGPT; fine-tuned models with no retrieval: T5-SSM 11B (Roberts et al., 2020); retrieval-augmented fine-tined models: REALM (Guu et al., 2020), DPR (Karpukhin et al., 2020), RAG (Lewis et al., 2020).

4.2 Main Results

The main results are shown in Table 2. Compared to directly prompting methods, our Self-Prompting method surpass the InstructGPT baseline by +15.5 EM on average, and the previous SOTA method GENREAD by +8.8 EM on average. This strongly indicates that the way of first generating high-quality and annotated pseudo dataset and using them for in-context learning can comprehensively invoke the capabilities of LLM in different aspects, which brings a significant improvement over the simple and crude ways of directly LLM prompting. Self-prompting is also better than retrieval-augmented prompting methods, which show that LLM itself has stored enough world knowledge in its paramaters, so there is no need to explicitly collect a large external corpus for retrieving.

We notice that Self-Prompting achieves higher EM than T5-SSM 11B on two datasets except WebQ, even though we do not give any training data to InstructGPT, showing that the potential of LLMs for ODQA is large under a zero-shot setting. Finally, we find that Self-Prompting gets comparable results to some powerful retrieval-augmented fine-tuned models with regard to the average score on three datasets, especially on TriviaQA we see more than +10 EM. On WebQ and NQ, Self-Prompting lags behind these methods, but we find that this is mainly caused by the features of these two datasets like fewer reference answers for each question or outdated answers. We will introduce such phenomenon in detail in the Analysis section with case studies.

5 Analysis

To save the cost of using the OpenAI APIs, we conduct several ablation studies on subsets of these three datasets by randomly selecting 1,000 samples from their test sets.

5.1 How to Use Generated Passages and Explanations

A crucial question is how to use the byproducts, namely the passages and explanations, to form a better format for in-context learning. Given a list of 10 demonstrations (Q, A, P, E)$\times$ 10 (by clustering-based retrieval), we investigate several input for-

Demos	Pred.	WebQ	NQ	TriviaQA	Avg.
**one iteration*					
QA	Q→A	35.6	**37.8**	68.2	47.2
QAE	Q→AE	**38.7**	37.2	**68.7**	**48.2**
QAP	Q→AP	37.6	35.6	67.6	46.9
QEA	Q→EA	34.6	33.8	61.6	43.3
QPA	Q→PA	32.0	31.2	57.8	40.3
QAEP	Q→AEP	35.4	37.6	67.8	46.9
QAPE	Q→APE	37.2	34.8	67.2	46.4
**two iterations*					
1: QP	Q→P	-	-	-	-
2: PQA	PQ→A	30.6	32.0	62.0	41.5
1: QE	Q→E	-	-	-	-
2: EQA	EQ→A	34.8	32.8	63.4	43.7

Table 3: Using different formats for in-context learning.

mats as shown in Table 3. The *one iteration* methods mean only one call of API is needed to generate the answer and the passage/explanation, while in *two iteration* methods the LLM needs generate the passage/explanation first and put them into input sequence to generate answers in the second API call. Detailed templates for these methods are in Appendix. From Table 3 we see that the simplest QA format is sufficient to output good performance, and only QAE surpasses it. The other three answer-then-explain formats QAP, QAEP, QAPE are worse than QA, which indicates that the redundant information in passages is harmful to LLMs. We also find that the two *chain-of-thought* style formats, QEA and QPA, are much worse than the baselines. In Lu et al. (2022a) and Wei et al. (2022b), similar findings are concluded that *chain-of-thought* is beneficial to complex, multi-hop math reasoning tasks but has little impact on commonsense questions. Finally, the *two iterations* methods also lead to a large performance drop, which makes them the worst settings considering the doubled inference time and cost.

5.2 Ways of Demonstration Selection

Since the size of the pseudo QA dataset automatically generated by the LLM is much larger than the number of examples we put in the input sequence for in-context learning, a proper selection method is necessary. We conduct experiments with four settings: randomly selecting QAs from the pool

Selection	WebQ	NQ	TriviaQA	Avg.
Random	$34.7_{1.4}$	$36.9_{1.5}$	$67.8_{1.6}$	$46.4_{1.3}$
Retrieve	36.2	37.0	66.8	46.7
ClusterCenter	35.0	**37.4**	**68.8**	47.1
RetrieveInCluster	**38.7**	37.2	68.7	**48.2**

Table 4: Different ways of selection demonstrations. We run random selection for 5 times with different seeds, and report their mean value and standard deviation.

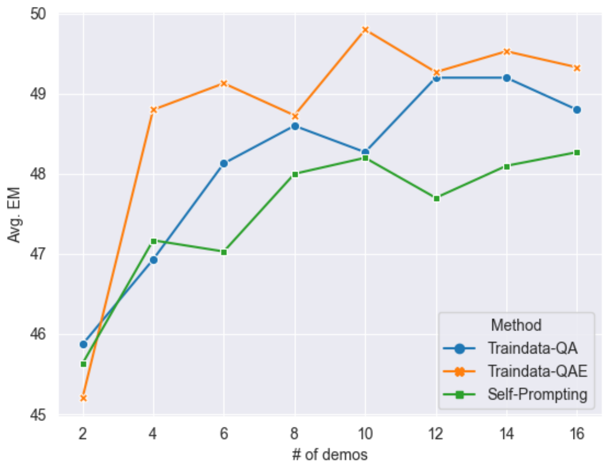


Figure 2: Performance of different numbers of demonstrations in Self-Prompting, as well as using a fixed set of training data as demonstrations for each dataset.

(Random), retrieving the most similar QAs with cosine similarity globally (Retrieve), selecting the closest QA to the centroid in each cluster (ClusterCenter), and retrieving the most similar QA in each cluster (RetrieveInCluster). The other hyper-parameters are kept the same, i.e., 10 demonstrations and QAE format. Results in Table 4 show that random selection performs the worst, and it also suffers from instability. Both the Retrieve and ClusterCenter methods have some improvements over Random, but the progress is not universal over the three datasets. The method we propose, RetrieveInCluster, can incorporate the advantages of both methods to select both diverse and semantically similar demonstrations, so it is robust enough to achieve satisfactory scores among all datasets.

5.3 Different Number of Demonstrations

A natural idea of in-context learning is to put as many as possible examples in the input sequence, so we also investigate the effect of different number of demonstrations, as shown in the green line in Figure 2. We report the average EM score over the

Topic: Movie or TV series **Example:** Hocus Pocus **Passage:** Hocus Pocus is a 1993 American Halloween fantasy comedy horror film directed by Kenny Ortega, produced by Walt Disney Pictures, and written by Neil Cuthbert and Mick Garris. The film stars Bette Midler, Kathy Najimy, and Sarah Jessica Parker as a trio of witches who are inadvertently resurrected by a teenage boy in Salem, Massachusetts. The film was released on July 16, 1993, and grossed $39 million in its opening weekend. It went on to gross $817 million worldwide. **Entity:** Neil Cuthbert and Mick Garris **Question:** Who wrote Hocus Pocus? **Explanation:** Hocus Pocus is a 1993 American Halloween fantasy comedy horror film directed by Kenny Ortega, produced by Walt Disney Pictures, and written by Neil Cuthbert and Mick Garris.
Topic: Actor or actress **Example:** George Clooney **Passage:** George Clooney is an American actor, director, producer, screenwriter, and businessman. He is the recipient of three Golden Globe Awards and two Academy Awards, one for acting in Syriana (2005) and the other for co-producing Argo (2012). In 2018, he was the recipient of the AFI Lifetime Achievement Award. **Entity:** Argo **Question:** What is the name of the film that won George Clooney an Academy Award for co-producing? **Explanation:** George Clooney won an Academy Award for co-producing the film "Argo."
Topic: Historical event **Example:** The Battle of Gallipoli **Passage:** The Battle of Gallipoli was a military campaign that took place during World War I. The campaign was fought by the British and French against the Ottoman Empire and lasted from April 1915 to January 1916. The battle was fought in an effort to force the Ottoman Empire out of the war, and to open up a supply route to Russia through the Dardanelles and the Black Sea. The campaign ended in failure, and resulted in the deaths of over half a million men. **Entity:** Dardanelles **Question:** What is the name of the waterway that was the site of the Battle of Gallipoli? **Explanation:** The Battle of Gallipoli was fought along the Dardanelles, a waterway that connects the Aegean Sea to the Sea of Marmara.

Table 5: Three examples in the pseudo QA dataset.

three datasets with number of demonstrations in {2, 4, . . . , 14, 16}. When the number is in 2-10, the performance of our method generally becomes better as the number increases, and using more than 10 examples does not bring significant improvements. As a result, we choose 10 demonstrations in our main experiments for both performance and cost considerations.

5.4 Comparison between Self-Prompting and Using Training Data

To evaluate the quality of the pseudo dataset generated by LLM, we randomly select a set of samples from their training sets for in-context learning, and annotate them with related Wiki passages and short explanation sentences as what is automatically done in Self-Prompting. Following Section 5.3, we report the average EM score on three subsets with 2 - 16 demonstrations, and try both the QA and QAE formats. Results in Figure 2 reveals that our Self-Prompting method performs on par with using the manually annotated training data. Compared with Traindata-QA, Self-Prompting is only about 1 EM lower than it at different number of demonstrations, showing that the LLM itself is powerful enough to tackle ODQA with a fine-grained and step-by-step guide, even without any training data. We also observe a stable boost from

Case	Examples	Ratio (%) WebQ	NQ	TriviaQA
AW, EW	**Q:** what is the only anagram of the word ‘english‘? **Ref:** Shingle; **Pred:** Elinghs	16*	40	59
Need Details	**Q:** what countries does greece share borders with? **Ref:** Turkey; **Pred:** several countries	10	3	1
Form	**Q:** who is judy garland father? **Ref:** Francis Avent Gumm; **Pred:** Frank Gumm	34	25	31
Multiple	**Q:** who are china’s neighbors? **Ref:** Pakistan; **Pred:** Russia	15	6	5
RW	**Q:** who plays caesar flickerman in the hunger games? **Ref:** Art Conforti; **Pred:** Stanley Tucci	9	12	1
Open	**Q:** what is there to do for fun in kansas city? **Ref:** Kemper Arena; **Pred:** visit Kansas City Zoo	9	1	1
Time	**Q:** who did carlos boozer play for? **Ref:** Utah Jazz; **Pred:** the Chicago Bulls	6	6	2
Unanswerable	**Q:** the legend of heroes trails in the sky the 3rd vita **Ref:** July 14, 2016; **Pred:** PlayStation Vita	1	7	0

Table 6: Error Analysis on 100 randomly select samples from each dataset with EM Score=0. To save space, only one reference answer is displayed if there are multiple ones. *We note that in WebQ, there are 2 cases where the answer is wrong but the explanation is correct, and 1 case with correct answer and wrong explanation.

Traindata-QA to Traindata-QAE, illustrating that the QAE format can be effective not only on the pseudo-data constructed by LLM, but also on the real training data.

5.5 Data Generation Quality Analysis

To further explore the quality of the pseudo dataset LLM generated, we pick three examples and put them here in Table 5 as a case study, with key sentences in passage highlighted in teal and answer entities in blue. Overall, these generated passages are accurate, but they still contain some factual mistakes, or hallucination (Ji et al., 2022), in the texts (highlighted by red). The questions generated for extract entities include different types (e.g., people, item, location). They are proper and answerable even with no context given, which is in line with common open-domain questions. As an important part of Self-Prompting, we can see the explanations written by the LLM are of high quality. In the first example, the LLM precisely extracts the key sentence from the passage; in the second example, the LLM successfully conducts co-reference resolution to replace *He* with *George Clooney* and removes redundant texts; in the last example, the LLM not only summarizes the key sentence, but also adds extra information not mentioned in ths passage to its output. In all, Self-Prompting can automatically generate pseudo but high quality ODQA dataset with passages and explanations as annotations.

5.6 Error Analysis

Finally we conduct a study for error analysis to see why Self-Prompting fails on some questions. We randomly select 100 questions that Self-Prompting gets EM Score=0 from each dataset, and manually examine them. From these 300 samples, we conclude 3 major and 8 minor types: **1) True Negative** - AW, EW (both the prediction and explanation are incorrect), Need Details (the prediction is not specific); **2) False Negative** - Form (the prediction and the reference are the same thing but with different form), Multiple (the prediction is not in the list of references but also a correct answer), RW (the reference itself is incorrect); **3) Bad Question** - Open (open question with no exact answers), Time (cannot be answered if not clarifing the time), Unanswerable (information to answer the question is incomplete). The results are shown in Table 6. We observe that there is a large number of False Negative among all three datasets (58, 43, 37), so Self-Prompting is largely under-estimated. Most of them are Form, which indicates that the EM Score is a sub-optimal metric to evaluate ODQA systems. The quality of WebQ and NQ are low according to the table, as they have lots of poorly annotated answers (Multiple, RW) and bad questions (Open, Time, Unanswerable). Especially, many questions in these two datasets rely on the Wiki Dump of a certain time point, and using the latest one even hurts the performance (Izacard et al., 2022; Yu et al., 2022). In TriviaQA, the True Negative rate is much more higher than the other two, so it can better reflect the performance of Self-Prompting. This is also a potential reason to explain why Self-Prompting outperforms fine-tuned baselines significantly but has lower scores on WebQ and NQ.

6 Conclusion

In this paper, we propose Self-Prompting Large Language Models (LLMs) for Open-Domain Question Answering (ODQA). Our method requires the LLM to generate pseudo QA dataset with matching passages and explanations, and use them for in-context learning. It successfully stimulates the potential of the LLM in ODQA by explicitly activate various language understanding abilities and elicit the world knowledge in its parameters. With no training data and external corpus, Self-Prompting surpasses previous SOTA significantly, and performs on par with several retrieval-augmented fine-tuned models. In future, we will improve the framework, for example to eliminate hallucination in generation and reduce manual design. We will also expand Self-Prompting to other kinds of NLP tasks to prove its universal effectiveness, and further release the power of LLMs in different fields.

References

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*, pages 1533–1544, Seattle, Washington, USA. Association for Computational Linguistics.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading Wikipedia to answer open-domain questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1870–1879, Vancouver, Canada. Association for Computational Linguistics.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2022. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. 2020. Retrieval augmented language model pre-training. In *International Conference on Machine Learning*, pages 3929–3938. PMLR.

Zhen Huang, Shiyi Xu, Minghao Hu, Xinyi Wang, Jinyan Qiu, Yongquan Fu, Yuncai Zhao, Yuxing Peng, and Changjian Wang. 2020. Recent trends in deep learning based open-domain textual question answering systems. *IEEE Access*, 8:94341–94356.

Gautier Izacard and Edouard Grave. 2021. Leveraging passage retrieval with generative models for open domain question answering. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 874–880, Online. Association for Computational Linguistics.

Gautier Izacard, Patrick Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane Dwivedi-Yu, Armand Joulin, Sebastian Riedel, and Edouard Grave. 2022. Few-shot learning with retrieval augmented language models. *arXiv preprint arXiv:2208.03299*.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Yejin Bang, Andrea

Madotto, and Pascale Fung. 2022. Survey of hallucination in natural language generation. *ACM Computing Surveys*.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611, Vancouver, Canada. Association for Computational Linguistics.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, Online. Association for Computational Linguistics.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *arXiv preprint arXiv:2205.11916*.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474.

Patrick Lewis, Yuxiang Wu, Linqing Liu, Pasquale Minervini, Heinrich Küttler, Aleksandra Piktus, Pontus Stenetorp, and Sebastian Riedel. 2021. PAQ: 65 million probably-asked questions and what you can do with them. *Transactions of the Association for Computational Linguistics*, 9:1098–1115.

Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. 2022a. What makes good in-context examples for GPT-3? In *Proceedings of Deep Learning Inside Out (DeeLIO 2022): The 3rd Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*, pages 100–114, Dublin, Ireland and Online. Association for Computational Linguistics.

Jiacheng Liu, Alisa Liu, Ximing Lu, Sean Welleck, Peter West, Ronan Le Bras, Yejin Choi, and Hannaneh Hajishirzi. 2022b. Generated knowledge prompting for commonsense reasoning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3154–3169, Dublin, Ireland. Association for Computational Linguistics.

Pan Lu, Swaroop Mishra, Tony Xia, Liang Qiu, Kai-Wei Chang, Song-Chun Zhu, Oyvind Tafjord, Peter Clark, and Ashwin Kalyan. 2022a. Learn to explain: Multimodal reasoning via thought chains for science question answering. In *The 36th Conference on Neural Information Processing Systems (NeurIPS)*.

Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. 2022b. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8086–8098, Dublin, Ireland. Association for Computational Linguistics.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *arXiv preprint arXiv:2203.02155*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, Peter J Liu, et al. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(140):1–67.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Adam Roberts, Colin Raffel, and Noam Shazeer. 2020. How much knowledge can you pack into the parameters of a language model? In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5418–5426, Online. Association for Computational Linguistics.

Ohad Rubin, Jonathan Herzig, and Jonathan Berant. 2022. Learning to retrieve prompts for in-context learning. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2655–2671, Seattle, United States. Association for Computational Linguistics.

Ellen M Voorhees et al. 1999. The trec-8 question answering track report. In *Trec*, volume 99, pages 77–82.

Wenya Wang, Vivek Srikumar, Hanna Hajishirzi, and Noah A Smith. 2022. Elaboration-generating commonsense question answering at scale. *arXiv preprint arXiv:2209.01232*.

Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V Le. 2022a. Finetuned language models are zero-shot learners. In *International Conference on Learning Representations*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Chi, Quoc Le, and Denny Zhou. 2022b. Chain of thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*.

Jiacheng Ye, Jiahui Gao, Qintong Li, Hang Xu, Jiangtao Feng, Zhiyong Wu, Tao Yu, and Lingpeng Kong. 2022. Zerogen: Efficient zero-shot learning via dataset generation. *arXiv preprint arXiv:2202.07922*.

Wenhao Yu, Dan Iter, Shuohang Wang, Yichong Xu, Mingxuan Ju, Soumya Sanyal, Chenguang Zhu, Michael Zeng, and Meng Jiang. 2022. Generate rather than retrieve: Large language models are strong context generators. *arXiv preprint arXiv:2209.10063*.

Qin Zhang, Shangsi Chen, Dongkuan Xu, Qingqing Cao, Xiaojun Chen, Trevor Cohn, and Meng Fang. 2022a. A survey for efficient open domain question answering. *arXiv preprint arXiv:2211.07886*.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. 2022b. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*.

Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. 2022c. Automatic chain of thought prompting in large language models. *arXiv preprint arXiv:2210.03493*.

Zihao Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021. Calibrate before use: Improving few-shot performance of language models. In *International Conference on Machine Learning*, pages 12697–12706. PMLR.

Fengbin Zhu, Wenqiang Lei, Chao Wang, Jianming Zheng, Soujanya Poria, and Tat-Seng Chua. 2021. Retrieving and reading: A comprehensive survey on open-domain question answering. *arXiv preprint arXiv:2101.00774*.

A Names and Required Number of Examples for Topics

We list the names and required number of examples for each topic in Table 7. In general, we require about 100 examples for each high-level category (e.g., *Art, History, Sports*).

Topic	Number
politician	100
athlete	40
sports team	40
sports event	40
country	40
city	60
historical figure	50
historical event	50
war	40
religion	20
singer	50
song	50
actor or actress	50
movie or TV series	50
writer	30
book	30
painter	30
painting	30
composer	30
classical music	30
tourist attraction	100
scientist	40
scientific term	40
video game	40
animal	40
plant	40
food	40
enterprise	50
international organization	50

Table 7: Names and required examples for the 29 manually designed topics.

Avg. Words	
Passage	77.41
Question	9.52
Answer	2.04
Explanation	13.37
Question Type	
where	204
how	335
who	898
what	2,814
when	475
which	151
others	6

Table 8: Statistics of the generated pseudo QA dataset.

Generation Step	Prompt	max_tokens	temperature
Example	List some {topic}, separated by '|':	1024	1
Passage	This is a passage from Wikipedia about the {topic}, {example}:\n	256	0
Entity	Here is a passage: {passage}\n\nExtract the named entities (like date, location, organization, character, number) in it, and separate them by '|'. If no named entity in it, write 'None' only.	50	0
Question	{passage} \n{entity} is the answer to the question:	50	0
Double Check	Passage: {passage}\nQuestion: {question}\nShort Answer (extracted from the passage, less than 6 words):	50	0
Explain	Passage: {passage}\nQuestion: {question} Answer: {answer}\nYou can refer to the passage and write a short explanation to this Question-Answer pair, "{answer}" must in the explanation:	50	0

Table 9: Parameters and detailed prompts used in each generation step.

Format	Template	max_tokens
**one iteration*		
QA	Question: {question} \n\n The answer (just one entity) is {answer}	20
QAE	Question: {question} \n\n The answer (just one entity) is {answer} because {explanation}	128
QAP	Question: {question} \n\n The answer (just one entity) is {answer} referring to the passage: {passage}	256
QEA	Question: {question} \n\n {explanation} So the answer (just one entity) is {answer}	256
QPA	Question: {question} \n\n {passage} So the answer (just one entity) is {answer}	256
QAEP	Question: {question} \n\n The answer (just one entity) is {answer} because {explanation} referring to the passage: {passage}	256
QAPE	Question: {question} \n\n The answer (just one entity) is {answer} referring to the passage: {passage} so {explanation}	256
**two iterations*		
1. QP	Generate a background document from Wikipedia to answer the given question. {question}\n{passage}	256
2. PQA	Passage: {passage} \n\n Question: {question} \n\n Referring to the passage above, the correct answer (just one entity) to the given question is {answer}	20
1. QE	Generate a piece of evidence to answer the given question. {question}\n{explanation}	256
2. EQA	Evidence: {explanation} \n\n Question: {question} \n\n Referring to the evidence above, the correct answer (just one entity) to the given question is {answer}	20

Table 10: Specific templates for different input formats.

B Details in Generating Pseudo QA Dataset

We present the exact prompts and API parameters used in self-generation steps in Table 9. After getting the passage, we use the NLTK toolkit[1] to remove the last sentence of it if the last sentence is incomplete or truncated.

C Statistics of the Generated Pseudo QA datasets

We show some statistics of the generated pseudo QA dataset in Table 8.

D Specific Templates in Different Formats

The specific templates used in main experiments (QAE) and in Section 5.1 are shown in Table 10. We refer to Yu et al. (2022) to design them. The `temperature` is 0 for all formats.

[1]https://www.nltk.org