Self-Prompting Large Language Models for Zero-Shot Open-Domain QA

Junlong Li[1,4] **, Jinyuan Wang**[2,4]**, Zhuosheng Zhang**[3,*]**, Hai Zhao**[1,4,*]
[1]Department of Computer Science and Engineering, Shanghai Jiao Tong University
[2]SJTU-Paris Elite Institute of Technology, Shanghai Jiao Tong University
[3]School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University
[4]Key Laboratory of Shanghai Education Commission for Intelligent Interaction
and Cognitive Engineering, Shanghai Jiao Tong University
{lockonn, steve_wang, zhangzs}@sjtu.edu.cn, zhaohai@cs.sjtu.edu.cn

Abstract

Open-Domain Question Answering (ODQA) aims to answer questions without explicitly providing specific background documents. This task becomes notably challenging in a zero-shot setting where no data is available to train tailored retrieval-reader models. While recent Large Language Models (LLMs) like GPT-3 have demonstrated their effectiveness in zero-shot ODQA using direct prompting methods, these methods still fall short of fully harnessing the potential of LLMs when implicitly invoked. In this paper, we propose a **Self-Prompting** framework to explicitly utilize the massive knowledge encoded in the parameters of LLMs and their strong instruction understanding abilities. Concretely, we prompt LLMs step by step to generate multiple pseudo QA pairs with background passages and explanations entirely from scratch. These generated elements are then utilized for in-context learning. Experimental results show that our method significantly surpasses previous state-of-the-art zero-shot methods on three widely-used ODQA datasets and even achieves comparable performance with various customized fine-tuned models on full training data. Our code is available at https://github.com/lockon-n/self-prompting.

1 Introduction

Open-Domain Question Answering (ODQA) is a longstanding task in natural language processing that aims to answer questions about world knowledge without explicitly providing specific background documents (Voorhees et al., 1999; Huang et al., 2020; Zhu et al., 2021; Zhang et al., 2022a). The most common approach for ODQA is to train a Retriever-Reader pipeline (Chen et al., 2017): (i) first retrieving the most related documents of the question, (ii) then applying the reader to extract or generate the final answer conditioned on these documents (Karpukhin et al., 2020; Lewis et al., 2020; Izacard and Grave, 2021). Despite the decent performance, they rely on large amount of training data and a large external knowledge corpus, so it is hard to expand these methods to zero-shot ODQA where no training data is available.

With the emergence of Large Langauge Models (LLMs) such as GPT-3 (Brown et al., 2020), OPT (Zhang et al., 2022b), PaLM (Chowdhery et al., 2022), InstructGPT (Ouyang et al., 2022), researchers start to leverage them for zero-shot ODQA tasks. These LLMs are capable of generating correct answers without any training data and external corpus with direct prompts (like *Q: {question} A:*). Some recent works go further to induce the instruction understanding abilities and use the abundant knowledge in their parameters by asking LLMs to generate rationales called *chain-of-thought* (Wei et al., 2022b; Kojima et al., 2022) or related background information (Yu et al., 2022; Sun et al., 2022) before answering. However, these naive prompting methods for zero-shot ODQA still fail to compete with customized fine-tuned models (Yu et al., 2022), since they fail to take full advantage of the powerfulness of LLMs.

In this paper, we focus on zero-shot ODQA with no training data and external corpus. Based on the characteristics of this task, we propose **Self-Prompting** the LLM to explicitly activate different capabilities of LLMs and combine these capabilities to improve performance. In general, we ask LLMs to automatically generate pseudo QA pairs for in-context learning instead of predicting the answers directly as in naive prompting methods. Self-prompting consists of two stages: preparation and inference. In the preparation stage, the LLM is required to generate a pseudo QA dataset in three steps: (i) write a short Wikipedia-style passage, extract named entities in this passage as answers;

* Corresponding authors. This paper was partially supported by Joint Research Project of Yangtze River Delta Science and Technology Innovation Community (No. 2022CSJGG1400).

Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), pages 296–310
June 16-21, 2024

(ii) raise questions for the answers; (iii) 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 effectively conducted with simple natural language prompts. In the inference stage, we propose a novel clustering-based retrieval method to select salient 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 as the input sequence, which is fed into the LLM to get the final answer.

We evaluate our approach on three ODQA benchmarks: WebQ (Berant et al., 2013), NQ (Kwiatkowski et al., 2019), and TriviaQA (Joshi et al., 2017). Experimental results show that our method significantly surpasses the direct prompting baselines and the previous state-of-the-art (SOTA) zero-shot method *GENREAD* (Yu et al., 2022). Our method even shows comparable performance with strong few-shot methods. We also conduct extensive analysis on the effects of different generated components, like the formats of the prompted sequence for in-context learning, ways of demonstration selection from the pseudo dataset, and the quality of generated QAs. In general, our contributions can be summarized as follows:

(i) We propose Self-Prompting to leverage multiple capabilities of LLMs for zero-shot ODQA. It can automatically build a pseudo but high-quality ODQA dataset in advance and select salient QA instances for in-context learning.

(ii) We propose a clustering-based retrieving method to effectively utilize the built pseudo QA dataset to select both semantically similar and diverse examples for each test sample.

(iii) We conduct extensive experiments to show the effectiveness of Self-Prompting on three ODQA tasks. It achieves new SOTA for zero-shot ODQA, and is comparable with some fine-tuned Retriever-Reader models and few-shot prompting methods.

2 Related Works

Retriever-Reader Models for ODQA The mainstream method to tackle ODQA tasks is the Retriever-Reader architecture (Zhang et al., 2023). It first leverages a retriever over a large knowledge corpus to select several related documents that may contain the answer. Then, a reader is used to process the retrieved documents and predict the 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). There are also two choices for the reader: extractive readers like BERT (Devlin et al., 2019) or generative ones like T5 (Raffel et al., 2020). A related work in this branch is PAQ (Lewis et al., 2021b), which generates 65 million probably-asked questions based on the complete dump of Wikipedia and directly retrieves these questions instead of documents.

LLM and In-context Learning Generally, LLMs refer to the pre-trained models with tens or hundreds of billions of parameters, such as GPT-3, 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 investigated 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 generate a large number of demonstrations used in in-context learning completely by the LLM itself from scratch.

Enhancing Models with LLM generation A recent line of research aims to use the generated contents of LLMs to facilitate the training of small models. For example, Ye et al. (2022) uses GPT-2 (Radford et al., 2019) to generate pseudo data to train tiny language models, and Wang et al. (2022) distills the knowledge of GPT-3 into GPT-2 for commonsense QAs. Another line of work directly uses contents generated by the LLM itself. Some works use LLMs to generate relevant contexts or background documents, which are used as additional inputs when answering questions (Liu et al., 2022b; Yu et al., 2022; Sun 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).

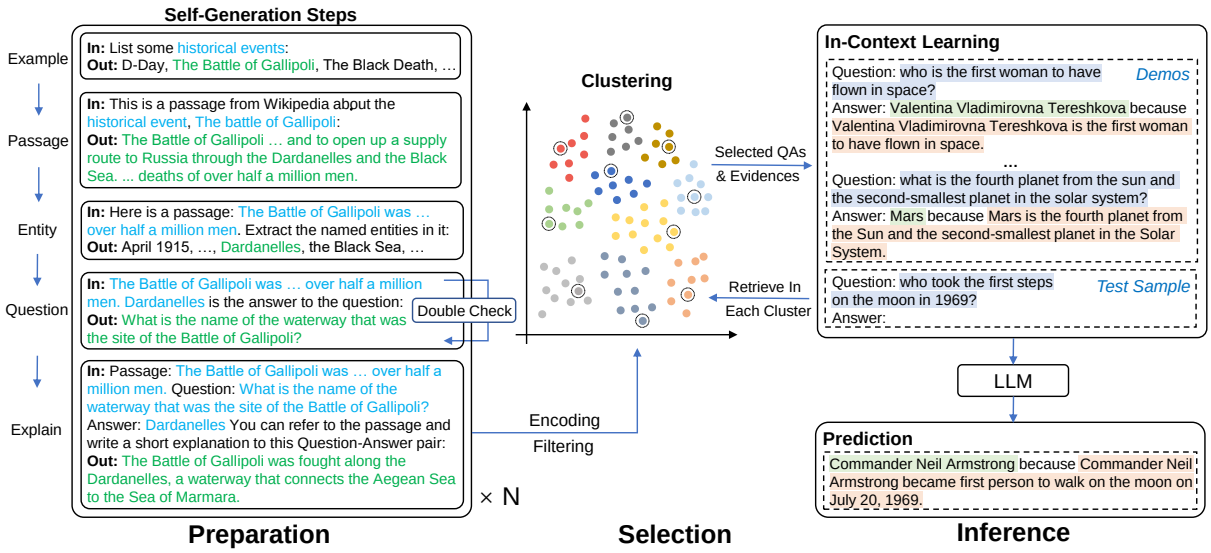


Figure 1: The overall framework for Self-Prompting on ODQA. In the self-generation steps, blue refers to contents generated in previous steps or manually designed topics, and green is the newly generated texts. In inference, question, answer, and explanation are question, answer, and explanation for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

3 Approach

This section presents the details of Self-Prompting, which has 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 examples in the pool through a clustering-based retrieval method as in-context demonstrations to help understand and answer the given question. An overall framework is shown in Figure 1.

3.1 Qseudo QA Dataset Generation

In the preparation stage, we prompt the LLM to automatically generate a large number of QA pairs as a pseudo dataset by the following four steps.

Passage Generation To ensure the diversity of the generated passages, we first sample various topics (such as *countries, books, tourist attractions*) that are likely to appear in ODQA based on the distribution of hierarchical WordNet synsets for entities appearing in the answers of TriviaQA (Joshi et al., 2017). Specifically, we first abstract and summarize these entities to define a set of high-level themes such as politics, sports, geography, film, television, etc. Then under each high-level theme, we further subdivide it into several topics; for instance, the sports theme is broken down into athletes, sports teams, and sporting events, leading to a complete list of topics. Regarding the number of examples needed for each topic, we empirically set the requirement for each high-level theme at 100 and evenly distribute this value among all the subordinate topics. Subsequently, we manually adjust these numbers, resulting in the data shown in our paper (Table 9 in Appendix B).

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 it. Through this, we obtain a huge number of examples covering different categories and they are leveraged to generate short Wikipedia-style passages with the following prompt "`This is a passage from Wikipedia about the {topic}, {example}:`". Note that we do not require the generated article to be an exact copy of the Wikipedia page but just a Wikipedia-style paragraph with enough factual information.

Named Entity Recognition We extract the named entities in these generated passages as the candidate answers. Usually, this step is conducted by a fine-tuned small model. In our framework, this is also done by the LLM itself. For a given 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 (e.g., date, name, and location) extracted in the previous step are used as the candidate answers. We then ask the LLM to write 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 double-check, i.e., re-answer the question based on the passage to see whether it can recover the entity, and only keep the sample when the answer predicted by the LLM has the same normalized form as the extracted entity.

Explain the QA pair For each QA pair, we ask the LLM to return a one-sentence explanation according to 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:`". This step elicits the summarization and induction skills of the LLM to provide a fine-grained annotation for the generated QA pairs without including too much redundant information in the original passage.

3.2 Dynamic In-context Demonstrations Selection for Inference

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

Clustering-based Retrieval Some previous works point out that using examples with high semantic similarity as in-context demonstrations brings benefits (Liu et al., 2022a), while others claim that a fixed set of examples based on clustering is better (Zhang et al., 2022c). We find it effective to combine these two ways. 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 encode the question using Sentence-BERT and retrieve the most similar example from each cluster with simple cosine similarity. This selection method balances the similarity and diversity of the demonstrations.

Answer then Explain The final step is to organize the 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. By doing so, the LLM can view more information than just switching to a QA mode, and it can also give out a brief explanation along with its answer. This is 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 our choice.

4 Experiments

4.1 Datasets and Settings

We conduct experiments on three English ODQA benchmarks, including WebQ (Berant et al., 2013), NQ (Kwiatkowski et al., 2019) and TriviaQA (Joshi et al., 2017). Data statistics are in Appendix A.

We use InstructGPT (Ouyang et al., 2022) (`text-davinci-002` of GPT-3 (Brown et al., 2020)) as the LLM following previous works (Wei et al., 2022b; Kojima et al., 2022; Yu et al., 2022). We also conduct experiments on Codex (Chen et al., 2021) (`code-davinci-002` of GPT-3 for code generation) to check the versatility of Self-Prompting. 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 B. 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 questions that are ambiguous without context (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 outputs an explanation sentence with no target answer span in it. For each passage, the upper limit of generating QA pairs is 10 to ensure diversity. Detailed parameters like `max_tokens` or `temperature` and prompt templates for LLM generation in each step are in Appendix C. We also provide a brief cost and running efficiency analysis in Appendix H. Following previous works on ODQA, we choose Exact Match (EM) as the evaluation metric, 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 heuristically perform extra post-processing on WebQ to extract only the first entity 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), Codex (Chen et al., 2021), GENREAD (Yu et al., 2022), RECITE (Sun et al., 2022); retrieval-augmented LLM prompting: DPR+InstructGPT,

Table 1: 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 parameters in this system (e.g., RAG uses 2 BERT-base with 110M×2 and 1 BART-large with 400M). Train Data refers to whether the system uses training data for training or as in-context demonstrations, and External Corpus means whether an external knowledge corpus is used to retrieve documents. ⋆This value is obtained by 5-shot in-context learning with samples from the training set as demonstrations. †This value is obtained on a subset of TriviaQA used in RECITE to save computational cost.

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 (RECITE is few-shot prompting)*							
InstructGPT	175B	✗	✗	18.6	20.9	52.6	30.7
Codex	175B	✗	✗	25.4	27.1	81.8†⋆	-
GENREAD (InstructGPT) (Yu et al., 2022)	175B	✗	✗	24.8	28.2	59.3	37.4
RECITE (Codex) (Sun et al., 2022)	175B	✓	✗	-	35.8	83.5†	-
**our method, self prompting by generating pseudo QA for in-context learning*							
Self-Prompting (InstructGPT)	175B	✗	✗	35.6	36.2	66.8/79.4†	46.2
Self-Prompting (Codex)	175B	✗	✗	38.9	40.7	84.3†	-

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 1. Compared to direct prompting, our Self-Prompting method surpasses the InstructGPT baseline by +15.5 EM on average and the previous SOTA method GENREAD by +8.8 EM on average. A similar improvement is also observed when using Codex as the backbone model, and it even achieves a higher score than RECITE, a strong few-shot baseline.

Self-Prompting yields comparable results to various powerful retrieval-augmented fine-tuned models with regard to the average EM on three datasets, showing that the LLM itself has stored enough world knowledge in its parameters and provides the potential as an implicit corpus for retrieving. We also 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 the LLM, showing the great potential of LLMs for ODQA under a zero-shot setting.

Although the generated QA data is not 100% perfect (see Sec 5.6), using them for in-context learning still improves the model performance, which is consistent with the conclusion in Min et al. (2022) where they demonstrate that the label space, the distribution of the input text, and the overall format of the sequence are key factors for in-context learning rather than absolute accuracy. Therefore, Self-Prompting can reasonably use the existing capabilities of LLMs to construct data with a similar distribution, and it improves model performance in an in-context learning way as expected.

5 Analysis

This section will present our analysis on the construction of in-context learning demonstrations, hyper-parameters in Self-Prompting, the generality of Self-Prompting on different LLMs, and quality analysis of the generated pseudo data. As LLMs like GPT-3 have paywalls, our analysis is conducted on the subsets of the three datasets by randomly selecting 1,000 samples from the test sets. Unless otherwise stated, we use InstructGPT in this section for these three subsets. Besides EM, we also report the F1 score because it is a more robust metric for tiny formal differences between the prediction and gold answer.

Table 2: Effects of using different formats for in-context learning in the inference stage. Q = question, A = answer, E = explanation sentence, P = passage. We report the average EM and F1 on three datasets.

Demos	Pred.	EM	F1
**one iteration*			
QAE	Q→AE	**48.2**	**58.3**
QAP	Q→AP	46.9	57.5
QEA	Q→EA	43.3	53.6
QPA	Q→PA	40.3	49.9
QAEP	Q→AEP	46.9	57.3
QAPE	Q→APE	46.4	57.1
**two iterations*			
1: QP	Q→P	-	-
2: PQA	PQ→A	41.5	51.8
1: QE	Q→E	-	-
2: EQA	EQ→A	43.7	54.7

5.1 How to Use Generated Passages and Explanations

A crucial question is how to use the byproducts, namely the passages and explanations, in 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 formats in Table 2. The *one iteration* methods mean only one call of API is needed to generate the answer and the passage/explanation, while in *two iterations* methods the LLM needs to generate the passage/explanation first and use them in the second API call. Detailed templates for these methods are in Appendix E. From Table 2, we see that the QAE format is the most effective, while the other three answer-then-explain formats (QAP, QAEP, and QAPE) are worse than QAE, which indicates that ntroducing too much redundant information in the form of passage is harmful to the LLM. We also find that the two *chain-of-thought* style formats, QEA and QPA, are much worse than other variants. In Lu et al. (2022a) and Wei et al. (2022b), similar findings are concluded that *chain-of-thought* is beneficial to complex reasoning tasks like math problems but has little impact on commonsense questions. Finally, the *two iterations* methods lead to a significant 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 generated by the LLM is much larger than the number of examples we can put in the input sequence for in-context learning, a proper selection strategy is necessary. We conduct experiments with four settings: randomly selection (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 3 show that random selection performs the worst and suffers from instability. The Retrieve and ClusterCenter methods have some improvements over Random. RetrieveInCluster, proposed to select diverse and semantically similar demonstrations, is robust enough to achieve satisfactory scores among all datasets.

Table 3: Different ways for demo selection. We report mean value and standard deviation on 5 runs of Random. The metrics are average EM and F1 on three datasets.

Selection	EM	F1
Random	$46.4_{\pm 1.3}$	$56.7_{\pm 1.0}$
Retrieve	46.7	57.1
ClusterCenter	47.1	57.2
RetrieveInCluster	**48.2**	**58.3**

Table 4: Performance of Self-Prompting (Codex) on different numbers of generated QA pairs.

Data Size	100	500	1,000	2,000	full
EM	51.8	51.7	52.2	52.4	**52.6**

5.3 Different Number of Demonstrations

A natural idea of in-context learning is to put as many examples as possible in the input, so we also investigate the effect of different numbers of demonstrations, as in Figure 2. We report the av-

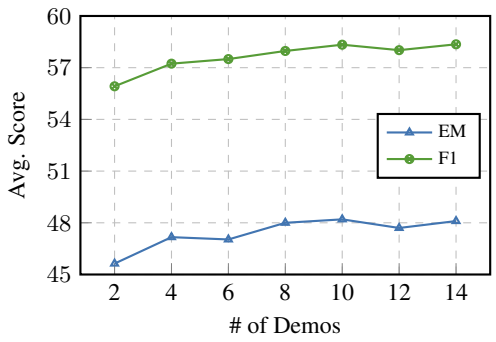


Figure 2: Average EM / F1 when using different numbers of demonstrations in Self-Prompting.

Table 5: The performance of our method for different LLMs. **Direct** means the models are required to output answers directly with no demonstrations, **Self-P** (**Self-Prompting**) uses the self-generated QAs as in-context learning samples.

Model	#Params	Method	WebQ			NQ			TriviaQA			Avg.		
			EM	F1	IE	EM	F1	IE	EM	F1	IE	EM	F1	IE
InstructGPT	175B	Direct	25.3	38.7	52.6	21.6	31.9	38.9	55.1	64.8	72.0	34.0	45.1	54.5
		Self-P	38.7	51.6	61.4	37.2	47.6	48.3	68.7	75.7	78.2	48.2	58.3	62.6
Codex	175B	Direct	27.6	44.5	55.9	27.1	38.0	45.8	67.6	73.6	76.3	40.8	52.1	59.3
		Self-P	44.0	55.8	64.6	41.3	51.7	51.2	72.4	79.5	81.8	52.6	62.3	65.9
GPT-NeoX	20B	Direct	14.9	27.7	38.8	5.9	12.3	28.7	19.1	25.8	37.8	13.3	21.9	35.1
		Self-P	27.6	40.6	41.7	19.2	26.5	26.9	38.3	43.8	46.4	28.4	37.0	38.3
Alpaca	7B	Direct	16.4	32.9	46.0	11.4	21.3	29.7	29.3	41.5	51.4	19.0	31.9	42.4
		Self-P	25.8	40.3	50.0	20.3	30.0	32.6	44.3	52.8	57.9	30.1	41.0	46.8

Table 6: Two examples in the pseudo QA dataset. teals are the key sentences for the questions and blues are the entities used as answers.

Topic: Video game
Example: Splatoon
Passage: Splatoon is a third-person shooter video game developed and published by Nintendo for the Wii U, which was released worldwide in May 2015. The game centers around characters known as Inklings—anthropomorphic squid that can transform between humanoid and cephalopod forms, and use colored ink to attack opponents and clear goals. Splatoon features several game modes, including four-on-four online multiplayer and a single player campaign.
Entity: the Wii U
Question: On what platform is Splatoon available?
Explanation: Splatoon is a third-person shooter video game developed and published by Nintendo for the Wii U.

Topic: Historical figure
Example: Augustus Caesar
Passage: Augustus Caesar (63 BC – 14 AD) was the first Roman emperor, ruling from 27 BC until his death in 14 AD. He was born Gaius Octavius Thurinus into an old and wealthy equestrian branch of the plebeian gens Octavia. His maternal great-uncle Julius Caesar was assassinated in 44 BC, and Octavius was named in Caesar's will as his adopted son and heir, then known as Octavianus (Anglicized as Octavian). He, Mark Antony, and Marcus Lepidus formed the Second Triumvirate to defeat the assassins of Caesar. Following their victory at Philippi, the Triumvirate divided the Roman Republic among themselves and ruled as military dictators. The Triumvirate was eventually torn apart by the competing ambitions of its members. Lepidus was driven into exile and stripped of his position, and Antony committed suicide following his defeat at the Battle of Actium by Octavian in 31 BC.
Entity: 31 BC
Question: In which year did Octavian defeat Antony at the Battle of Actium?
Explanation: The Battle of Actium took place in 31 BC, and was the decisive battle of the Final War of the Roman Republic.

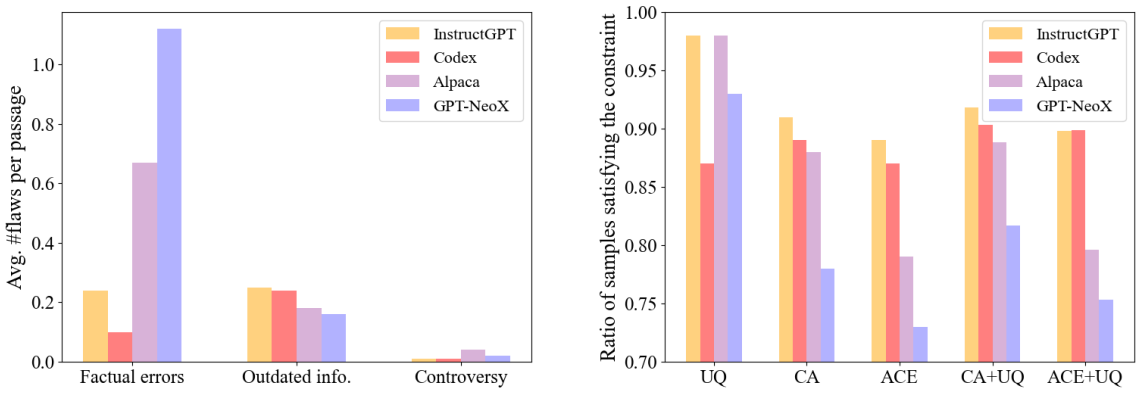
erage EM and F1 over the three datasets. When the number ranges from 2 to 10, the performance generally becomes better as the number increases, and using more than 10 examples does not bring significant improvements. As a result, we choose 10 demonstrations in our main experiments for both performance and cost considerations.

5.4 Impact of the size of generated pseudo data for Self-Prompting

In our preliminary experiments, we find that Self-Prompting can achieve strong performance with substantially fewer generated examples than our current data size of about 5,000. We report the average EM on the three datasets used for analysis in Table 4 for Codex. While we chose the maximum data size for our main experiments to demonstrate the peak capabilities, these results highlight that comparable gains can also be achieved with good data efficiency and greatly reduced overhead.

5.5 General Effectiveness with LLMs of Different Sizes

Besides InstructGPT and Codex, we also try two smaller LLMs to see how Self-Prompting performs on different sizes of models: GPT-NeoX (20B) (Black et al., 2022) and Alpaca (7B) (Taori et al., 2023). Since the performance of large language models on ODQA is highly underestimated with EM and F1 (Kamalloo et al., 2023), we also report the Instruct-Eval (IE) (Kamalloo et al., 2023) score with GPT-4 (OpenAI, 2023a) (`gpt-4-1106-preview`) as the evaluator model. The results in Table 5 show that for LLMs with



(a) Average # of flaws in a passage.

(b) Quality of the question, answer and explanation.

Figure 3: Quantitative study of the generated data. In (b), UQ = Unambiguous question, CA = Correct answer, ACE = Explanation is both accurate and helpful, CA+UQ and ACE+UQ = ratio of CA and ACE given UQ, respectively. In (a) lower value indicates a higher quality, while in (d), a higher value is better.

parameters of different orders of magnitude, Self-Prompting boosts their performance significantly on all three datasets. It also narrows the performance gap between different LLMs, even if the gap is large under direct prompting.

5.6 Data Generation Quality Analysis

To further explore the quality of the generated pseudo data, we display two examples in Table 6 as a case study, with key sentences in teal and answer entities in blue. The questions generated for extract entities are proper and answerable with no context given, which is in line with ODQA. As a key part of Self-Prompting, we see that the explanations written by the LLM are reasonable. In the first example, the LLM precisely extracts the key sentence from the passage, while in the second, the LLM not only identifies relevant information but also adds effective explanations.

We also conduct a quantitative study to detect the hallucination issue (Ji et al., 2023) in the generated contents. We collect the data generated by all four LLMs used in Sec 5.5 and manually selected 100 samples (Q, A, P, E) from each of these four sources with the same topics (e.g., passages and QAs generated by different models but all about *The Legend of Zelda*) for fair comparison. For each sample, we use GPT-4 (OpenAI, 2023b) based New Bing[1] to check: (i) the number of factual errors, outdated information and controversial opinions in the generated passage; (ii) if the question is unambiguous without context; (iii) if the answer is correct for the question; (iv) if the explanation is both accurate and helpful for question answering.

All LLMs inevitably generate outdated information, as they cannot access the Internet and are limited by training data. The two smaller models, Alpaca and GPT-NeoX, make significantly more factual mistakes when they generate the required passages, as shown in Figure 3(a) while InstructGPT and Codex only have one factual error in every 5 and 10 passages respectively. From Figure 3(b), we see that (question, answer, explanation) triples by InstructGPT are better than those from other models on almost all concerned metrics, showing its strong instruction understanding ability.

Table 7: Comparison between Self-Prompting and in-context learning with training data (Traindata-ICL). We report average EM/F1 on the non-overlapping subsets.

Method	EM	F1
Traindata-ICL	36.0	48.7
Self-Prompting	34.2	46.0

5.7 Comparison between Self-Prompting and Using Training Data

Since the generated pseudo data is not flawless, it is natural to see the gap between Self-Prompting and in-context learning with real training data as a strong baseline (Traindata-ICL). To avoid the test-train data overlap problem, we use the non-overlapping subsets annotated in Lewis et al. (2021a) (425 WebQ samples, 357 NQ samples, and 254 TriviaQA samples) to conduct experiments in this section. Since there are no explanations in the training data, we only use a simple *question-answer* format. Other procedures and

[1] https://bing.com/chat

hyper-parameters, like the clustering-based selection and 10 in-context demonstrations, are kept the same. We report the average EM and F1 on the three datasets. The results in Table 7 show that even though the size of available demonstrations is much larger (e.g., 70K training data for NQ v.s. 4.8K pseudo data by Self-Prompting), Self-Prompting still obtains comparable performance with using training data for in-context learning (performance gap less than 2 EM and 3 F1). This indicates that even without laborious and costly human annotation, the LLMs can still effectively elicit their own knowledge and answer open-domain questions.

6 Conclusion

In this paper, we propose Self-Prompting LLMs for Zero-Shot Open-Domain Question Answering. Our method induces the LLM to generate pseudo QA pairs with matching passages and explanations and use them for in-context learning. It successfully stimulates the potential of the LLM in ODQA by explicitly activating various language understanding abilities and eliciting knowledge in its parameters. With no training data and external corpus, Self-Prompting surpasses previous SOTA significantly and performs on par with several strong retrieval-augmented fine-tuned models and few-shot prompting methods. It is generally effective across LLMs under different sizes and is able to provide accurate and helpful explanations alongside the answers, making it a more reliable and interpretable framework for turstworthy AI systems.

7 Limitations

In this paper, various experiments rely on the OpenAI APIs, whose usage needs to be paid and may cause a huge cost when generating a large number of tokens. Therefore, this may affect the reproducibility of this work. We also note that the pseudo QA generating process needs some trial and error to find good prompt templates for each step as shown in Table 14 in Appendix C, so there is still room to reduce the human effort in the whole process.

8 Ethics

In this work, we use three ODQA datasets for experiments. The NQ and TriviaQA datasets use the Apache-2.0 code, and the license for WebQ is not specified. In Self-Prompting, any suggested prompt templates refrain from gathering or employing personal information concerning other individuals. The specific prompt templates employed can be found in Appendix C and E. Notably, all prompt templates in this paper are devoid of any discriminatory language targeting individuals or groups, and they do not pose any potential harm to the safety of others.

We also recognize the importance of considering the potential societal impacts of the synthesized data. To address this, we have primarily utilized the OpenAI Moderation API[2] to detect if Self-Prompting generated content is free from harmful material. We find that all 977 passages and 4,883 question-answer-explanation triples revealed no toxic content. While our current synthetic data is free from toxicity, we acknowledge the need for safeguards against potential risks in larger-scale generation, especially for contents that are not toxic but contain explicit or implicit bias. Introducing measures such as the OpenAI Moderation API to detect and halt toxic outputs, setting rules in the prompts for LLM behavior (Bai et al., 2022), and employing controllable decoding strategies, from token bans to sophisticated methods that dynamically shift the output token distribution (Li et al., 2023; Liu et al., 2021), are possible solutions to manage this issue.

References

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al. 2022. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*.

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.

Sid Black, Stella Biderman, Eric Hallahan, Quentin Anthony, Leo Gao, Laurence Golding, Horace He, Connor Leahy, Kyle McDonell, Jason Phang, Michael Pieler, USVSN Sai Prashanth, Shivanshu Purohit, Laria Reynolds, Jonathan Tow, Ben Wang, and Samuel Weinbach. 2022. Gpt-neox-20b: An open-source autoregressive language model. *arXiv preprint arXiv:2204.06745*.

[2]https://platform.openai.com/docs/guides/moderation

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.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. 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.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. 2023. Survey of hallucination in natural language generation. *ACM Computing Surveys*, 55(12):1–38.

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.

Ehsan Kamalloo, Nouha Dziri, Charles Clarke, and Davood Rafiei. 2023. Evaluating open-domain question answering in the era of large language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5591–5606, Toronto, 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, Pontus Stenetorp, and Sebastian Riedel. 2021a. Question and answer test-train overlap in open-domain question answering datasets. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 1000–1008, Online. Association for Computational Linguistics.

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

Yuhui Li, Fangyun Wei, Jinjing Zhao, Chao Zhang, and Hongyang Zhang. 2023. Rain: Your language models can align themselves without finetuning. *arXiv preprint arXiv:2309.07124*.

Alisa Liu, Maarten Sap, Ximing Lu, Swabha Swayamdipta, Chandra Bhagavatula, Noah A Smith, and Yejin Choi. 2021. Dexperts: Decoding-time controlled text generation with experts and anti-experts. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 6691–6706.

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.

Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2022. Rethinking the role of demonstrations: What makes in-context learning work? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 11048–11064, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

OpenAI. 2023a. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

OpenAI. 2023b. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

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.

Zhiqing Sun, Xuezhi Wang, Yi Tay, Yiming Yang, and Denny Zhou. 2022. Recitation-augmented language models. *arXiv preprint arXiv:2210.01296*.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford alpaca: An instruction-following llama model. https://github.com/tatsu-lab/stanford_alpaca.

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*.

Qin Zhang, Shangsi Chen, Dongkuan Xu, Qingqing Cao, Xiaojun Chen, Trevor Cohn, and Meng Fang. 2023. A survey for efficient open domain question answering. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14447–14465, Toronto, Canada. Association for Computational Linguistics.

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*.

Datasets	Size	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 8: Statistics for the test split of 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.

A Statistics of the datasets

The statistics of the datasets we use in Table 8. We find that test samples in TriviaQA have much longer question length than the other two datasets. TriviaQA also provides more reference answer, so models tend to have a higher score on this dataset.

B Names and Required Number of Examples for Topics

We list the name and required number of examples for each topic in Table 9. In general, we require about 100 examples for each high-level category (e.g., *Art, History, Sports*).

C Details in Generating Pseudo QA Dataset

We present the exact templates and API parameters used in self-generation steps in Table 14. After getting the passage, we use the NLTK toolkit[3] to remove the last sentence of it if the last sentence is incomplete or truncated.

D Statistics of the Generated Pseudo QA datasets

We show some statistics of the generated pseudo QA dataset in Table 10.

E Specific Templates in Different Formats

The specific templates used in main experiments (QAE) and in Sec 5.1 are shown in Table 15. We refer to Yu et al. (2022) to design them. The `temperature` is 0 for all formats.

F Training Small Models with the Pseudo QA dataset

We investigate if the built pseudo ODQA dataset can be used to train a specialized smaller model. To implement it, we retrain the reader of a RAG

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

Table 9: Names and required examples for the 29 manually designed topics.

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 10: Statistics of the generated pseudo QA dataset.

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 11: The indexing and searching time for Self-Prompting.

Device	Indexing		Search time (1,000 queries)	
	Encoding pseudo data	Clustering	Encoding queries	Cluster-based Retrieval
CPU	25.33s	1.86s	4.20s	3.21s
1 GPU	5.82s	1.69s	0.30s	1.57s

Table 12: The performance of two different RAG models on the NQ subset.

Model	EM	F1
Pre-trained RAG	36.10	44.01
RAG, reader retrained	19.00	27.44

Table 13: Comparison between Self-Prompting and a real-time generation variant.

Method	EM	F1
Self-Prompting	37.2	47.6
Real-time Generation Variant	35.3	45.5

(Lewis et al., 2020) pre-trained on the full NQ trining set[4] on our pseudo data from BART-large, while fixing the retriever module (we are unable to jointly train both modules as the original RAG did with our synthetic data). The EM and F1 on the NQ subset used in the analysis section are in Table 12.

This huge performance gap is likely caused by: 1) The pre-trained model benefits from more in-domain examples (70K+ NQ training examples vs 5K pseudo data). There is also significant train-test overlap in NQ as reported in Lewis et al. (2021a), 2) Retraining only the reader while fixing the retriever causes a mismatch, and 3) The reader is trained to only process one passage at a time, instead of comparing multiple passages simultaneously as the pre-trained reader. Given these results, we conclude that in current setting, it is a better practice to use the generated data for in-context learning samples of LLM itself than training a smaller supervised model.

G Real-time Pseudo Data Generation Variant for Self-Prompting

We also investigate the potential of a real-time generation variant of Self-Prompting. Specifically, for each test sample, we first ask the LLM to identify the main entity in the question and then generate a short passage based on it. The rest generation steps are all kept the same. In short, we reinforce the relevance of in-context demonstrations but weaken the diversity. We show the performance of this variant on the NQ subset in the analysis section.

The results in Table 13 show that this real-time variant does not bring improvement over Self-Prompting. This aligns with our earlier finding that cluster-based selection outperforms global similarity retrieval (in Table 3), indicating diversity is important for effective prompting. While real-time generation of highly-related examples is promising intuitively, it seems that some variability in the prompts is beneficial to avoid overfitting to a narrow context. Additionally, the higher inference delay and cost of real-time generation make it less practical. Overall, these results emphasize the balance of relevance and diversity when constructing demonstrations, and some offline preparation can be more efficient.

H Cost and Running Efficiency

We provide a brief analysis of the cost and running efficiency of Self-Prompting. The total cost for constructing the pseudo QA dataset is about $120 and it takes about 6 hours to collect the data.

The indexing and search time over the pseudo data in Table 11. As a comparison, the encoding/indexing/search time for DPR (Karpukhin et al., 2020) (statistics from their original paper) is 8.8h for encoding the complete Wikipedia dump on an 8-GPU machine, 8.5h for indexing the Wikipedia passage embeddings, and the search rate is 995 queries per second.

I More Data Quality Analysis

We provide more statistics for the quality analysis on our generated pseudo data in Figure 4 as a supplement to Sec 5.6, including the average number of flaws in the generated passages normalized by passage length, and the ratio of passages with at least one flaw. Similar to Figure 3, they also show the superiority of the OpenAI models over Alpaca and GPT-NeoX.

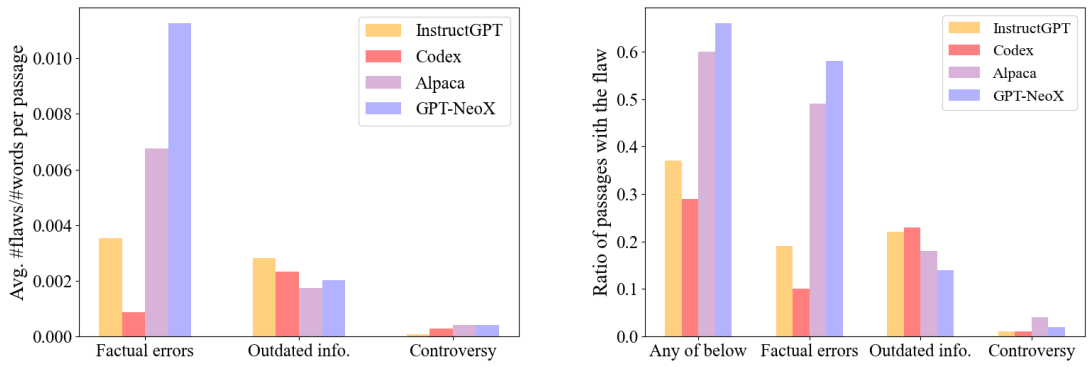
[4] https://huggingface.co/facebook/rag-sequence-nq

Table 14: Parameters and detailed prompts used in each generation step, the temperatures are all set as 0 (except generating diverse examples for a certain topic) to increase the correctness of the generated content at each step.

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 15: Specific templates for different input formats and the parameters of calling APIs.

Format	Template	max_tokens
**one iteration*		
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



(a) Same to Figure 3(a), but normalized by passage length.

(b) Ratio of passages with specific kind of flaws.

Figure 4: More statistics of quality analysis for our generated pseudo data.