Engineering the Best In-Context Input for GPT-3 in the OpenQA Task

Kaili Huang
Computer Science
kaihuang@stanford.edu

Gurdeep Sullan
Computer Science
sullan@stanford.edu

Ofure Ebhomielen
Computer Science
ofure@stanford.edu

Abstract

GPT-3, since its release, has garnered the attention of the NLP community due to its versatility across a wide range of NLP tasks. In this work, we use GPT-3 to approach the OpenQA task, where the model needs to answer input questions without being given the golden passage that contains the correct answer. We work on engineering the GPT-3 input to improve its performance on the OpenQA task in the in-context learning setting. We propose three different in-context learning example selection strategies that are based on lexical, syntactic and semantic similarities. We also propose two example ordering mechanisms: random order and reverse order. Overall, we found that: (1) The semantic similarity-based model with reversely ordered examples achieves highest performance, with an F1 score of 0.5738 on the dev set and 0.4949 on the test set. (2) The lexical and syntactic similarity-based models both outperform the baseline model. (3) The number of examples (either too many or too few) affects the model's performance.

1 Introduction

Between 2017 and 2019, there was a rise in the adoption of the "*pre-train and fine-tune*" paradigm for solving NLP tasks. In this paradigm, a Language Model (LM) with a fixed architecture is pre-trained on large datasets, and in the process, learns some robust general-purpose features of the language that it is modeling. The pre-trained LM is then adapted to fit several downstream tasks by fine-tuning it using task-specific objective functions.

Currently, there is a new wave in which this "pre-train and fine-tune" paradigm is being replaced by one popularly called "*pretrain, prompt and predict*" (Liu et al., 2021). In this new paradigm, rather than using objective engineering to adapt a LM to downstream tasks, a textual prompt is used to reformulate downstream tasks to look like those solved during the original ML training. For example, the input: "`[passage]. According to the passage, [question] [answer]`" requires the LM to fill the blank with the correct `[answer]` to the input `[question]` from the given `[passage]`. By selecting appropriate inputs, we can guide the pre-trained LM to predict the desired output and often without additional task-specific training (Radford et al., 2019; Raffel et al., 2019; Brown et al., 2020).

In addition to making the pre-training and fine-tuning tasks more similar, this approach enables users explain the task to the model, making it easier for the model to understand the task. Hence, given a number of appropriate examples and prompts as inputs, an unsupervised pre-trained LM can be used to solve a lot of different downstream tasks, and this is referred to as *in-context learning* (Xie et al., 2021). This then necessitates input engineering i.e. designing a good input that allows a LM to solve the given task. For this project, we have access to a substantial number of training documents to design the input to GPT-3.

In this paper, we apply GPT-3's power and versatility to solve in-context learning problems for the OpenQA task. Open-domain Question Answering (OpenQA) is an NLP-task that aims to answer a question in the form of natural language without a given golden passage, but with access to a large-scale unstructured documents. In particular, we use ColBERTv2, a retriever, to retrieve the most relevant passage from the collection of SQuAD train set passages. And we propose different strategies (syntactic, lexical, semantic similarity-based models and their combination, etc.) to select similar train set examples. Then, we combine the examples, the passage and the input question to create the in-context input for GPT-3. We evaluate the performance of GPT-3 on different designs of in-context inputs. Our main goal is to understand how to better select in-context examples that work well with GPT-3's in-context learning capabilities. Besides that, we also explore the effect of different example numbers and task descriptions.

2 Related Work

2.1 Retriever

For the in-context learning task, the gold passage which contains the answer is not provided to the system. Therefore, a retriever is responsible for retrieving a series of relevant documents from a given collection of documents to obtain the final answer.

Classical information retreival (IR) mechanisms assign terms and documents sparse representations. For example, DrQA (Chen et al., 2017) applies TF-IDF matching to search over Wikipedia, and BERTserini (Yang et al., 2019) uses BM25 as the ranking function.

In recent years, the Dense Retriever has evolved rapidly, which learns distributed representations that generally captures more meaningful semantic information. Many dense retrievers encode queries and documents into independent single-vector presentations and then compute relevance score. For example, ORQA (Lee et al., 2019) utilizes independent BERT-based encoders (Devlin et al., 2018) to encode questions and documents respectively and calculates the relevance score using the inner product. However, this type of models suffer from only capturing the shallow interactions between queries and documents. On the other hand, modeling the token-level interaction between question-document pairs can be overly expensive. An effective tradeoff is to combine the benefits of two.

ColBERTv2 (Santhanam et al., 2021), for example, is proposed to both boosts retrival quality of multi-vector models and improves efficiency. Akin to ColBERT (Khattab and Zaharia, 2020), ColBERTv2 encodes both passage d query q into multi-vector representations. The passages are encoded during offline indexing, and quries are encoded at search time. The query q's similarity to a passage d is then captured by:

$$S_{q,d} = \sum_{i=1}^{N} \max_{j=1}^{M} Q_i \cdot D_j^T$$

To build a simple, uniform supervision scheme, ColBERTv2 works on selecting challenging negatives, avoiding mislabeled positives, and eliminating penalty for retrieving false negatives. The model is trained on MC MACRO Passage Ranking and outperforms other retrievers with a highest MRR@10 (39.7 for in-domain evaluation). ColBERTv2 is also evaluated on a series of out-of-domain benchmarks. Besides, the residual representation helps reduce the space footprint by $5-8\times$.

2.2 Reader

The Reader is the main model of the system responsible for obtaining the answer to the test question. Overall, there are two main categories for the choice of the Reader model: extractive reader and generative reader.

Extractive reader assumes that the answer exists in the context. Usually, it predicts the start and end position of an answer span either from the most probable document(s) or from all retrieved documents. DrQA (Chen et al., 2017), DPR (Karpukhin et al., 2020), REALM (Guu et al., 2020), etc. follow this settings.

Generative reader, on the other hand, tries to generate the answer directly as a natural language sequence. Seq2Seq models such as Bidirectional and Auto-Regressive Transformers (BART) (Lewis et al., 2019) and the Text-To-Text Transfer Transformer (T5) (Raffel et al., 2019) are used to generate the answer (Zhu et al., 2021).

An extremely powerful generative model that has been used in the in-context learning setting with oustanding success is the GPT-3 model. Generative Pretrained Transformer 3, as its name suggests, is a recurrent generative language model. It has been pretrained on tokens from Common Crawl, WebText2, Books1, Books2, and Wikipedia (over 40GB of text data). As a result it has a large number of parameters, and has been found to work well on a variety of tasks without the need of finetuning. (Brown et al., 2020).

2.3 Prompt Generation

Generative readers like GPT-3 in the in-context learning setting require prompts with a few examples of the task at hand in order to generate the answer. In order to design the prompts, a number of different strategies have been proposed.

LM-BFF (Gao et al., 2020) provides a suite of techniques for fine-tuning language models in a few-shot manner. The scenario is fine-tuning BERT (Devlin et al., 2018) or RoBERTa (Liu et al., 2019) on a small number of examples. The paper follows the prompt-based prediction and introduces an automatic prompt generation method. For example, for a sentiment classification task, the label is

mapped into a chosen word (e.g., "great" for positive examples, "terrible" for negative examples). Then, a template (e.g., "It was great.") is appended to the original sentence to produce a full positive or negative example. Examples, together with the input to be classified, are fed into a Seq2Seq model to make a prediction. Then, akin to GPT-3's "in-context learning" paradigm, an example selection strategy is designed: randomly sample a single example from each class for each input.
KATE (Liu et al., 2022) uses K-Nearest Neighbors algorithm to search the embeddings of examples and find the closest training examples generate the GPT-3 input. The study explored the effectiveness of different encoding strategies, and found that encoders that were fine-tuned on natural language matching tasks gave better performance when used to select the in-context examples as input for the question-answering task.

3 Data & Metrics

In our work, we use the Stanford Question Answering Dataset (SQuAD 1.1) (Rajpurkar et al., 2016). SQuAD is a reading comprehension dataset, with questions generated by crowdworkers on a set of Wikipedia articles. The answer to every question is a span of text, from the corresponding reading passage. SQuAD 1.1 is made up of about 100,000 questions, with the train size amounting to 87,599 examples. The articles used in building SQuAD were partitioned into 80:10:10 for training, validation and test sets respectively.
SQuAD is not originally designed for the OpenQA task, so we modified its usage to fit in our settings. The detail is discussed in Section 4.4.
We use EM and Macro F1 scores as the quantitative metrics. As a qualitative measure, we demonstrate a few of examples generated by our proposed methods and qualitatively assess the difference in examples generated by different models.

4 Methods

As is shown in Figure 1, we adopt a Retriever-Reader pipeline to approach the in-context learning OpenQA problem. The following subsections describe the ColBERTv2 retriever, GPT-3 reader and example selection strategies, respectively, with a final subsection outlining the system development process.

4.1 Retriever

For the passage retriever, we use a ColBERTv2 retriever initially trained on MS MARCO and pre-index all the passages in SQuAD as the retriever's corpus ($\sim$100K).

4.2 Reader

For the reader, we use the OpenAI GPT-3 API [1]. Specifically, we use the Curie engine. Although OpenAI doesn't seem to reveal the parameter size of the Curie model, it's the second powerful model and strikes a good balance between the performance and our budget.
For each input question, we use different example selection strategies to select similar examples from the SQuAD train set, and construct a GPT-3 input which is a concatenation of the following elements:

1. A brief task description (optional). We run experiments to assess whether including it improves the performance or not.

2. A list of n selected SQuAD train set examples. Each example is a (question, passage, answer) tuple. We run experiments to find the best example selection strategy and the best value of n.

3. The input question.

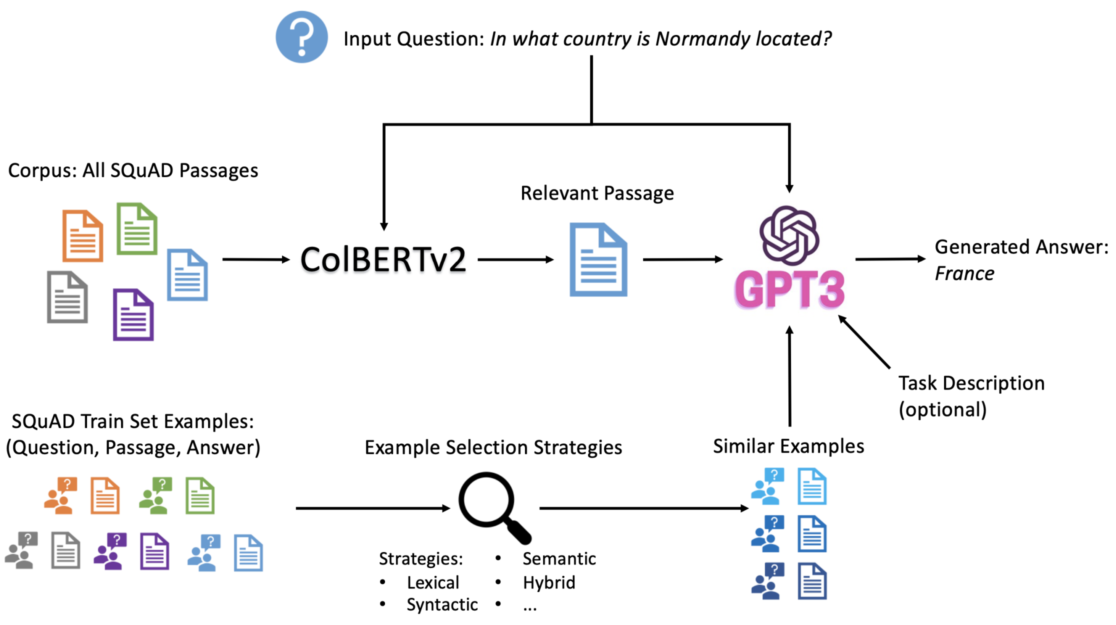
A real example of the GPT-3 input is shown in Appendix A.

4.3 Example Selection Strategies

To explore what types of examples helps GPT-3's in-context learning, we propose three example selection strategies base on different types of similarity measurement: lexical, syntactic, semantic. Each strategy aims to select examples with questions similar to the input question from a certain aspect. Lexical similarity between input question and example questions can be measured by word match, and syntactic similarity focuses on the sentence structure, which in our setting can be reflected by the question type (e.g., "when", "where", "who" questions, etc). Semantic similarity, on the other hand, focuses more on the underlying meaning of the whole sentence. A simplified example is shown below.

Input question: In what country is Normandy located?

[1]https://openai.com/api/



Input Question: *In what country is Normandy located?*
Corpus: All SQuAD Passages
Relevant Passage
ColBERTv2
GPT3
Generated Answer: *France*
Task Description (optional)
SQuAD Train Set Examples: (Question, Passage, Answer)
Example Selection Strategies
Similar Examples
Strategies:
• Lexical
• Syntactic
• Semantic
• Hybrid
• ...

Figure 1: Our Retriever-Reader Pipeline.

- Lexically similar question: Who ruled the *country* of *Normandy*?
- Syntactically similar question: *In what country* is the Cevennes?
- Semantically similar question: *Where* is Yingxiu *located*?

The following subsections describe these three strategies, together with a hybrid strategy.

4.3.1 Lexical Similarity

To measure the lexical similarity, we use TF-IDF as a similarity measurement model. TF-IDF calculates term frequency and inverse document frequency and multiplies them to compute term-document scores. We consider the example questions as "documents" and the input question as "query". With the term-document scores, we vectorize all the documents and build inverted indexing offline. For each query, we vectorize it, compute its cosine similarity to each document, and select the top similar documents to construct the GPT-3 input.

4.3.2 Syntactic Similarity

For the syntactic similarity, we naively use rule-based methods to classify questions into different types and questions within the same type are viewed as similar. Since the questions in the SQuAD dataset have fairly simple structures, a straightforward way is to pre-classify the example questions based on their first word or first few words (e.g., who, when, where, in what country, etc). Then for each input question, we classify it in the same way and select examples from the category it belongs to.

4.3.3 Semantic Similarity

In the end, to find semantically similar questions, we use sentenceBERT (Reimers and Gurevych, 2019) to encode questions and use dot product to measure the similarity between input question and example questions. We then select example questions that have top similarity scores.

4.3.4 Hybrid Strategy

Since the different example selection strategies are supposed to focus on different aspects of similarity, a combination of these different strategies have the potential to achieve overall better performance. To validate this hypothesis, we further run experiments with a hybrid strategy: for each input question, use different strategies to sample several examples and combine them to build the input.

4.4 System Development

With different example selection strategies, different choices of the example number, and different task descriptions, we build a series of retriever-reader systems. Then, we use the QA pairs of SQuAD dev set to compare the different systems we propose and select the best method. Due to budget constraints, we use a randomly sampled dev set of size 200 for development. After that, we test our best method on a held-out test set of size 300 which is randomly selected from SQuAD dev set and is disjoint from our dev set.

5 Experiments & Results

We present three sets of our main experiments and demonstrate the results. The corresponding analysis and discussion are presented in Section 6.

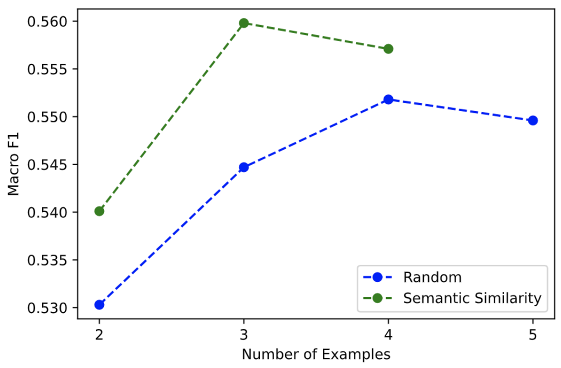
5.1 Example Selection Strategy

As described in Section 4.3, we apply different example selection strategies to retrieve relevant SQuAD train set examples for each input question. We experiment with three proposed strategies: Lexical Similarity Strategy, Syntactic Similarity Strategy, and Semantic Similarity Strategy.
We compare the EM and Macro F1 scores of these strategies with the baseline strategy (randomly samples examples for each input question).
Moreover, to further explore the effect of example ordering, we experiment different orderings for the best performing method (Semantic Similarity Strategy): random and reversely ordered, where the former randomly shuffles the top relevant examples, and the latter sorts the top relevant examples in a descending order of their similarity scores. An input with reversely ordered examples looks like:
{...3rd most similar example...}{...2nd most similar example...}{...most similar example...}{...input question...}.
With the most relevant example closest to the question, we assume it can lead to a different result from the random ordering.
For these experiments, we fix the number of examples to be 3 and don't include a task description. The experiment results are shown in Table 1.
We also experiment on the hybrid strategy but are excluding those results from our main report. The results, analysis, and the reason we exclude them are shown in Appendix B.

5.2 Number of Examples

In order to explore the effect of different numbers of examples on GPT-3's in-context learning, we experiment on the Random and Semantic Similarity Strategy with varying example numbers.
For the random strategy, we experiment with 2,3,4,5 examples. For the Semantic Similarity Strategy, we experiment with 2,3,4 examples. We randomly order selected examples for each input question, and don't include the task description for these experiments.
Table 2 shows the experiment results. Figure 2 shows the trends of Macro F1 with regard to different example numbers for the random baseline and the semantic model.

Figure 2: Performance trends for different number of Examples.



5.3 Task Description

Previous literature (Brown et al., 2020) follows a setting where a brief task description is given to GPT-3 together with a few examples. To examine the effectiveness of such task descriptions, we experiment on different wording of the description. Appendix C shows the full list of task descriptions that we experimented with.
Besides, as shown in Appendix A, we've been following a structured and less natural fashion of formatting the input examples. As a contrast to that, we also experiment with a natural language-modified input shown in Appendix D.
We fix the number of examples as 3, and randomly order selected examples for each input question. The results are shown in Table 3.

5.4 Evaluation on Test Set

Due to limited budget, we conducted all the other experiments on the dev set, and only evaluated our best model and the baseline model on the held-out test set. We use 3 examples for each input question and exclude the task description. The test result is shown in Table 4.

Selection Strategy	Example Order	EM	Macro F1
Random	Random	0.425	0.5460
Lexical Similarity	Random	0.440	0.5457
Syntactic Similarity	Random	0.425	0.5521
Semantic Similarity	Reversely Ordered	**0.460**	**0.5738**
Semantic Similarity	Random	0.450	0.5649

Table 1: Experiment results for different example selection strategies and different ordering of selected examples for each input query. The first line is the baseline model. The highest EM and F1 scores are in bold font.

#Examples	Selection Strategy	EM	Macro F1
2	Random		0.5303
3	Random		0.5447
4	Random		**0.5518**
5	Random	0.455	0.5496
2	Semantic Similarity		0.5401
3	Semantic Similarity	0.455	**0.5598**
4	Semantic Similarity	0.450	0.5571

Table 2: Experiment results for different numbers of examples selected for each input query. The highest F1 scores for each model are in bold font. Some EM scores are missing due budget constraint.

6 Analysis

We analyze the results presented in the previous section and discuss our findings.

6.1 Syntactic Model

We had hypothesized that the model run with syntactically similar examples would perform relatively well. However, as Table 1 shows, this similarity method did not perform much better than the random baseline. Upon further inspection of generated syntactically similar examples, it was apparent that the method used was perhaps overly simple. The strategy to use questions with the same first word (i.e. "Who", "What", "Where", etc.) would sometimes find example questions in the train set that had the same first word but had drastically different question structure. An example is shown below:

Test question:

- What may also be required of teachers, in some areas?

Train examples:

- What individuals live at Fatima House at Notre Dame?
- What type of religion is Kirant Mundhum?
- What is another name for ancestor worship?

In the example above, the test question has the same first word ("What"), but the question structure itself is different from the train examples. It has a comma and a clause at the end, and the train examples do not. Additionally, some of the training examples may be considered to be in different categories from eachother. For example, "What individuals ... " is more similar to a question that would be asking "Who ..." instead of "What ...". Thus, the simple rule-based approach used here could be too simplistic to get meaningful separation of syntactically similar questions.

6.2 Lexical and Semantic Models

The results shown in Table 1 revealed that the lexical and semantic models both provided some improvement in EM scores over the random baseline. However only the semantic method provided improvement in F1 score. We hypothesize that since the semantic model looks for questions that are similar in meaning and then adds the question, its associated answer, and background into the GPT-3 input, it adds additional information relevant to answering the test question that helps GPT-3 generate the answer.
Moreover, finding semantically similar examples casts a wider “net” and not only captures examples that are similar to the test question in simple term-

Prompt Type	Task Description	EM	Macro F1
Structured Input		0.425	0.5460
Structured Input	Find the correct answer to this question.	0.415	0.5238
Structured Input	Answer this question.	**0.445**	**0.5614**
Structured Input	I am a highly intelligent question answering bot ...	0.400	0.5365
Natural Input		0.380	0.5011

Table 3: Experiment results for different task description which shows at the beginning of the GPT-3 input. The last line is for natural inputs as compared to structured ones (Appendix A shows an example). The highest EM and F1 scores are in bond font.

Selection Strategy	Example Order	EM	Macro F1
Random	Random	0.370	0.4933
Semantic Similarity	Reversely Ordered	0.373	0.4949

Table 4: Evaluation results on the test set of our best model and the baseline model.

matching, but also captures those examples that are similar in meaning.
Appendix B presents the results and analysis of the hybrid strategy and the reason we exclude them from our main report. It didn't yield performance any better than these methods alone.

6.3 Examples Ordering

The third variable that was modulated was the order of examples in the prompt that was fed into GPT-3. The ordering was either random, or in increasing level of similarity. Based on the results shown in Table 1, having increasing order of similarity as input improves performance in the semantic model. We hypothesize that having the most similar example closest to the test question in the input helps GPT-3 learn and understand the best example the strongest. While GPT-3 has demonstrated substantially good memory capacity and can recall verbatim, understanding natural language and answering given questions is much harder than merely memorizing, while the in-context setting without a fine-tuning procedure also adds to the difficulty. In this context, our experiment results revealed a potentially non-trivial impact of the example ordering mechanism.
It would be interesting to perform repeated experiments and examine these results with a varied number of examples to see if the gap in performance widens with more examples. Our results show that a wisely chosen example number can enhance the model's performance.

6.4 Number of Examples

Table 2 and Figure 2 shows that for both the baseline model and the semantic model, increasing the number of examples results in a performance increase followed by a decrease. For the random baseline model, the optimal number we found was 4, and for the semantic model, the optimal is 3.
Choosing the number of examples involves a tradeoff: while more examples can potentially help the model to learn the task better, they also add noise and may affect the performance by posting a heavier burden on the model's memorizing capacity.
As a note, Table 2 has overlapped experiments but the results slightly differ. It's because the two sets of experiments were conducted in different runs. We didn't average among multiple runs due to limited budget. It's necessary to generate averaged results among multiple runs for the future work.

6.5 Task Description

We tried a number of different task descriptions for the question answer task, and found that we were not able to observe a consistent improvement above baseline for any one description. Although Table 3 shows that "Answer this question" outperforms the baseline, while others don't, a different set of experiment runs (shown in Appendix E) show substantially different results. It's also not convincing to claim that "Answer this question" is a good task description, while a very similarly worded sentence "Find the correct answer to this question" is not.
One possible explanation is that the GPT-3 input is generally long (with around 1000 tokens), so the task description is too short compared to the entire

input, thus not having an observable impact.
Again, we didn't average among different runs due to limited budget, and future experiments can conduct duplicated experiments to examine if different task descriptions lead to consistent different in results.
We also tried a modified input example, where the examples were reformatted to look like how a question would naturally occur in an online forum. We also did not see an improvement in this setting.

6.6 Evaluation on Test Set

As Table 4 shows, our best model is only slightly better than the baseline model. This disagrees with our findings on the dev set. It's possible that we overfitted our methods on the dev set. However, since we only have a test size of 300, the result generated on it can be biased. A more thorough test evaluation should be undertaken in our future work.

7 Conclusion

Overall, we present work on improving the performance of GPT-3 on the in-context learning Open Question Answering task by engineering the input to GPT-3. We show the results of varying the selection strategy of input train examples, modifying the number of examples, and modifying the task description/prompt language. The best performing system had an input with training examples that were most semantically similar, and had the examples in order of increasing similarity. We found that the lexical and syntactic similarity methods did not offer as much improvement above random, and the optimal number of examples in the input was between 3-4 examples. The input with examples from each of the three similarity methods did not offer much improvement over the semantic model. Additionally, adding the task description or modifying the input to sound more natural did not improve results over the baseline description-less input. These results show that modulating different aspects of the input to GPT-3 can improve performance on the in-context OpenQA task. Future work, detailed below, is need to understand the robustness of these proposed methods and identify other potential input-modification strategies.

Known Project Limitations & Future Work

Known Project Limitations

As mentioned in the previous sections, due to time and cost constraints, we were not able to perform repeated experiments on each of the models and average out the results. Our reported results are for one run of each experiment. On the other hand, we batched the examples before passing into GPT-3 and it may have compensated for averaging out the results. For each batch, all test examples in that batch have the same randomly generated examples in GPT-3 input.
As discussed in Section 6.1, our design choice for the syntactic model was rather simplistic and led us to getting example questions that differ syntactically from the input question. If time would have permitted, we could have made use of regEx or some other pattern matching logic to enhance the syntactic model.
The majority of our experiment results is reported on our dev set. Given the limited budget, we only ran the best model and the baseline on the held-out test set. Moreover, limiting our dev size to 200 and test size to 300 can been too small of a scope to truly evaluate our model's performance.

Future Work

With more time and cost budget, we would want to run duplicated experiments and average out the results. We would also experiment on a larger dev set and a larger test set.
Additionally, we would want to confirm our hypothesis that the syntactic model should show substantially better performance than the baseline of randomly generated examples. To do this, there is need to design a stronger syntactic model.
Another interesting future work is to hand-select a few examples that generally work well for different input questions. We may also observe certain patterns within these examples and thus can design automatic example generation methods.
As mentioned in 2.2, we are using OpenAI's second powerful GPT-3 model due to limited budget. We recommend a rerun of the experiments on stronger GPT-3 models to get better and potentially more robust results.

Authorship Statement

GS: Worked on code for syntactic and lexical similarity methods, as well as custom input

generation. Ran preliminary experiments on baseline model and lexical similarity models. Wrote sections on prior literature, methods and analysis for this paper.
OE: Worked on the introduction, parts of the literature review, data, future work and project limitations. Did some research on prompt generation.
KH: Implemented the lexical and semantic similarity models, the hybrid model, and different example ordering mechanisms. Conducted the experiments. Wrote sections for related work, methods, experiments & results and analysis for this paper.

References

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

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.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Tianyu Gao, Adam Fisch, and Danqi Chen. 2020. Making pre-trained language models better few-shot learners. *arXiv preprint arXiv:2012.15723*.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. Realm: Retrieval-augmented language model pre-training. *ArXiv*, abs/2002.08909.

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.

Omar Khattab and Matei Zaharia. 2020. Colbert: Efficient and effective passage search via contextualized late interaction over bert. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, pages 39–48.

Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. 2019. Latent retrieval for weakly supervised open domain question answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6086–6096, Florence, Italy. Association for Computational Linguistics.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*.

Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. 2022. 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.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2021. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *arXiv preprint arXiv:2107.13586*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

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, and Peter J Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *EMNLP/IJCNLP (1)*, pages 3980–3990. Association for Computational Linguistics.

Keshav Santhanam, Omar Khattab, Jon Saad-Falcon, Christopher Potts, and Matei Zaharia. 2021. Colbertv2: Effective and efficient retrieval via lightweight late interaction. *arXiv preprint arXiv:2112.01488*.

Sang Michael Xie, Aditi Raghunathan, Percy Liang, and Tengyu Ma. 2021. An explanation of in-context learning as implicit bayesian inference. *arXiv preprint arXiv:2111.02080*.

Wei Yang, Yuqing Xie, Aileen Lin, Xingyu Li, Luchen Tan, Kun Xiong, Ming Li, and Jimmy Lin. 2019. End-to-end open-domain question answering with bertserini. In *NAACL HLT 2019 - 2019 Conference of the North American Chapter of the Association for Computational Linguistics*, NAACL HLT 2019 - 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Demonstrations Session, pages 72–77. Association for Computational Linguistics (ACL). Publisher Copyright: © 2019 The Association for Computational Linguistics. Copyright: Copyright 2020 Elsevier B.V., All rights reserved.; 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL HLT 2019 ; Conference date: 02-06-2019 Through 07-06-2019.

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*.

Appendix

A GPT-3 Input Example

An example input that contains three examples retrieved by the lexical, syntactic, semantic similarity models, respectively:

```
Title: Atlantic_City,_New_Jersey
Background: The median income for
 ↪  a household in the city was
 ↪  $26,969, ..., including 29.1%
 ↪  of those under age 18 and
 ↪  18.9% of those age 65 or
 ↪  over.
Q: What percentage of the
 ↪  population in the city were
 ↪  living below the poverty
 ↪  line?
A: 23.6%

Title: London
Background: There are 366 railway
 ↪  stations in the London
 ↪  Travelcard Zones on an
 ↪  extensive above-ground
 ↪  suburban railway network ...
 ↪  Clapham Junction is the
 ↪  busiest station in Europe by
 ↪  the number of trains passing.
Q: How many railway stations are
 ↪  utilized by London's railway
 ↪  network?
A: 366

Title: Guinea-Bissau
Background: Guinea-Bissau's GDP
 ↪  per capita is one of the
 ↪  lowest in the world, ... The
 ↪  economy depends mainly on
 ↪  agriculture; fish, cashew
 ↪  nuts and ground nuts are its
 ↪  major exports.
Q: How much of the population
 ↪  lives below the poverty line?
A: More than two-thirds

Title: Kenya
Background: years below 1990
 ↪  levels. The infant mortality
 ↪  rate is high at approximately
 ↪  44 deaths per 1,000 children
 ↪  in 2012 ... According to 2009
```

```
Q: How many Kenyans are living
 ↪  below the poverty level?
A:
```

B Results for Hybrid Strategy

We also conducted experiments with the hybrid strategy described in Section 4.3.4 and compared their results to the three strategies. However, those were run on a different day and generated a different set of results (the results for the same model are not very consistent). This may be resulted from a different random seed of GPT-3. Due to budget constraint, we were not able to rerun experiments for the hybrid ones. We discuss this randomness factor in the limitation section.
The combined strategy where one example of each of lexical, semantic, and syntactic similarity together did not yield performance that was any better then these methods alone. Therefore, further experimentation on order of examples did not focus on this hybrid method.

C Task Descriptions

Here's the list of task descriptions that we experimented with:

- I am a highly intelligent question answering bot. If you ask me a question that is rooted in truth, I will give you the answer. If you ask me a question that is nonsense, trickery, or has no clear answer, I will respond with Unknown.
- Find the correct answer to this question.
- Answer this question.
- Translate this sentence.
- Let's think step by step.

D Natural Input Example

The first three Title/Context/Question/Answer examples are train examples, and the last example is a Question/Answer prefix that is the test example.
I searched on Google and found this article: Kathmandu
The article said Kirant Mundhum is one of the indigenous animistic practices of Nepal. It is practiced by Kirat people. Some animistic aspects of Kirant beliefs, such as ancestor worship (worship of Ajima) are also found in Newars of Kirant origin. Ancient religious sites believed to be worshipped by ancient Kirats, such as Pashupatinath, Wanga Akash Bhairabh (Yalambar) and Ajima are now worshipped by people of all Dharmic religions in Kathmandu. Kirats who have migrated from other parts of Nepal to Kathmandu practice Mundhum in the city.
Q: What type of religion is Kirant Mundhum?
So your answer is: animalistic
I searched on Google and found this article: xxx
The article said xxx
Q: xxx
So your answer is: xxx
I searched on Google and found this article: xxx
The article said xxx
Q: xxx
So your answer is: xxx
Q: What group was responsible for causing more violence in Wittenberg?
So your answer is: xxx

E Another Set of Task Description Experiments

Table 5 shows the results of another set of task description experiments, which disagree with Table3. Also, some adversarial experiments also lead to good performance (e.g. "Translate this sentence."). The results are believed to be not consistent.

Task Description	**EM**	**Macro F1**
I am a highly intelligent question answering bot...	0.405	0.5403
Find the correct answer to this question.	**0.430**	**0.5558**
Answer this question.	0.390	0.5485
Translate this sentence.	0.400	0.5451
Let's think step by step.	0.385	0.5387

Table 5: Experiment results for different task description which shows at the beginning of the GPT-3 input. The highest EM and F1 scores are in bond font.