Turning Zero Shot into Few Shot via Self-prompting for Classification

Michael Gunn, Justin Lin, Lilesh Kurella, Gaozheng Liu, Kulbir Singh Ahluwalia
{mwgunn2, jklin6, lileshk2, gl11, ksa5}@illinois.edu

Abstract

In general, there is a significant gap between LLMs zero-shot and few-shot performance. Zero-shot classification performance is preferable from a user's point of view, as providing examples is time-consuming and cumbersome. This project explores the possibility of achieving results superior to zero-shot on classification tasks without needing to utilize information and examples outside of each problem instance independently. The idea is to, at inference time, first prompt the LLM to generate its own examples, then feed those examples as a prefix to the intended question with the goal of improved performance via relevant few shot-examples. To a user, this would potentially enable the best of both worlds of zero-shot querying, with close to few-shot performance.

1 Introduction

In this project, we aim to explore a novel approach to improving LLM classification performance using a pipeline to self-prompt in order to generate class-specific pseudo-labeled examples at inference time to be used as in-context examples. We examine our approach on the Words in Context (WiC) and Stanford Sentiment Treebank (SST2) classification tasks, while maintaining a method that can be generalized to other classification tasks.

We propose and explore an end-to-end pipeline that turns an instance of a classification problem into a series of self-prompts to generate relevant examples, and then combines them into one final prompt containing both the generated examples and the target problem instance at inference time. This approach aims to bridge the gap between zero-shot and traditional few-shot prompting while maintaining the ease of use of zero-shot querying.

All our pipelines can be found at the following Github

2 Background and Motivation

Previous work has indicated that language models seem to have a fundamental gap between their zero-shot and few-shot performance. To bridge this gap in performance while maintaining the ease of use to users provided by zero-shot performance other works have attempted fine-tuning models on generated training examples. We aim to instead turn zero-shot queries into few-shot queries at inference time by first prompting an LLM to generate examples to be used as in-context examples in the final query.

It has been shown in previous work such as FewGen (Meng et al., 2023), SuperGen (Meng et al., 2022), Zero-Gen (Ye et al., 2022) and other works that Pretrained Language Models (PLMs) can achieve improved performance by fine-tuning on examples with corresponding output labels that they generate themselves. This project aims to determine whether this concept can be taken one step further and employed on a frozen language model to achieve improved performance using generated examples as a prefix. Similar works, such as Self-Instruct (Wang et al., 2023), also demonstrate the filtering out of incorrect, low-quality, or similar generations before using their own generated instruction and corresponding instance data for instruction tuning.

Insights from Self-Prompting Large Language Models for Zero-Shot Open-Domain QA (ODQA)(Li et al., 2023) affirm and strengthen the foundation for our proposed concept. ODQA uses its self-prompting method to first learn the context of the QA task, and after fine-tuning those results, the LLM is used on QA tasks utilizing that context. The ODQA approach found significant improvements on multiple QA datasets and with different LLMs. The success of ODQA is inspiring, although differs from our solution in certain aspects. Currently, our solution is targeting classification in

comparison to QA tasks, and instead of predetermining context before evaluating a set of prompts, our goal is to generate a context for a given problem instance at inference time. This change to augmenting with examples that are generated at inference time makes our method generalizable to new classification task settings since examples are generated based on the problem instance in question and not retrieved from a premade example set.

We chose to test our proposed method on two classification tasks which are explained in detail in Section 3. While these two tasks don't capture all possible classification settings, we placed significant emphasis during our development and testing to employ an approach that can be generalized. We believe that while we were only able to evaluate our method on these two tasks, the idea of turning classification problems into pseudo-few-shot queries by generating examples of each class is transferable to other classification settings.

3 Data

We tested our approach of self-generated prompt augmentation on the task of classification using two datasets, Words in Context (WIC) and Stanford Sentiment Treebank 2 (SST2). The concept of our method is easily expandable to other classification tasks, but we use these two datasets to demonstrate specifically binary classification and our method's success on problems with different levels of difficulty. For our testing purposes, we use only the test sets of both datasets. We also only sample 500 cases when measuring our baseline and 250 for our other test cases. Additionally, no filtering was required to use the data in our pipeline, however, we did sample randomly from the test datasets.

3.1 Task description WIC

The task of 'Words in Context' is defined as identifying whether a target word is used in a similar or different sense in two different sentences. This task is also known as 'Word Sense Disambiguation' which requires understanding the contextual meaning of the target word. We obtain this dataset from Super Glue (Wang et al., 2020). This problem poses a significant challenge for all models preceding GPT-4 (OpenAI, 2023), with accuracy well under human performance. This task thus serves as a challenging setting for our method.

3.2 Task description SST2

The task of 'Stanford Sentiment Treebank 2' is defined as identifying whether a sentence has a positive or negative sentiment. The SST2 dataset is known to be an easier task compared to WIC, as it has in general higher baseline accuracies on the models we tested. We decided to test our approach on this dataset to see if task difficulty correlated with accuracy. We obtain this dataset from HuggingFace (Socher et al., 2013).

4 Methodology

Our pipeline comprises two parts: Example Generation, and Combination and Classification. Example Generation utilizes a prompt-based approach to produce examples corresponding to a given question. Combination and Classification integrate the original question with the generated examples to classify them under specific labels. Our input schema consists of a classification label with L choices, and a total of c input sentences to compare. The dataset we are using is Word in Context which consists of paired sentences sharing a target word. The classification labels are 'same' and 'different' based on the word sense of the target word in the pair of sentences.

The first step, Example Generation is to generate examples according to the original question. It outputs m more example sentences on an input sentence with each label. This generates for all combinations of our input sentences and corresponding labels thus we call our generation prompt n times where $n = L * c * m$. In terms of the Words in Context task, we take both sentences and generate two sets of five examples, one set with the same sense label and another set with a different sense label. A randomization strategy is used for generated examples. The ordering of 'same' and 'different' examples is randomized. The position of the reference sentence (original sentence) in each example is also varied. This strategy aims to eliminate patterns that could influence classification accuracy.

In the second step, Combination and Classification combine the randomized generated examples with the original question to form an integration prompt. The generated examples are regarded as self-generated few-shot examples, serving as a prefix to the final prompt as our few-shot examples. Our final prompt outputs the predicted label of the shared target word from the original sentences.

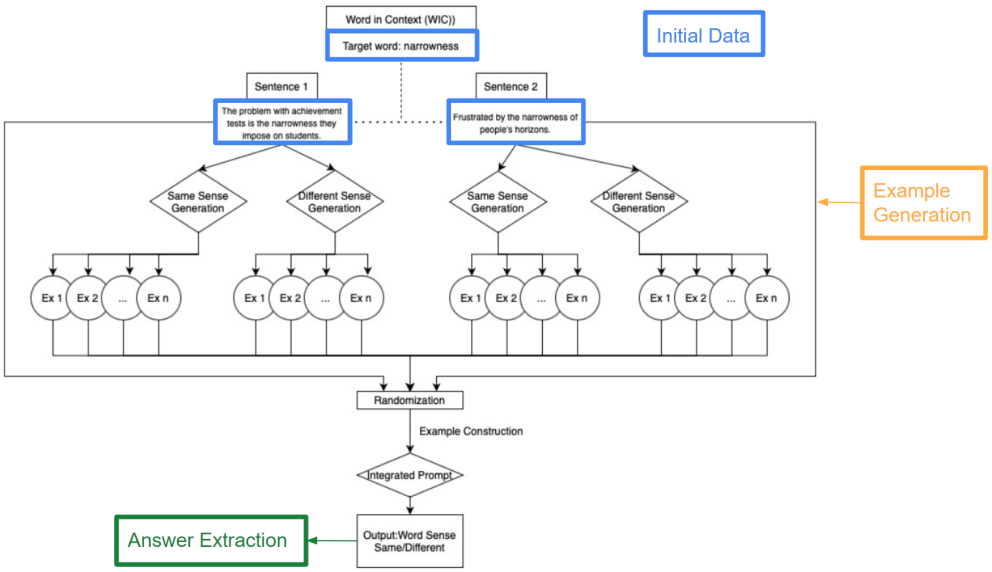

Figure 1: The methodology paradigm. The methodology contains two steps: 1. Example Generation: Utilizes a prompt-based approach to produce examples corresponding to a given question. 2. Combination and Classification: Integrates the original question with the generated examples to classify them under specific labels.

4.1 Example Generation

Llama-2-13b-chat-hf was used to generate additional sentence pairs with labels in similar or different word senses, as well as examples of each sentiment for the respective tasks at inference time. The chat version of Llama2's 13B model generated consistent outputs that understood and respected the task description in the input prompt. The example generation prompts results in examples that are consistent with the provided format. Extracting the generated sentences can be performed with simple splitting based on new lines and the location of the colon in the output. The extracted sentences are then fed into the final prompt construction step to have their order randomized and placed into the final query prompt.

4.1.1 Alternative Example Generation Approaches

Alternative prompting strategies, to generate examples of multiple classes at once, and other variations were tried. From our experimentation, especially using a relatively small model, the simpler we could make the generation step the better the resulting class specific examples were. For this reason, we stuck with the example generation of one class at a time, as this yielded the best performance and generalized best for other classification tasks.

4.1.2 WiC Example Generation

The exact example generation prompt for WIC given an input sentence *s1* is as follows:
"""
Consider the following sentence. Sentence 1: {*s1*}
Generate {*num_examples_to_generate*} diverse sentences with different sentence structures containing the word {*target*} where {*target*} has the same word sense as in sentence 1.
Sentence 1:
"""

4.1.3 SST2 Example Generation

The exact example generation prompt for SST2 given an input sentence *s1* is as follows:
"""
Consider the following sentence {*s1*}. Generate {*num_examples_to_generate*} diverse positive sentiment sentences with different sentence structures.
Sentence 1:
"""

4.2 Final Prompt Integrating Examples

Our final prompt consists of the original task description and the examples generated from the example generation. The task description includes the format of the examples provided as well as a description of the target label space. When including the generated examples in the final prompt we randomize the order of the examples of each class.

The final prompt template we use can account for as many in-context examples as desired, with all coming before the problem instance that is being classified. An example of the final augmented prompt, featuring a random order of self-generated examples with similar and different word senses can be seen in Table 1.

In Table 1 the constructed final prompt can be seen with all components highlighted. It contains primarily correct generated examples, but pair 3 consists of a generated sentence that is semantically very similar to the provided reference sentence. This is precisely the type of example that we attempted to filter in our final attempt, as discussed in section 4.4. The other example generations are correctly labeled and placed within the template with the "Answer" determined by the prompt used to generate the example, thus which pseudo-label should apply. Lastly, the problem instance sentences are appended in the same format as the examples, and we prompt for the target classification using "Answer:".

A very similar final prompt is used for SST2 but modified since that task contains only 1 sentence to be classified. The task description is modified to describe the objective accordingly, but the format of the task description, followed by generated examples with their pseudo-labels, and lastly the problem instance in question and a prompt for the output class, is replicated.

4.3 Answer Extraction and Accuracy Calculation

For the Words in Context (WiC) task, the output label has a binary value of '1' when the extracted label is 'Yes' for a similar word sense, or '0' if the extracted label is 'No' for a different word sense of the target word in two sentences. For SST2 there is a similar binary label representing whether the target sentence has positive or negative sentiment. In both tasks, and for both our baseline and the proposed method, the prompt yields an output of "yes", "no", "positive", or "negative" which can be easily translated to the corresponding label for each respective task. To obtain a final accuracy on either task and for either method of prompting, we simply calculate the accuracy with respect to the ground truth label provided in the dataset. Both datasets contain a balanced number of each class, so calculating more sophisticated label-specific metrics is not necessary.

	Determine if the word **'window'** in each pair of sentences is used with a broadly similar meaning or if there's a significant difference in its usage. Focus on the overarching sense of the word, rather than subtle nuances. Does **'window'** have a broadly similar meaning in both sentences? Answer 'yes' for similar or 'no' for different.
Sentence 1	You have a two-hour **window** of clear weather to finish working on the lawn.
Sentence 2	The **window** of the car was shattered in the accident.
Answer	No
Sentence 1	The **window** to her heart was closed, and she couldn't let anyone in.
Sentence 2	You have a two-hour **window** of clear weather to finish working on the lawn.
Answer	Yes
Sentence 1	**Window** of opportunity.
Sentence 2	The **window** of opportunity for the new project is now open.
Answer	Yes
Sentence 1	The **window** of the store was covered in posters and sign
Sentence 2	**Window** of opportunity.
Answer	No
Sentence 1	You have a two-hour **window** of clear weather to finish working on the lawn.
Sentence 2	**Window** of opportunity.
Answer:	Yes

Table 1: An example of a final combined prompt used for WiC. The text highlighted in red is part of the template. Text in blue comes from the problem instance in question. Text in orange is generated examples from the preceding steps. The output which corresponds to the classification is in green.

4.4 Filtering Generated Examples

We believe that a significant portion of the effectiveness of our approach depends on the quality of the generated examples which are used to construct the final prompt. This is true for both the classification problems we tested, but also in general for any classification tasks. The examples should be representative of the tasks which they are pseudo-labeled with.

Our final experiment which we conducted was filtering of the generated examples. For the filtering task for the Words in Context task, the Sub-Sentence Encoder (Chen et al., 2023) was used to generate contextual embeddings corresponding to the self-generated example pairs.

We believed that this approach would allow us to select the sentence pairs with a high semantic similarity score. It also removes repetitive examples from our context generation making our final

prompt more concise for the model. Our method for filtering consisted of comparing semantic similarity scores between the original input sentence and the self-generated example. Self-generated example pairs that exceed a predefined threshold are filtered from all possible pairs and are then included in the prompt.

This method achieved accuracies lower than without filtering and thus was not used for our final results, though we believe properly optimized filtering should yield improvements.

Our context generation sometimes yields examples that do not correctly correspond with the target label. Hence, the self generated example pairs with top-k cosine similarity scores between the subsentence encoder embeddings of the self-generated examples were chosen for the positive similar word sense labels. For the different word sense case, the self-generated pairs with bottom-k cosine similarities were chosen with the negative label. We believe that filtering these incorrect examples with the most probable label could lead to performance improvements, however, we were unable to accomplish this within our project since the resulting accuracy was lesser than that of the unfiltered pipeline.

5 Experimental Results

5.1 Models

In addition to Llama2-13b-chat, other models such as Llama2-7b-chat, Mistral v0.1, and Flan-T5-large were tried to generate examples for prompt augmentation. The outputs of these other LLMs were inferior to those generated using Llama2-13b-chat and would require entirely separate prompt tuning to obtain results. As a result, we decided to focus our efforts on maximizing performance using Llama2-13b-chat while maintaining an approach that could be applied to other capable models.

5.2 Baseline Performance

To test the effectiveness of our approach on the WiC and SST2 tasks we first needed to analyze the baseline capabilities of the Llama-2-13b-chat with 4-bit quantization. The WiC and SST tasks are both classification tasks in their simplest form, with 2 target labels for each.

5.2.1 Words in Context Baseline Performance

In WiC the output labels are "same" and "different", in reference to the word sense of the target word in each pair of example sentences. We tested a number of zero shot prompts as our baseline with the goal of maximizing the resulting accuracy. From our experimentation we found that the best accuracy when using the following prompt to get a label given a pair of sentences *s1, s2* along with the target word *target*:

"""

Determine if the word '*target*' in both sentences is used with a broadly similar meaning or if there's a significant shift in its usage. Focus on the overarching sense of the word, rather than subtle nuances.
Sentence 1: *s1*
Sentence 2: *s2*
Does '*target*' have a broadly similar meaning in both sentences? Answer 'yes' for similar or 'no' for different.
Answer:

"""

As seen in Table 2, using this prompt we achieved 61% accuracy on the first 500 instances of WiC. The model seemed to understand the task and the label space well, with all outputs being from the target label space. This accuracy outperforms random guessing, but still leaves much to be desired in terms of performance. The WiC task is challenging in general, with only the most cutting edge LLM's like GPT-4 (OpenAI, 2023) achieving near or above human performance. The difficulty of this task and the low baseline performance offer challenges for our self generated example approach, which motivated testing on an easier classification problem in the form of SST2.

5.2.2 Stanford Sentiment Treebank (SST2) Baseline Performance

To evaluate whether our method of prompt augmentation with self-generated examples works better if the underlying task is easier for the model being tested we evaluated the performance of the Stanford Sentiment Treebank (SST2) problem. This problem is also a classification task, with the target labels being "positive" or "negative", in reference to the general sentiment of the sentence in question. This is a well-understood task for most LLMs, with many methods achieving high classification performance. We achieved the best results using Llama2-13b-chat 4-bit quantized model with the following baseline zero shot prompt given an input sentence *s1*:

"""

Determine if the following sentence has positive or negative sentiment. Answer 'positive' or

	Baseline 0-shot	Self-gen 4-shot*	Self-gen 12-shot*	Self-gen 16-shot*
WiC Accuracy	60%	55%	60%	64%

Table 2: Llama2-13b-chat (4bit quantized) WiC performance. * Self-gen n-shot means that n/4 examples of each of the following were included: same sense as s1, different sense as s1, same sense as s2, different sense as s2. This results in an even number of examples for both target labels and uses both provided problem instance sentences.

	Baseline 0-shot	Self-gen 2-shot*	Self-gen 6-shot*
SST2 Accuracy	89%	93%	93%

Table 3: Llama2-13b-chat (4bit quantized) SST2 performance. * Self-gen n-shot means that n/2 examples of each label were included in the final prompt.

'negative'. Sentence: *s1*
Answer:
"""

As seen in Table 3, with this prompt we achieved 89% accuracy on the first 500 items of the SST2 dataset. This result is in line with others zero-shot performance results on the dataset. We chose this task as our second evaluation in order to examine whether our idea of using self generated label specific examples could offer more improvement on an already high performing classification task.

5.3 Evaluation of our pipeline

After establishing the baseline performance of the model, we used on both WiC and SST2, we evaluated the performance impact of employing self-generated label-specific examples as in context examples. The process of generating the examples has been discussed in section 4.1. After generating class-specific relevant examples, we constructed the final prompt which combines the problem instance data with the generated examples to obtain a final output label for the problem instance. We were able to achieve a performance improvement on both tasks using our method, with WiC accuracy going from 60% to 64%, and SST2 accuracy going from 89% to 93%, as can be seen in Tables 2 and 3.

The entire process of generating examples does come with an inference time increase inherently since one problem query is instead turned into 2 or more label-specific example generation queries, depending on the specifics of the task, which are then combined into one final prompt. This slowdown in inference time did limit our ability to test a larger subset of each dataset, as well as how many variations of prompts for each step we were able to test. In addition to testing many different prompt formats, we also examined the impact of the number of self-generated examples included in the final prompt on performance.

5.4 Error Analysis

During our testing, we did recognize that particularly for the WiC task, some of the self-generated examples were incorrect. By this, we mean that when prompted to generate a set of sentences that used the target word with the same or different word sense as one of the two problem instance sentences, some of the resulting generations did not align. When asking it to generate sentences using the target word in the same word sense as a reference sentence, some of the examples would use the word with a different semantic meaning. These incorrect examples were still the minority, as many of the self-generated examples did align with the prompt's request.

While most examples are correct, we believe that these incorrect examples could potentially be misleading as they lead to incorrectly pseudo-labeled examples in the final prompt. The basis for this conclusion is our empirical observation that by including enough examples in the final prompt we were still able to achieve a performance improvement over the baseline. The idea is that enough examples overpower the effect of the misleading examples, allowing for improvement with a sufficient number of in-context examples included.

5.5 Prompt Engineering and Hyperparameters

To improve the quality of self-generated examples, we engineered the prompts to extract more knowledge from the language model. In the WiC task, our approach involves two significant changes. One change is to generate different sentence structures, and the other change is to generate diverse sentences. Both changes aim to increase the diversity of the examples. We observed that the model gen-

erated a list of similar examples, which did not provide more information gain for the task, and did not alter the wrong classification.

Furthermore, we want to build the robustness of the model. We use "general same meaning" instead of "same meaning" when generating examples of the same word sense. This adjustment allows the model to ignore minor differences in word usage.

Regarding the varied parameters of our experiments, we primarily vary the number of examples generated for each case and each classification label. In the WiC task, we experimented with 1, 3, and 4 examples for each class and based on each reference sentence, meaning 4, 12, and 16 total examples. For the SST2 task, we used 1 and 3 examples of each class, meaning 2 and 6 total examples were used. Our results suggest a positive correlation between the increased number of examples and the resulting classification accuracy. Some generation hyperparameters such as decoding strategy were tested, but we found HuggingFace Transformer defaults to yield the best performance.

5.6 Impact of Model Size and Generalization

We believe that the slightly unintuitive result 2 of including a small number of generated examples leading to a performance decrease compared to the zero-shot baseline has a few contributing factors. First is the size of the model that we are working with. Llama2 13b is a reasonably sized model, but it has been shown in many works that larger models are more capable of utilizing in-context examples (Gao et al., 2021). Additionally, we believe that the approach we propose is more effective the better the model is on the task in a zero-shot context. This is because if the model already can accomplish the task reasonably well, then it also can generate higher quality and more correct examples to be used in the final augmented prompt.

This is the primary motivation for our testing on SST2, and our later efforts to attempt to filter the generated examples for quality. We aimed to ensure that the technique we propose generalizes to a wide range of classification tasks despite our limitation of the number of tasks we could test. We believe the general approach of generating examples representing each possible class, and then including those examples to turn a zero-shot classification instance into a pseudo-few-shot instance can offer performance improvements. We were able to demonstrate those performance improvements on both of the classification tasks that we examined, as seen in tables: 2 3.

6 Conclusion

In this project we aimed to explore a method of prompting LLMs to generate class-specific examples relevant to a given classification problem instance at inference time, and whether those generated examples could serve as in-context few-shot examples to improve classification performance. We were able to obtain a performance improvement over baseline zero shot prompting on the model we focused on while using no new task-specific information outside of the problem instance being answered.

We believe that this approach has the potential to improve classification performance for a variety of LLMs and in a variety of classification problem settings. In principle, we demonstrated that the addition of examples which the same model itself can generate, can provide useful contextual knowledge that improves performance. We can turn a zero-shot query into a series of example generation queries, followed by a final prompt construction that integrates the generated examples to achieve accuracy near traditional few-shot prompting.

The removal of the necessity to include dataset-specific few-shot examples maintains the ease of use of zero-shot querying while bridging the performance gap. We believe that our proposed method can serve as inspiration for further self-prompting techniques, and also warrants further examination using larger and more capable models. The area of assessing and maximizing LLM capability without any additional training remains in its infancy, and this work demonstrates that clever prompting and pipelines can extract capability above simple zero-shot prompting.

7 Appendix

7.1 Contributions

- Michael: Initial concept, prompt engineering for performance, pipeline construction and refinement, running experiments. Writing: Introduction, motivation, Experimental results, conclusion.
- Justin: Automation of pipeline code, SST2 pipeline. Writing: Data, Methodology
- Gaozheng: Creating pipeline diagram, offering alternative example generation approach

and run experiments. Writing: Alternative example generation approach

- Lilesh: Automation of pipeline code, SST2 pipeline. Writing: Data, Methodology
- Kulbir: Add Filtering in pipeline, Generate JSONs, Writing: Filtering self-generated examples.

8 References

References

Sihao Chen, Hongming Zhang, Tong Chen, Ben Zhou, Wenhao Yu, Dian Yu, Baolin Peng, Hongwei Wang, Dan Roth, and Dong Yu. 2023. Sub-sentence encoder: Contrastive learning of propositional semantic representations.

Tianyu Gao, Adam Fisch, and Danqi Chen. 2021. Making pre-trained language models better few-shot learners.

Yuang Li, Yu Wu, Jinyu Li, and Shujie Liu. 2023. Prompting large language models for zero-shot domain adaptation in speech recognition.

Yu Meng, Jiaxin Huang, Yu Zhang, and Jiawei Han. 2022. Generating training data with language models: Towards zero-shot language understanding.

Yu Meng, Martin Michalski, Jiaxin Huang, Yu Zhang, Tarek Abdelzaher, and Jiawei Han. 2023. Tuning language models as training data generators for augmentation-enhanced few-shot learning.

OpenAI. 2023. Gpt-4 technical report.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2020. Superglue: A stickier benchmark for general-purpose language understanding systems.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. Self-instruct: Aligning language models with self-generated instructions.

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. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 11653–11669, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.