SELF-DEMOS: Eliciting Out-of-Demonstration Generalizability in Large Language Models

Wei He[1], **Shichun Liu**[1], **Jun Zhao**[1], **Yiwen Ding**[1], **Yi Lu**[1], **Zhiheng Xi**[1], **Tao Gui**[2*], **Qi Zhang**[1*], **Xuanjing Huang**[1]
[1] School of Computer Science, Fudan University
[2] Institute of Modern Languages and Linguistics, Fudan University
whe23@m.fudan.edu.cn, {tgui,qz}@fudan.edu.cn

Abstract

Large language models (LLMs) have shown promising abilities of in-context learning (ICL), adapting swiftly to new tasks with only few-shot demonstrations. However, current few-shot methods heavily depend on high-quality, query-specific demos, which are often lacking. When faced with *out-of-demonstration* (OOD[1]) queries, methods that rely on hand-crafted demos or external retrievers might fail. To bridge the gap between limited demos and OOD queries, we propose **SELF-DEMOS**, a novel prompting method that elicits the inherent generalizability in LLMs by query-aware demo generation. The generated demos strategically interpolate between existing demos and the given query, transforming the query from OOD to ID. To evaluate the effectiveness of our approach, we manually constructed **OOD-Toolset**, a dataset in the tool-using scenario with over 300 real-world APIs and 1000 instances, each consisting of three tool-use cases as demos and an OOD query. Thorough experiments on our dataset and two public math benchmarks have shown that our method can outperform state-of-the-art baselines in the OOD setting. Moreover, we conduct a range of analyses to validate SELF-DEMOS's generalization and provide more insights.[2]

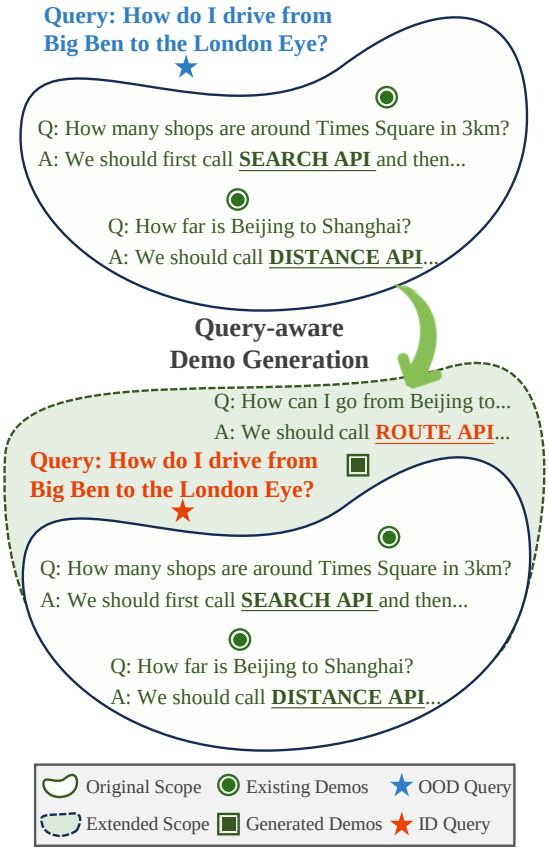


Figure 1: An example of how query-aware demo generation works. In the tool-using scenario, there is a gap between the user query and the available tool-use cases in the original scope since they require different APIs. This can lead to errors if the LLM is unfamiliar with the ROUTE API. After interpolating new demos between the existing ones and the OOD query, LLMs can perform better in the extended scope.

1 Introduction

Large language models (LLMs) have achieved impressive performance across a wide range of tasks, ranging from mathematical reasoning to tool using (Brown et al., 2020a; Kojima et al., 2022; Qin et al., 2023; Xi et al., 2023). The models learn to perform unseen downstream tasks simply by conditioning on a prompt containing input-output pairs (i.e., *few-shot demonstrations*, Brown et al., 2020a). This paradigm, also known as in-context learning (ICL), has been found its effectiveness considerably influenced by the quality and relevance of the demos provided (Liu et al., 2022; Dong et al., 2023). Thus, how to provide high-quality demos becomes an essential challenge in LLM applications.

The leading few-shot techniques typically hinge on hand-crafted task-specific demos or extensive

* Corresponding authors.

[1]OOD refers to "Out-of-Demonstration" in this paper, not the commonly understood "Out-of-Distribution". Similarly, ID stands for "In-Demonstration".

[2]Code & Data: https://github.com/hewei2001/Self-Demos.

Findings of the Association for Computational Linguistics: NAACL 2024, pages 3829–3845
June 16-21, 2024

demo libraries (Wei et al., 2022c; Liu et al., 2022; Rubin et al., 2022). However, crafting demos for each unique query is impractical, and the demo libraries are also unable to cover all the potential queries. The issue arises when faced with out-of-demonstration (OOD) queries, resulting in poorer performance due to the gap between existing demos and new queries.

An alternative strategy is prompting the LLMs to self-generate relevant demos, thereby guiding themselves toward resolving the query (Kim et al., 2022; Chen et al., 2023b; Yasunaga et al., 2023). However, these works often overlook a critical point: instead of blindly recalling relevant demos based on queries, we can perform interpolation between existing demos and queries, as depicted in Figure 1. By strategically interpolating, we can derive more relevant and accurate demos from existing ones, which have proven helpful for the final response (Liu et al., 2022; Halawi et al., 2023). Specifically, we introduce **SELF-DEMOS**, a novel prompting method that may fully elicit the model's potential out-of-demonstration generalizability. Unlike previous works, we developed a complete workflow incorporating pre- and post-processing steps around the demo generation. Before the demos are generated, we first prompt the model to *"give a general understanding of the user query"*, thereby simplifying the complexity of the analysis in subsequent steps. Then, we generate query-aware demos and select the most high-quality ones through *Best-of-N sampling* (Nakano et al., 2021). These selected demos will be used for the final response along with the initial available demos.

To evaluate our approach's efficacy in the OOD context, we manually construct **OOD-Toolset**, a dataset tailored for tool-using scenarios as delineated by Tang et al. (2023). Our dataset includes over 300 real-world APIs and 1000 instances, each consisting of three tool-use cases as demos and an OOD query. Moreover, we benchmarked our method with two public mathematical datasets, GSM8K (Cobbe et al., 2021) and MATH (Hendrycks et al., 2021), to validate its adaptability in different scenarios. The primary experimental findings reveal that SELF-DEMOS outperforms state-of-the-art baselines in solving OOD queries. We also conducted ablation studies and other extensive experiments to gain more insights into our method. Collectively, our analyses show that we have found a more efficient way to elicit the potential OOD generalizability in LLMs.

Our contributions are summarized as follows:

1. We proposed SELF-DEMOS, a novel prompting method to elicit the out-of-demonstration (OOD) generalizability in LLMs.

2. We manually constructed OOD-Toolset, a tool-using dataset for better verifying the potential OOD generalizability in LLMs.

3. We conducted extensive experiments to validate SELF-DEMOS's effectiveness and generalization under different settings.

2 Related Work

2.1 In-Context Learning

The rise of LLMs such as ChatGPT (OpenAI, 2022) and LLaMA (Touvron et al., 2023) has revolutionized the field. With the model size scaling, LLMs demonstrate remarkable capabilities of ICL (Brown et al., 2020b; Wei et al., 2022b), which learns to perform tasks by specific instructions and demonstrations. Additionally, insights from scaling laws (Wei et al., 2022b) also highlight the LLMs' potential for out-of-distribution generalization. It refers to the challenge where model inputs deviate from their training distribution (Wang et al., 2023a). If stimulated effectively, this generalization capability can empower LLMs to address queries outside the training corpus (Collins et al., 2022), enhancing utility in dynamic and open-ended scenarios.

2.2 Optimizing Demonstrations for ICL

The performance of LLMs may be influenced by the quantity, relevance, diversity, and truthfulness of demonstrations (Chen et al., 2023a; Levy et al., 2023; Min et al., 2022; Halawi et al., 2023). There are two primary paradigms to optimize demonstrations and steer models towards generalization.

Demo Retrieval for ICL. LLMs are sensitive to the choice of demonstrations. Therefore, researchers have focused on using retrieval modules to find the most representative demos for ICL. One effective strategy is leveraging existing retrievers based on semantic similarity metrics between the available demos and queries (Liu et al., 2022; Agrawal et al., 2023; Gao et al., 2023; Luo et al., 2023). Another method employs ranking scores derived from fine-tuned language models (Rubin et al., 2022; Shi et al., 2022).

Demo Generation for ICL. Rather than extracting existing demos, demo generation aims to self-generate exemplars that closely align with the input. Kim et al. (2022) initially employed language models to produce demos from pre-defined labels. Subsequent works adopted a two-stage approach of generating and selecting demos (Li et al., 2022; Zhang et al., 2023; Shao et al., 2023). In contrast, our work leverages the intrinsic capabilities of LLMs to identify superior demos via best-of-N sampling.

Besides, there are approaches akin to ours. Chen et al. (2023b) adopt multi-steps to construct demonstration pairs, while Yasunaga et al. (2023) prompt LLMs to recall relevant demos before answering. However, our method stands out by combining pre- and post-processing steps around demo generation to guarantee the high quality of generated demos.

2.3 Eliciting LLMs' Power with Prompts

Efforts to enhance LLMs include finetuning with specific instructions (Wei et al., 2022a) and employing prompting strategies like Chain-of-Thought (CoT, Wei et al., 2022c). Our approach adopts the prompt-based strategy and draws inspiration from studies of the "self" series (Madaan et al., 2023; Wang et al., 2023b; Chen et al., 2023b). The essence of "self" is to leverage the model's inherent power, without external modules. Our method positions the LLM itself as an analyzer, generator, and selector, aiming to elicit its intrinsic generalizability to resolve OOD queries.

3 Methodology

In this section, we first introduce the construction process of OOD-Toolset. Next, we provide a detailed description of the SELF-DEMOS method, which is illustrated in Figure 2.

3.1 OOD-Toolset Construction

Recent works are evaluated on benchmarks such as BIG-Bench (Srivastava et al., 2022) and GSM8K (Cobbe et al., 2021). However, since these datasets may have been inadvertently included in the training data of LLMs, there is a risk of overestimating their ability to generalize to OOD query (Zhou et al., 2023). To mitigate this, we chose the tool-using scenarios that are less likely to occur during model training for assessment. Specifically, we constructed the dataset following the two steps:

Data Collection. Our original data derives from the tool-use corpus created by ToolAlpaca (Tang et al., 2023). It was composed of a wide range of real-world APIs complete with API descriptions, usage specifications, and multiple simulated tool-use cases. However, despite the dataset's comprehensiveness, we noted that the initial AI-generated tool-use cases contain some errors, such as ambiguous queries and incorrect API calls in response. These minor errors may prevent accurate judgment in our evaluation. Therefore, we engaged human annotators to manually refine the corpus, producing a high-quality version for more reliable assessment. Additional details and an example of OOD-Toolset are provided in Appendix B.

OOD Setting. We retained the user's queries and corresponding API calls from tool-use cases as input-output pairs for the evaluation. In addition, we kept the API descriptions and usage specifications from the refined corpus as context for LLMs. For each test instance, we provided three cases from the same API as initial available few-shot demos (also referred to as *seed demos*, or D_{seed}). Notably, in the OOD setting, the sub-APIs in seed demos differ from those needed in the final query.

Take the MAP tool for example, which contains three sub-APIs: DISTANCE, ROUTE, and SEARCH API. For instance, if the DISTANCE and SEARCH APIs serve as seed demos, the user's query might pertain to the ROUTE API. This design tests the model's ability to understand and apply tool-using patterns across different functions, allowing us to explore the OOD generalizability in LLMs.

3.2 SELF-DEMOS

We executed the whole workflow by prompting the model itself. The prompt template for each step is illustrated in Appendix C.

Query Understanding. The first step involves comprehensive query understanding. Given the model $\mathcal{M}$ and a query q, we employ a zero-shot method:

$$u = \mathcal{M}(p_1 \,||\, q), \tag{1}$$

where p_1 is the prompt for query understanding, $||$ denotes concatenation, and u is the generated understanding. During this pre-processing step, we aim to reduce the disparity between the initial seed demonstrations and the ultimate target query. As shown in Figure 2, when given a query that involves MAP API, we guide the model to generate an un-

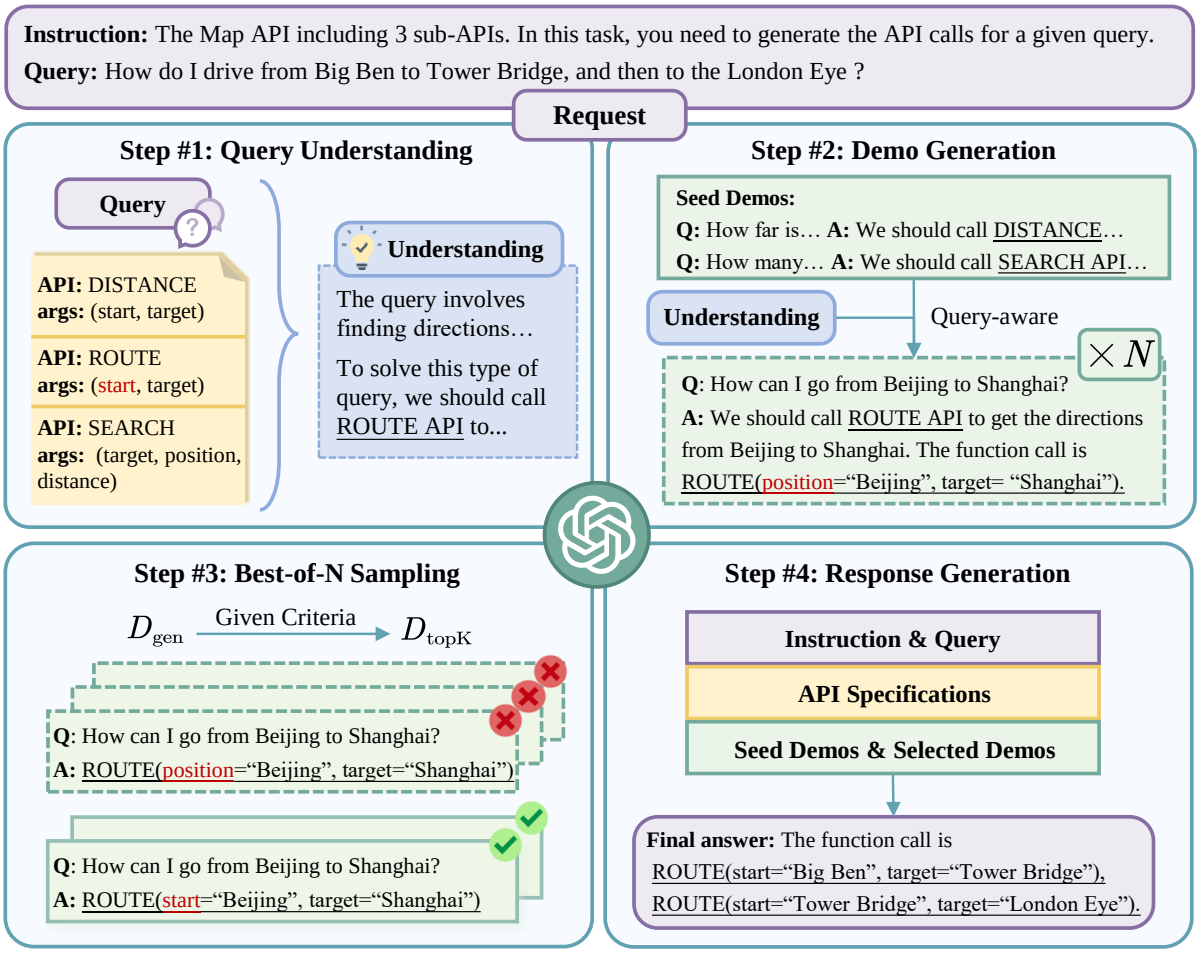


Figure 2: An overview of the proposed SELF-DEMOS prompting method in tool-using scenario.

derstanding focused on the more specific ROUTE sub-API. Furthermore, this step resembles a chain-of-thought process (Wei et al., 2022c), which may reduce the cognitive load in subsequent steps. This is helpful to enhance the relevance and accuracy of the generated demos.

Query-aware Demo Generation. Based on the distilled understanding u and seed demos D_{seed}, we generate query-aware demos as:

$$D_{\text{gen}} = \{d_1, d_2, ..., d_N\} = \mathcal{M}(p_2 \,||\, q, u, D_{\text{seed}}), \tag{2}$$

where p_2 is the prompt for demo generation, D_{gen} is the set of generated demos, and N is the number of demos to be generated. The seed demos, while not directly linked to the specific query, showcase potential tool-using patterns of MAP API, offering guidance for the generation. We call the model N times to generate N demos separately, alleviating the difficulty of a single try and avoiding the model falling into consecutive errors in one response. In this phase, we extend the original scope of the demos to a broader boundary.

Best-of-N Sampling. It has been argued that LLMs are unlikely to self-critique their outputs without an external validator (Stechly et al., 2023; Valmeekam et al., 2023). Consequently, we assume that while models might not calibrate and refine outputs, they could still discern the superior output from a variety. Therefore, we employ a Best-of-N sampling strategy, where the model is prompted to select the best K demos from the N generated demos based on special criteria:

$$D_{\text{topK}} = \mathcal{M}(p_3 \,||\, D_{\text{gen}}, C, K), \tag{3}$$

where p_3 is the prompt for sampling, D_{topK} is the subset of K demos sampled from the generated ones, conditioned on criteria C.

This process is inspired by preference learning, where multiple samples are generated and the one with the highest reward model score is chosen (Nakano et al., 2021). It is worth noting that our criteria, which include the demos' accuracy, relevance,

and potential helpfulness for the final response, are given to the model via prompts. Our sampling criteria are more nuanced and do not rely on an external retriever. This is where SELF-DEMOS differs from methods such as Synthetic Prompting (Shao et al., 2023), which also selects demos after generation.

Response Generation. Finally, we leverage the sampled demos D_{topK} and the initial seed demos D_{seed} to generate the final response:

$$r = \mathcal{M}(p_4 \,||\, D_{\text{seed}} \cup D_{\text{topK}}, q), \tag{4}$$

where p_4 is the prompt for response generation, $\cup$ denotes the concatenation of two sets and the r is the final response. The concatenation ensures that the model benefits from the query-specific demos in D_{topK}, while also incorporating the beneficial diversity and quality of D_{seed} (Levy et al., 2023; Halawi et al., 2023).

4 Experiments

To evaluate the effectiveness of SELF-DEMOS, we conduct extensive experiments for comparison and analysis.

4.1 Experimental Setups

Foundation Models. We use GPT-3.5 (the `gpt-3.5-turbo-0613` version) for most of our experiments, with only one additional experiment using the Llama-2-Chat model family, to validate the generalization of SELF-DEMOS across different model sizes. For all LLMs, we set the parameter $temperature = 0$ for stable responses except for the sampling step, where we set $temperature = 0.7$ to introduce diversity.

Tasks & Datasets. We evaluate the proposed method in two reasoning-intensive tasks: tool-using and mathematical problem-solving.

In the tool task, we developed the OOD-Toolset for evaluation. Details of the construction process are described in section 3.1. In the math task, we employed two public datasets: GSM8K (Cobbe et al., 2021), featuring elementary math word problems, and MATH (Hendrycks et al., 2021), containing complex problems from high school competitions. We evaluate the entire GSM8K testing set and a randomly selected subset from the MATH testing set. Distinct OOD settings are designed for math tasks. For GSM8K, we manually created several outlier samples, ensuring that the testing set did not contain problems with similar contexts. For MATH, since the problems were categorized into seven subjects and five difficulty levels, we used problems from different subjects but the same level to meet the OOD condition. The dataset statistics are presented in Table 1.

Evaluation Metric. In the report for the math tasks, we present the exact match accuracy for each problem. For the tool task, which may require multiple API calls in one case, we assess accuracy using both exact and partial matches. Partial matches are awarded half the score if the model's response includes only part of the required API calls.

4.2 Baselines

We compare SELF-DEMOS with the following baselines, including two methods that are designed for demo generation:

Zero-shot and Zero-shot + CoT (Brown et al., 2020a; Kojima et al., 2022). Prompt the model with the task description, test input, and no demonstration. Besides, the CoT method integrates a trigger prompt *"let's think step by step"*.

Few-shot (Wei et al., 2022c). Employ a fixed set of seed demos we constructed for each OOD query. For the GSM8K and MATH datasets, which include solutions with labeled reasoning steps, the demos also feature CoT steps to enhance the model's problem-solving capabilities.

Self-ICL (Chen et al., 2023b). A multi-step framework for zero-shot in-context learning by prompting the LLM itself to generate pseudo-inputs and labels. Unlike our method, they generate inputs and labels separately and then merge them into demos, with no other pre- and post-processing steps. We have also adapted it into a few-shot variant to make it comparable.

Analogical Prompting (Yasunaga et al., 2023). A single-step prompting method that guides LLM to recall relevant demos and knowledge before solving a given problem. Here we let it generate demos for the vanilla version and our few-shot variant. The vanilla Self-ICL and Analogical Prompting methods initially generate three demos each. However, in the few-shot variant, we adjust this to two demos to better align with our approach.

4.3 Main Results

Table 2 shows the performance of each method on three datasets. We can find that: (1) The better

Dataset Name	Size	Demo Source	Avg. #tokens of Query	Avg. #tokens of Demo	Avg. #tokens of Context (Few-shot)
OOD-Toolset	1,057	Same tool, different sub-APIs	35.5	53.8	496.0
GSM8K	1,319	Manually created outliers	59.0	136.8	526.1
MATH	1,000	Same level, different subjects	69.1	291.9	1002.1

Table 1: Statistics of three datasets in the OOD setting.

Prompting Method	OOD-Toolset		GSM8K	MATH	Average
	Exact Acc	Part Acc	Acc	Acc	
Zero-shot	64.5	68.4	75.0*	33.0*	60.2
Zero-shot + CoT	66.1	70.9	75.8*	33.9*	61.7
Few-shot	71.9	76.6	76.2	35.1	65.0
Self-ICL (Zero-shot)	67.0	71.1	76.6	34.6	62.3
Self-ICL (Few-shot)	71.5	76.0	78.0	**37.9**	65.9
Analogical Prompting (Zero-shot)	67.8	72.0	77.8*	37.3*	63.7
Analogical Prompting (Few-shot)	71.1	75.4	75.7	36.3	64.6
SELF-DEMOS (ours)	**75.1**	**79.4**	**78.2**	**37.9**	**67.7**

Table 2: Main results of different prompting methods on three datasets. All the results are with GPT-3.5-Turbo. The best performance for each task is in **bold**. The (*) indicates that results are from Yasunaga et al. (2023).

performance of few-shot over zero-shot (+ CoT) shows the LLM's capacity to discern and apply underlying patterns from seed demos to OOD queries, indicating a degree of inherent generalizability. Furthermore, the OOD-Toolset measures this ability more accurately than the two public math datasets, validating the necessity of creating unseen scenarios and OOD structures of instances. (2) Only a few-shot method does not fully unlock the model's capability. In contrast, the methods with demo generation, especially SELF-DEMOS, present superior performance, underscoring their potential to serve as a reliable prompting strategy in OOD scenarios. (3) Self-ICL, which generates Q&A separately, serves a similar purpose to our Best-of-N Sampling step by enhancing the accuracy of generated demos. Thus, it yields performance that is closest to our method. However, this framework may also lead to mismatches of Q&A pairs, i.e., the model fails to answer the questions it generates, which may affect subsequent responses. (4) Seed demos bring little benefit to the Analogical Prompting method and may even be harmful. This could be because the additional demos are irrelevant to the instructions of analogical reasoning, which require the model to do multiple tasks. The seed demos fail to guide the model in different tasks and may distract the model from the whole process. Overall, SELF-DEMOS outperforms all baselines in solving OOD queries.

Pre- & Post-processing Method	OOD-Toolset
w/o Pre-processing	72.9 / 77.5
+ Directly Answering	72.3 / 77.0
+ Query Understanding	**75.1 / 79.4**
w/o Post-processing	74.1 / 78.7
+ Self-Critique	74.3 / 78.8
+ Best-of-N Sampling	**75.1 / 79.4**
+ Best-of-N Sampling & Self-Critique	74.6 / 79.0

Table 3: Ablation study of pre- & post-processing methods on **OOD-Toolset**. The upper rows show the impact of different pre-processing steps, with the other steps remaining consistent with the original. The following rows show the impact of post-processing steps, again keeping all other steps consistent with the original.

4.4 Ablation Study

Table 3 presents the results of our ablation study. We compare a range of pre- and post-processing methods and their influence.

Pre-processing Methods. We performed the following settings: no pre-processing before generating demos, directly answering the query before generating, and query understanding before generating. The result shows that either no pre-processing or directly answering will compromise performance. Notably, the absence of pre-processing tends to yield homogenous outputs despite our introduction of randomness, potentially due to the model's

challenge in reconciling the demanded relevance and diversity. Direct answer generation also diminishes performance, as initial errors propagate, leading to more erroneous or ambiguous answers in subsequent steps. Hence, a robust pre-processing strategy enhances model performance by ensuring diverse and correct initial responses.

Post-processing Methods. We performed the following settings: no post-processing after generating two demos, self-critique after generating two, sampling the best two demos after generating five, and self-critique after sampling. In the self-critique step, we prompt the model to verify and refine the D_{gen} or D_{TopK} according to the same criteria C. However, the result indicates that LLMs are no better at verifying their own outputs, echoing the findings of Stechly et al. (2023). This also discourages us from constantly improving the quality of demos through iterative verification.

5 Discussion

5.1 Consistency when Model Scaling

Figure 3 presents the results on varying sizes of the foundation model, ranging from Llama-2-7B-Chat to Llama-2-70B-Chat. According to the results, analogical reasoning did not work on smaller models, likely due to their limited capacity to follow hard instructions. The Self-ICL method encountered similar issues, with the small models' inability to provide accurate demos compromising their effectiveness. In contrast, our method, which incorporates extra processing steps around demo generation and lowers the task difficulty, proved more adaptable even when the model is weaker (∼10B parameters). It suggests that our approach is highly adaptable and can be more effective for resource-limited or mobile scenarios.

5.2 Effectiveness Toward Complex Tasks

In the main results, we can observe that both Self-ICL and SELF-DEMOS have shown a considerable improvement on the most challenging MATH datasets. This may suggest that the methods of generating demos in advance are more effective for complex tasks, as we will detail here.

Table 4 presents the full results on the MATH dataset across different complexity levels. Analogical Prompting, as a single-step prompting method, is most effective for simple problems, showing an entirely different trend from the other methods. This aligns with our previous analysis that high model ability is required for analogical reasoning. In contrast, Self-ICL and our method significantly gain in more complex problems. With its greater focus on the relevance and correctness of demos, SELF-DEMOS outperforms others in solving the most difficult level 5 problems.

Level	Prompting Method			
	FS	Self-ICL + FS	Analog + FS	SELF-DEMOS
1	70.2	71.3 (↑ 1.6)	**80.9** (↑ 15.2)	74.5 (↑ 6.1)
2	58.9	61.9 (↑ 5.1)	**63.1** (↑ 7.1)	58.3 (↓ 1.0)
3	37.4	38.7 (↑ 3.5)	**39.9** (↑ 6.7)	39.1 (↑ 4.5)
4	28.0	**34.7** (↑ 23.9)	24.0 (↓ 14.3)	**34.7** (↑ 23.9)
5	12.4	13.8 (↑ 11.3)	11.6 (↓ 6.4)	**14.6** (↑ 17.7)

Table 4: Evaluating prompting methods on the **MATH** dataset at different complexity levels. The **Level** corresponds to problem complexity, with higher values indicating greater difficulty. The percentage of performance improvements / declines compared to the few-shot method (FS) is denoted by (↑) / (↓).

5.3 Comparing with Demo Retrieval

A key motivation for our idea is to provide relevant demos for problem-solving, without using an external retriever or demo library. So, is our approach comparable enough to retrieval-based solutions? To answer this question, we created two baselines that retrieve exemplars relevant to the given query from external data (i.e., the training set of GSM8K and MATH, which includes labeled Q&A pairs). Table 5 shows the results of these methods on two math datasets.

Undoubtedly, the retrieval-based methods perform well, with the dense retriever achieving the highest scores due to its effective representation of latent semantics (Karpukhin et al., 2020). Besides, SELF-DEMOS also shows competitive performance, especially on the MATH dataset. This could be due to more complex questions in the MATH dataset, resulting in intricate semantic connections that cannot be easily captured by a statistical algorithm like BM25 (Robertson et al., 2009). In contrast, the GSM8K dataset has more uniform and centrally distributed questions, making it more suitable for retrieval-based approaches.

Overall, SELF-DEMOS can still be a good option when resources are limited and retrieval is less feasible. Moreover, it's worth noting that the techniques of demo generation and retrieval are not mutually exclusive. Our method is particularly well-suited for a "**cold start**" and once a certain amount of demos is accumulated, we can then em-

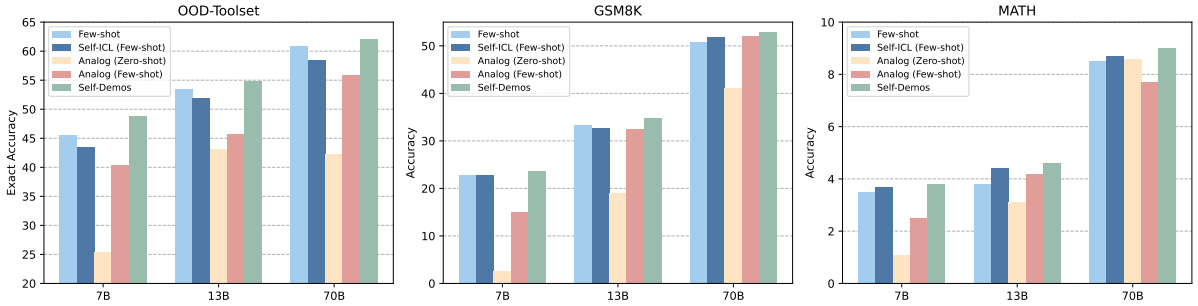
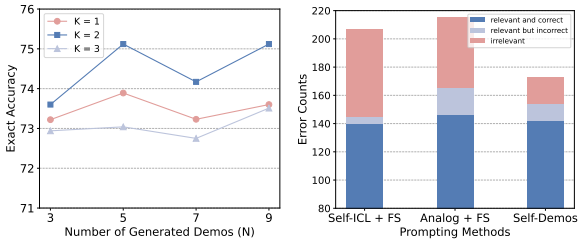


Figure 3: Performance comparison on Llama-2-Chat model family. SELF-DEMOS consistently improves performance across multiple model sizes from 7B, 13B to 70B parameters.

Demonstrating Method	Dataset	
	GSM8K	MATH
Demo Retrieval (Sparse)	79.5	37.0
Demo Retrieval (Dense)	**79.7**	**38.1**
Demo Generation (SELF-DEMOS)	78.2	37.9

Table 5: Comparison with demo retrieval methods on the **GSM8K** and **MATH** datasets. The (Sparse) means sparse retrieval using the BM25 algorithm, and the (Dense) means dense retrieval using `text-embedding-ada-002` API to generate sentence embedding and apply cosine similarity. Both baselines retrieve the Top 5 similar samples from the training set as demonstrations.



(a) (b)

Figure 4: (a) Comparison of SELF-DEMOS with varying numbers of self-generated demonstrations (N) and selected training exemplars (K). (b) Error distribution of different methods. Demos yielding incorrect answers can be categorized into three types based on relevance and accuracy. Both results are on the **OOD-Toolset**.

ploy a complementary retrieval strategy to improve efficiency and reduce incremental costs.

5.4 Number of Demonstrations Matters

We examine the impact of varying the number of self-generated demos (N) and selected demos (K) in the tool-using task. The details are shown in Figure 4a. Notably, the model performs better when selecting two demos. We suspect that a singular demo is insufficient to grasp all using patterns of an API and additional samples ($K = 3$) may introduce noise and instabilities and hinder model learning. Our configuration ($K = 2, N = 5$) not only maximizes accuracy but also ensures efficiency in computational costs. In our experiments, we further observed a tendency for the model to preferentially select demos positioned towards the front, indicating the phenomenon of position bias (Ko et al., 2020; Nori et al., 2023).

5.5 Error Analysis

Furthermore, we manually analyze the errors of SELF-DEMOS, comparing with the two baselines of demo generation in Figure 4b. Errors were categorized into three distinct types: (1) **Irrelevant demos**: These exemplars are generated in a similar distribution and fail to interpolate between seed demos and given queries. (2) **Relevant but incorrect demos**: This category includes syntactical errors and redundant or inaccurate parameters. The issues contribute to false information propagation and interfere with the final output. (3) **Relevant and correct demos**: Even with correct demonstrations, errors can occur due to the model's inherent limitations and the generalization gap. Based on Figure 4b, all three methods have similar results in Category 3 with approximately 140 errors. However, SELF-DEMOS stands out by greatly lowering the errors in the first two categories. This suggests that SELF-DEMOS is better at generating relevant exemplars, which improves generalization across novel and unseen tasks.

5.6 Computational Overhead Analysis

Our method, based on a multi-step framework, naturally leads to additional computational overhead. In Table 6, we detail this overhead for each method

Prompting Method	Cost	OOD-Toolset
Few-shot	0.54	71.9 / 76.6
Few-shot + SC (5 Paths)	2.71	72.5 / 77.2
Few-shot + SC (10 Paths)	5.41	72.2 / 77.0
Self-ICL (Few-shot)	2.37	71.5 / 76.0
Analogical Prompting (Few-shot)	1.21	71.1 / 75.4
Self-Demos (Standard)	4.81	**75.1 / 79.4**
Self-Demos (KV Cache Reuse)	2.84	**75.1 / 79.4**

Table 6: Comparison of computational costs on **OOD-Toolset**. The cost is calculated according to OpenAI price list[3], measured in dollars per thousand uses. The methods with similar costs are underlined.

and present another computationally demanding baseline, Self-Consistency, which samples various reasoning paths and generates a consistent answer using a majority vote strategy (Wang et al., 2023b). Complete calculation specifics can be found in Appendix D. Statistically, the standard SELF-DEMOS incurs a higher overhead compared to other approaches, primarily due to the demo generation phase that involves repeating the input N times to generate N demos. This leads to numerous redundant computations (i.e., KV vectors), a drawback that can be alleviated through caching and reusing (Pope et al., 2022). It can be achieved by specifying the parameter $n = N$ upon API invocation[4]. The trick cuts overhead by approximately 41%, reaching computational efficiency on par with Self-ICL and Self-Consistency (5 Paths). However, despite Self-ICL's step 2 necessitating multiple calls to model, its distinct query for each input prevents KV cache reuse (Chen et al., 2023b).

Moreover, SELF-DEMOS offer substantial **long-term cost efficiency**. When demos are limited, the use of our method does result in a higher computational overhead initially. But over time, the high-quality demos that we generate can be preserved, and when a certain amount of them is accumulated, we can apply complementary demo selection methods to reduce the incremental cost and flatten the cost curve. Refer to Appendix A for details.

6 Conclusion

This paper focuses on addressing the challenge of out-of-demonstration (OOD) queries in few-shot learning scenario. We present a novel prompting method, SELF-DEMOS, which elicits the OOD generalizability in LLMs by generating query-aware demos. Our method strategically interpolates between existing demonstrations and the OOD queries, effectively transforming them into in-demonstration (ID) queries. In an OOD setting, SELF-DEMOS achieved state-of-the-art results on the proposed OOD-Toolset and two public mathematical benchmarks. For future works, we aim to explore the scalability of the SELF-DEMOS method across diverse domains and to integrate unsupervised learning techniques to refine the quality of generated demos further.

Limitations

We summarize the limitations of our method as follows: (1) SELF-DEMOS is designed to resolve the out-of-demonstration queries, which can steadily improve downstream task performance, but the process involves additional costs. In Section 5.6, we explore the computational overhead, allowing users to make informed trade-offs depending on their specific task scenarios. (2) Our method necessitates certain capabilities of the model. Although we have done empirical experiments and demonstrated our approach works for weaker models compared to other baselines, it still requires the models to have a certain degree of instruction-following ability.

Ethics Statement

In this paper, we have followed ethical standards and principles to ensure the accuracy and validity of our research. The dataset was manually cleansed to ensure the removal of any sensitive or personal information. The human-annotated data is collected and used in compliance with relevant ethical guidelines. During the data construction process, we followed ToolAlpaca's terms under the Apache License 2.0 (Tang et al., 2023).

Acknowledgment

The authors wish to thank the anonymous reviewers for their helpful comments. This work was partially funded by National Natural Science Foundation of China (No.62206057,61976056,62076069), Shanghai Rising-Star Program (23QA1400200), Natural Science Foundation of Shanghai (23ZR1403500), Program of Shanghai Academic Research Leader under grant 22XD1401100, CCF-Baidu Open Fund, and CCF-Baichuan Fund.

[3] API Pricing - OpenAI API
[4] API Reference - OpenAI API

References

Sweta Agrawal, Chunting Zhou, Mike Lewis, Luke Zettlemoyer, and Marjan Ghazvininejad. 2023. In-context examples selection for machine translation. In *Findings of the Association for Computational Linguistics: ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 8857–8873. Association for Computational Linguistics.

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

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared 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 M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020b. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Jiuhai Chen, Lichang Chen, Chen Zhu, and Tianyi Zhou. 2023a. How many demonstrations do you need for in-context learning?

Wei-Lin Chen, Cheng-Kuang Wu, and Hsin-Hsi Chen. 2023b. Self-icl: Zero-shot in-context learning with self-generated demonstrations. *CoRR*, abs/2305.15035.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *CoRR*, abs/2110.14168.

Katherine M. Collins, Catherine Wong, Jiahai Feng, Megan Wei, and Joshua B. Tenenbaum. 2022. Structured, flexible, and robust: benchmarking and improving large language models towards more human-like behavior in out-of-distribution reasoning tasks. *CoRR*, abs/2205.05718.

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Zhiyong Wu, Baobao Chang, Xu Sun, Jingjing Xu, Lei Li, and Zhifang Sui. 2023. A survey on in-context learning.

Lingyu Gao, Aditi Chaudhary, Krishna Srinivasan, Kazuma Hashimoto, Karthik Raman, and Michael Bendersky. 2023. Ambiguity-aware in-context learning with large language models. *CoRR*, abs/2309.07900.

Danny Halawi, Jean-Stanislas Denain, and Jacob Steinhardt. 2023. Overthinking the truth: Understanding how language models process false demonstrations. *CoRR*, abs/2307.09476.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the MATH dataset. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick S. H. 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 2020, Online, November 16-20, 2020*, pages 6769–6781. Association for Computational Linguistics.

Hyuhng Joon Kim, Hyunsoo Cho, Junyeob Kim, Taeuk Kim, Kang Min Yoo, and Sang-goo Lee. 2022. Self-generated in-context learning: Leveraging auto-regressive language models as a demonstration generator. *CoRR*, abs/2206.08082.

Miyoung Ko, Jinhyuk Lee, Hyunjae Kim, Gangwoo Kim, and Jaewoo Kang. 2020. Look at the first sentence: Position bias in question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 1109–1121. Association for Computational Linguistics.

Takeshi Kojima, Shixiang (Shane) Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. In *Advances in Neural Information Processing Systems*, volume 35, pages 22199–22213. Curran Associates, Inc.

Itay Levy, Ben Bogin, and Jonathan Berant. 2023. Diverse demonstrations improve in-context compositional generalization. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 1401–1422. Association for Computational Linguistics.

Junlong Li, Zhuosheng Zhang, and Hai Zhao. 2022. Self-prompting large language models for open-domain QA. *CoRR*, abs/2212.08635.

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: The 3rd Workshop on Knowledge Extraction and Integration for Deep Learning Architectures, DeeLIO@ACL 2022,*

Dublin, Ireland and Online, May 27, 2022, pages 100–114. Association for Computational Linguistics.

Man Luo, Xin Xu, Zhuyun Dai, Panupong Pasupat, Seyed Mehran Kazemi, Chitta Baral, Vaiva Imbrasaite, and Vincent Y. Zhao. 2023. Dr.icl: Demonstration-retrieved in-context learning. *CoRR*, abs/2305.14128.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Sean Welleck, Bodhisattwa Prasad Majumder, Shashank Gupta, Amir Yazdanbakhsh, and Peter Clark. 2023. Self-refine: Iterative refinement with self-feedback. *CoRR*, abs/2303.17651.

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, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pages 11048–11064. Association for Computational Linguistics.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, Xu Jiang, Karl Cobbe, Tyna Eloundou, Gretchen Krueger, Kevin Button, Matthew Knight, Benjamin Chess, and John Schulman. 2021. Webgpt: Browser-assisted question-answering with human feedback. *CoRR*, abs/2112.09332.

Harsha Nori, Yin Tat Lee, Sheng Zhang, Dean Carignan, Richard Edgar, Nicolò Fusi, Nicholas King, Jonathan Larson, Yuanzhi Li, Weishung Liu, Renqian Luo, Scott Mayer McKinney, Robert Osazuwa Ness, Hoifung Poon, Tao Qin, Naoto Usuyama, Chris White, and Eric Horvitz. 2023. Can generalist foundation models outcompete special-purpose tuning? case study in medicine. *CoRR*, abs/2311.16452.

OpenAI. 2022. Openai: Introducing chatgpt. Website. `https://openai.com/blog/chatgpt`.

Reiner Pope, Sholto Douglas, Aakanksha Chowdhery, Jacob Devlin, James Bradbury, Anselm Levskaya, Jonathan Heek, Kefan Xiao, Shivani Agrawal, and Jeff Dean. 2022. Efficiently scaling transformer inference. *CoRR*, abs/2211.05102.

Yujia Qin, Shengding Hu, Yankai Lin, Weize Chen, Ning Ding, Ganqu Cui, Zheni Zeng, Yufei Huang, Chaojun Xiao, Chi Han, Yi Ren Fung, Yusheng Su, Huadong Wang, Cheng Qian, Runchu Tian, Kunlun Zhu, Shihao Liang, Xingyu Shen, Bokai Xu, Zhen Zhang, Yining Ye, Bowen Li, Ziwei Tang, Jing Yi, Yuzhang Zhu, Zhenning Dai, Lan Yan, Xin Cong, Yaxi Lu, Weilin Zhao, Yuxiang Huang, Junxi Yan, Xu Han, Xian Sun, Dahai Li, Jason Phang, Cheng Yang, Tongshuang Wu, Heng Ji, Zhiyuan Liu, and Maosong Sun. 2023. Tool learning with foundation models.

Stephen Robertson, Hugo Zaragoza, et al. 2009. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends® in Information Retrieval*, 3(4):333–389.

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, NAACL 2022, Seattle, WA, United States, July 10-15, 2022*, pages 2655–2671. Association for Computational Linguistics.

Zhihong Shao, Yeyun Gong, Yelong Shen, Minlie Huang, Nan Duan, and Weizhu Chen. 2023. Synthetic prompting: Generating chain-of-thought demonstrations for large language models. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 30706–30775. PMLR.

Peng Shi, Rui Zhang, He Bai, and Jimmy Lin. 2022. XRICL: cross-lingual retrieval-augmented in-context learning for cross-lingual text-to-sql semantic parsing. In *Findings of the Association for Computational Linguistics: EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pages 5248–5259. Association for Computational Linguistics.

Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R. Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, Agnieszka Kluska, Aitor Lewkowycz, Akshat Agarwal, Alethea Power, Alex Ray, Alex Warstadt, Alexander W. Kocurek, Ali Safaya, Ali Tazarv, Alice Xiang, Alicia Parrish, Allen Nie, Aman Hussain, Amanda Askell, Amanda Dsouza, Ameet Rahane, Anantharaman S. Iyer, Anders Andreassen, Andrea Santilli, Andreas Stuhlmüller, Andrew M. Dai, Andrew La, Andrew K. Lampinen, Andy Zou, Angela Jiang, Angelica Chen, Anh Vuong, Animesh Gupta, Anna Gottardi, Antonio Norelli, Anu Venkatesh, Arash Gholamidavoodi, Arfa Tabassum, Arul Menezes, Arun Kirubarajan, Asher Mullokandov, Ashish Sabharwal, Austin Herrick, Avia Efrat, Aykut Erdem, Ayla Karakas, and et al. 2022. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *CoRR*, abs/2206.04615.

Kaya Stechly, Matthew Marquez, and Subbarao Kambhampati. 2023. GPT-4 doesn't know it's wrong: An analysis of iterative prompting for reasoning problems. *CoRR*, abs/2310.12397.

Qiaoyu Tang, Ziliang Deng, Hongyu Lin, Xianpei Han, Qiao Liang, and Le Sun. 2023. Toolalpaca: Generalized tool learning for language models with 3000 simulated cases. *CoRR*, abs/2306.05301.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti

Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton-Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurélien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models. *CoRR*, abs/2307.09288.

Karthik Valmeekam, Matthew Marquez, and Subbarao Kambhampati. 2023. Can large language models really improve by self-critiquing their own plans? *CoRR*, abs/2310.08118.

Jindong Wang, Cuiling Lan, Chang Liu, Yidong Ouyang, Tao Qin, Wang Lu, Yiqiang Chen, Wenjun Zeng, and Philip S. Yu. 2023a. Generalizing to unseen domains: A survey on domain generalization. *IEEE Trans. Knowl. Data Eng.*, 35(8):8052–8072.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023b. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Jason Wei, Maarten Bosma, Vincent Y. 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 *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. 2022b. Emergent abilities of large language models. *Trans. Mach. Learn. Res.*, 2022.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. 2022c. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pages 24824–24837. Curran Associates, Inc.

Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, Rui Zheng, Xiaoran Fan, Xiao Wang, Limao Xiong, Yuhao Zhou, Weiran Wang, Changhao Jiang, Yicheng Zou, Xiangyang Liu, Zhangyue Yin, Shihan Dou, Rongxiang Weng, Wensen Cheng, Qi Zhang, Wenjuan Qin, Yongyan Zheng, Xipeng Qiu, Xuanjing Huan, and Tao Gui. 2023. The rise and potential of large language model based agents: A survey. *CoRR*, abs/2309.07864.

Michihiro Yasunaga, Xinyun Chen, Yujia Li, Panupong Pasupat, Jure Leskovec, Percy Liang, Ed H. Chi, and Denny Zhou. 2023. Large language models as analogical reasoners. *CoRR*, abs/2310.01714.

Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. 2023. Automatic chain of thought prompting in large language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Kun Zhou, Yutao Zhu, Zhipeng Chen, Wentong Chen, Wayne Xin Zhao, Xu Chen, Yankai Lin, Ji-Rong Wen, and Jiawei Han. 2023. Don't make your LLM an evaluation benchmark cheater. *CoRR*, abs/2311.01964.

Appendix

A Supplementary Experiments on GPT-4

Model	OOD-Toolset	Cost
Few-shot		
GPT-4	76.50 / 79.75	∼ 1.12
SELF-DEMOS		
GPT-4 in all steps	80.50 / 83.50	∼ 4.95
GPT-3.5 in all steps	75.50 / 79.50	∼ 0.57
GPT-3.5 reuse GPT-4 demos in step4	76.50 / 79.75	∼ 0.13

Table 7: Comparison of performance and overhead on more powerful models (i.e, GPT-4). The cost is calculated according to OpenAI price list, measured by total dollars spent on 200 instances.

We conducted GPT-4 tests on 200 random OOD-Toolset instances and used its generated demos as inputs for GPT-3.5 in SELF-DEMOS step 4, as detailed in Table 7.

Based on the results, we observe that: (1) GPT-4's advanced capabilities allow it to match the performance of GPT-3.5 using SELF-DEMOS with simply a few-shot approach. However, given the model's enhanced capabilities, it comes with a higher cost. (2) GPT-4 still benefits from the our proposed method, and the high-quality demos it generates remain effective for weaker models. This shows the reusability of demos and proves the way for SELF-DEMOS to reduce long-term costs.

B Details of OOD-Toolset

The raw data from ToolAlpaca (Tang et al., 2023) including the training and testing sets, comprises 468 tool APIs and 4,369 tool-use cases. Due to a lack of validation of the content generated by GPT-3.5, the dataset may contain specific errors, such as ambiguous queries due to outdated or insufficient information and incorrect API calls due to null or wrong values being passed. To address these issues, we implemented a data cleansing process in the following steps:

Rule-based Cleaning. We structured each tool API in the raw data into a dictionary with keys for **API Name**, **Description**, **Usage Specification**, and **Tool-use Cases**. The API name identifies the tool, and the description outlines its purpose. The usage specification clarifies the API call format and required parameters. The tool-use cases consist of user queries and corresponding function call lists. The rule-based cleaning process involved:

- We removed entries with missing keys and formatting errors, particularly those that did not follow the JSON format in the function calls.
- We removed user queries that required more than three function calls to be resolved due to their complexity.
- We removed parameters not directly related to the core functionality of tools, such as API keys and sensitive user information.
- We removed tools with fewer than 3 instances or fewer than 3 functions to ensure that OOD scenarios could be built.

After the first cleaning round, a total of 322 tools and 2,788 instances remained.

Manual Data Cleaning. In manual data cleaning, we emphasize the solvability of given queries. The manual data cleaning process involved:

- We strive to minimize dependencies between function calls, avoiding scenarios where a subsequent function call relies on the results returned by preceding ones. This is to ensure that these queries can be answered in a round of dialog.
- While we avoided the exposure of sensitive user information, some necessary parameters within function calls, such as the email address in the email API, are subjected to obfuscation using a placeholder, for instance, *user@example.com*.
- Time and location information should be explicitly mentioned in the queries, avoiding the use of ambiguous pronouns such as 'today', 'tomorrow', and 'my home'.
- We confirmed the consistency of parameter values with their data types as defined in the usage specifications.

After the second cleaning round, the dataset comprised 321 tools and 2,625 instances. Table 8 presents an illustrative example of the cleaned dataset.

Query and Demonstration Construction. After two rounds of data cleaning, the correctness and solvability of the data have been ensured. Then, we proceeded to select instances from the tool-use cases and construct corresponding demonstrations. During the selection process, we tended to choose longer instances as queries, considering them to be more challenging. Following that, we randomly sampled three other instances from the remaining use cases of the same tool as demos. Note that the sub-APIs to be called for the demos should be different from those required for the chosen queries to fulfill the OOD settings.

Finally, we obtained a set of 1,057 queries, forming our testing set. Table 9 presents an instance of OOD-Toolset.

C Prompt Templates

The prompt templates of SELF-DEMOS for each step in tool-using tasks are presented in Table 10, 11, 12, and 13. Similarly, the prompt templates in mathematical problem-solving tasks are presented in Table 14, 15, 16, and 17.

D Details of Computational Overhead

The details about the computational overhead of each methods are shown in Table 18.

E Case Study

Even SELF-DEMOS performs better than all other methods, there are instances where it falied while others succeeded. We have picked up 3 representative cases for further analysis: (1) SELF-DEMOS succeeded while few-shot / Self-ICL failed, (2) few-shot succeeded while SELF-DEMOS failed, and (3) both failed. Due to space constraints, we put the full case study in our GitHub repository.

API Name: MAP

Description: MAP API is used for calculating distances, planning routes, and locating points.

Usage Specifications:
DISTANCE: Calculate the distance between two points.
Parameters: {"start": "Required. String. The starting point for the distance calculation.", "target": "Required. String. The destination point for the distance calculation."}

ROUTE: Generate a travel route between two points.
Parameters: {"start": "Required. String. The starting point for the route.", "target": "Required. String. The destination point for the route."}

SEARCH: Locate nearby points within a set distance.
Parameters: {"target": "Required. String. The target point to search around.", "position": "Required. String. The current position of the user.", "distance": "Required. Integer. The search radius in kilometers."}

Tool-use Cases:
Query: How far is Beijing to Shanghai?
Function calls: [DISTANCE(start="Beijing", target="Shanghai")]

Query: How many shops are around Times Square in 3km?
Function calls: [SEARCH(target="shop", position="Times Square", distance=3)]

Query: Show me the route from Los Angeles to San Francisco.
Function calls: [ROUTE(start="Los Angeles", target="San Francisco")]

Query: Are there any bookstores within 5km of Central Park?
Function calls: [SEARCH(target="bookstore", position="Central Park", distance=5)]

Query: How do I drive from Big Ben to Tower Bridge, and then to the London Eye?
Function calls: [ROUTE(start="Big Ben", target="Tower Bridge"),
ROUTE(start="Tower Bridge", target="London Eye")]

Query: What's the distance from my home at 123 Main St to the grocery store at 456 Oak St, and from there to my office at 789 Pine St?
Function calls: [DISTANCE(start="123 Main St", target="456 Oak St"),
DISTANCE(start="456 Oak St", target="789 Pine St")]

Table 8: An illustrative example of the cleaned dataset, composed of four parts: **API Name**, **Description**, **Usage Specifications**, and **Tool-use Cases**. Among them, the tool-use cases are stored as lists.

Seed Demos:
Query: How far is Beijing to Shanghai?
Function calls: [DISTANCE(start="Beijing", target="Shanghai")]

Query: How many shops are around Times Square in 3km?
Function calls: [SEARCH(target="shop", position="Times Square", distance=3)]

Query: Are there any bookstores within 5km of Central Park?
Function calls: [SEARCH(target="bookstore", position="Central Park", distance=5)]

Query: How do I drive from Big Ben to Tower Bridge, and then to the London Eye?

Table 9: An instance of OOD-Toolset corresponds to the tool in Table 8, where the function required for the **Query** is ROUTE. Consequently, tool-use cases related to this sub-API should not be included in the **Seed Demos**.

The {tool_name} API is used for {description}. In this task, you need to give a general understanding of the user query and determine which function should be called to solve the query.

 # Tool Specification:
{specification}

 # User Query:
{query}

 # Instruction:
Generate a general understanding here. In particular, you need to explicitly indicate the name of the function that should be called.

Table 10: Prompt template for Query Understanding (Step 1) on the OOD-Toolset.

The {tool_name} API is used for {description}. In this task, you need to give an example of when to use the API based on the specification.

 # Tool Specification:
{specification}

 # Demonstration:
{seed_demos}

 # Instruction:
Generate an example of how to use the {function_mentioned_in_step1} function.
- After "Query: ", describe the user query.
- After "Function Calls: ", give the function calls in the format of ["function_name(parameter=value)"].

Table 11: Prompt template for Query-aware Demo Generation (Step 2) on the OOD-Toolset.

The {tool_name} API is used for {description}. Here are some examples of how to use the API. In this task, you must check the examples for correctness and select one or two best examples to keep.

 # Tool Specification:
{specification}

 # Check List:
- Syntax errors: the function calls should conform to the format like "function_name(parameter=value)".
- Redundant parameters: the function calls must conform to the parameter list in the tool specification.
- Value passing errors: the values of parameters should be of the correct type and reasonable.
- Unsolvable errors: the query should be solvable with the given function.

 # Examples to be Checked:
{generated_demos}

 # Instruction:
Select one or two best examples to keep. If there are not enough correct examples, just keep one.
For your output:
- After "Selection: ", give the serial numbers of your choice in the format of <x>, <y>.
- After "Explanation: ", give the reason why you keep the examples.

Table 12: Prompt template for Best-of-N Sampling (Step 3) on the OOD-Toolset.

The {tool_name} API is used for {description}. In this task, you must generate the function calls for a given query.

 # Tool Specification:
{specification}

 # Demonstration:
{seed_demos}
{selected_demos}

 # Instruction:
Solve the following user query.
Query: {query}
Function calls: Give your answer in the format of ["function_name(parameter=value)"].

Table 13: Prompt template for Response Generation (Step 4) on the OOD-Toolset.

In this task, you need to give a general understanding of mathematical problems, which can be applied to all similar questions in the same scenario.

 # Problem:
{question}

 # Instruction:
Give a general understanding of this problem in one line. Highlight the general solution methodologies to solve this type of problem. Focus on the problem-solving approach without delving into specific numerical values or answers.
You can refer to this template for your understanding: This problem involves...To solve this type of problem...

Table 14: Prompt template for Query Understanding (Step 1) on the GSM8K and MATH datasets.

In this task, you need to recall mathematical problems. When presented with a math problem, recall another relevant problem as an example. The example should help answer the initial problem.

 # Problem:
The initial problem:
{question}

The understanding you can refer to:
{understanding}

 # Demonstration:
{seed_demos}

 # Instruction:
Recall one example of a math problem relevant to the initial problem. The example should be distinct from the initial problem (e.g., involving different numbers and names).
- After "Question: ", describe the problem you generate in one line.
- After "Answer: ", explain the step-by-step solution and enclose the ultimate answer in \boxed{}.

Table 15: Prompt template for Query-aware Demo Generation (Step 2) on the GSM8K and MATH datasets.

In this task, you need to check the correctness of these math Q&A pairs and select one or two best examples to keep for answering the initial problem.

 # The initial problem:
`{Question}`

 # Check List:
- The calculation process in the solution must be correct and without ambiguity.
- The examples should be relevant and helpful in solving the initial problem.

 # Examples to be checked:
`{generated_demos}`

 # Instruction:
Select one or two best examples to keep. If there are not enough correct and helpful examples, just keep one.
For your answer:
- After "Selection: ", give the serial numbers of your choice in the format of <x>, <y>.
- After "Explanation: ", give the reason why you keep this example.

Table 16: Prompt template for Best-of-N Sampling (Step 3) on the GSM8K and MATH datasets.

Your task is to tackle mathematical problems step by step. You can refer to these demonstrations to give your reasoning process.

 # Demonstration:
`{seed_demos}`
`{selected_demos}`

 # Instruction:
Solve the following problem step by step.
Question: `{Question}`
Answer: Explain the step-by-step solution and enclose the ultimate answer in \boxed{} here.

Table 17: Prompt template for Response Generation (Step 4) on the GSM8K and MATH datasets.

Prompting Method	Avg. #tokens of Input	Avg.#tokens of Output	Cost	OOD-Toolset
Few-shot	496.0	22.6	0.54	71.9 / 76.6
Few-shot + SC (5 Paths)	$496.0 \times 5 = 2480.0$	$22.6 \times 5 = 113.0$	2.71	72.5 / 77.2
Few-shot + SC (10 Paths)	$496.0 \times 10 = 4960.0$	$22.6 \times 10 = 226.0$	5.41	72.2 / 77.0
Self-ICL (Few-shot)	$456.4 + 498.4 \times 2 + 625.1 = 2078.3$	$78.7 + 23.6 \times 2 + 22.2 = 148.1$	2.37	71.5 / 76.0
Analogical Prompting (Few-shot)	598.0	304.5	1.21	71.1 / 75.4
Self-Demos (Standard)	$323.6 + 490.8 \times 5 + 776.4 + 606.4 = 4160.4$	$3.4 + 58.0 \times 5 + 7.7 + 22.5 = 323.6$	4.81	**75.1 / 79.4**
Self-Demos (KV Cache Reuse)	$323.6 + 490.8 + 776.4 + 606.4 = 2197.2$	$3.4 + 58.0 \times 5 + 7.7 + 22.5 = 323.6$	2.84	**75.1 / 79.4**

Table 18: Average number of input and output tokens of different methods on **OOD-Toolset**. In the equation, each term being added represents the average number of tokens per step (used only within a multi-step framework), while each multiplier indicates the number of times that step is called.