Open Grounded Planning: Challenges and Benchmark Construction

Shiguang Guo[1,3]* **Ziliang Deng**[1,3]* **Hongyu Lin**[1]† **Yaojie Lu**[1]†
Xianpei Han[1,2,4] **Le Sun**[1,2,4]
[1]Chinese Information Processing Laboratory [2]State Key Laboratory of Computer Science
Institute of Software, Chinese Academy of Sciences, Beijing, China
[3]University of Chinese Academy of Sciences, Beijing, China
[4]Key Laboratory of System Software, Chinese Academy of Sciences
{guoshiguang2021,dengziliang2021,hongyu,luyaojie}@iscas.ac.cn
{xianpei,sunle}@iscas.ac.cn

Abstract

The emergence of large language models (LLMs) has increasingly drawn attention to the use of LLMs for human-like planning. Existing work on LLM-based planning either focuses on leveraging the inherent language generation capabilities of LLMs to produce free-style plans or employs reinforcement learning approaches to learn decision-making for a limited set of actions within restricted environments. However, both approaches exhibit significant discrepancies between the open and executable requirements in real-world planning. In this paper, we propose a new planning task—open grounded planning. The primary objective of open grounded planning is to ask the model to generate an executable plan based on a variable action set, thereby ensuring the executability of the produced plan. To this end, we establish a benchmark for open grounded planning spanning a wide range of domains. Then we test current state-of-the-art LLMs along with five planning approaches, revealing that existing LLMs and methods still struggle to address the challenges posed by grounded planning in open domains. The outcomes of this paper define and establish a foundational dataset for open grounded planning, and shed light on the potential challenges and future directions of LLM-based planning. Our code and datasets are at https://github.com/Shiguang-Guo/Open-Grounded-Planning

1 Introduction

Human life is filled with tasks of varying complexities, from simple activities like brewing coffee to more substantive pursuits such as learning new skills. By utilizing our understanding of the world, we can formulate plans for tasks and execute these steps in sequence. Although we can employ innumerable strategies and plans to achieve our objectives, the scenario is significantly more complex for artificial intelligence. Grounding plans to open action sets for tasks in open domains poses one of the challenges for AI.

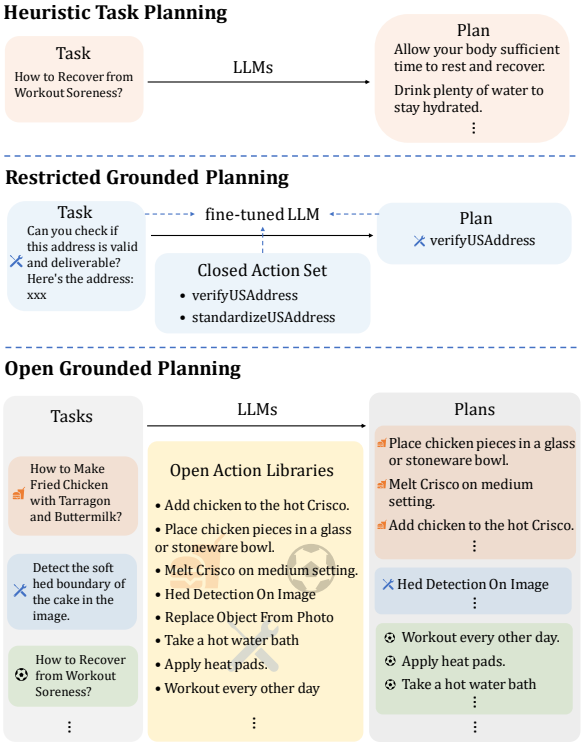


Figure 1: **Heuristic Task Planning**: Free and arbitrary planning. **Restricted Grounded Planning**: Domain-specific planning on small action sets, usually given in a context window. **Open Grounded Planning**: Planning on extensive action sets in various domains.

Some prior research has delved into the planning ability of Large Language Models (LLMs) and found that LLMs can engage in planning to some extent using their internal knowledge through

*Equal contribution
†Corresponding author

Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 4982–5003

common-sense reasoning (Zhao et al., 2023; Brown et al., 2020). However, these plans are often heuristic, coherent, and rational in natural language, yet possess a high degree of freedom and cannot serve as executable instructions for AI agents (Yao et al., 2023; Huang et al., 2022). In other words, these plans are not grounded in an actionable space. In addressing the issue of grounded planning, various approaches have been explored in fields like robot controlling (Ahn et al., 2022; Wang et al., 2023b) and tool use (Qin et al., 2023; Li et al., 2023; Tang et al., 2023). Typically, model fine-tuning is applied for performance improvement in certain restricted scenarios (Song et al., 2023; Shen et al., 2023; Yuan et al., 2023). However, these methods can only enable models to perform planning on a limited set of actions for specific domain tasks (Lin et al., 2023; Wu et al., 2023; Hao et al., 2023). As the task domain becomes broad, the action space becomes vast and open, these grounded planning methods appear too restricted to handle the steeply increased complexity (Wang et al., 2023b).

The capability to perform a wide range of actions and to devise viable, comprehensive plans by selecting suitable actions from an extensive action library for tasks in various domains epitomizes both a vision and a future trend in LLMs. Consequently, in this work, we introduce the concept of ***Open Grounded Planning*** to advance the research on LLM planning across broad fields and a rich array of potential actions. We delineate the concept in two distinct dimensions:

- **Grounded Planning:** LLM is required to compose plans utilizing only the actions available within the executable action sets.
- **Open Planning:** We aspire for the model to conduct planning within an extensive set of actions in an open domain that contains various task fields.

Moreover, we collect datasets from three major areas, including daily life, tool use, and robot sandbox scenarios. All collected datasets have been transformed into a uniform format, including task objectives, constraint conditions, golden steps, and candidate action sets. Building upon this foundation, we have developed a benchmark to assess the performance of various models and methods in the Open Grounded Planning task.

Besides, to address the open grounded planning challenges, we proposed a novel *Retrieve and Rewrite* framework. The method utilizes the LLMs to generate an initial plan and iteratively rewrites this plan using actions retrieved based on the current planning situation.

We conducted comprehensive experiments on four commonly used methods and our *Retrieve and Rewrite* method for current planning tasks using GPT-3.5, Vicuna-7B, and LLaMA-2-7B fine-tuned with a small amount of domain knowledge. The explored methods include retrieval-based methods and inference-based methods. We observed that fine-tuning contributes much to bridge the gap between smaller models and extremely large-scale language models by raising the instruction following and task understanding abilities. Various methods exhibited trade-offs regarding the executability and quality of generated plans. Generalizing ability from in-domain to out-of-domain planning tasks exists to a certain extent.

Generally speaking, our contributions are:

- We proposed the concept of Open Grounded Planning. We envision future artificial intelligence systems being able to plan tasks within open domains, and having the ability to ground plans onto open executable action sets.
- We constructed a benchmark consisting of datasets from diverse domains for Open Grounded Planning and an automated evaluating procedure to assess the performance of different models and methods.
- We introduced the *Retrieve and Rewrite* framework to address challenges in Open Grounded Planning tasks, and conducted comprehensive experiments on state-of-the-art models with various methods, and found that current models and methods still struggle with Open Grounded Planning tasks.

2 Related Work

Large language models are trained on data containing extensive common knowledge and exhibit certain planning and common-sense reasoning abilities (Zhao et al., 2023; Brown et al., 2020). Prompting can be employed to guide large language models in generating plans for given tasks, and these plans often possess a high degree of freedom, making them challenging to execute in specific environments (Huang et al., 2022). To facilitate the

grounding of generated plans to an AI agent's executable action space, prior research has extensively explored grounded planning tasks (Lin et al., 2023; Wu et al., 2023). Some approaches opt for a global planning strategy based on the task, aiming to directly generate plans that can be grounded to the execution environment in a single step (Song et al., 2023; Shen et al., 2023; Yuan et al., 2023; Wang et al., 2023a). Conversely, other methodologies employ iterative interactive approaches as the primary means of plan generation to adapt to changes in the environment and conditions (Ahn et al., 2022; Wang et al., 2023b). However, these approaches often demonstrate limited effectiveness, completing constrained tasks with a finite set of actions within a singular domain.

In open-domain environments, the enormity of tasks and action sets poses significant challenges, making it increasingly difficult to bridge the gap between plans generated by large language models and the execution of real-world tasks. Therefore, in our work, we raise the challenge of open grounded planning and compile benchmark data from multiple domains ranging from everyday life to tool use and robot control scenarios, which consist of tens of thousands of tasks and actions. We also utilized our benchmark to assess the performance of mainstream proprietary models and open-source models with various planning methods on open grounded planning tasks.

3 Open Grounded Planning Benchmark

In this section, we initially present the task definition of Open Grounded Planning and its associated challenges. Subsequently, we introduce the Open Grounded Planning Benchmark, encompassing the construction of the dataset, evaluation metrics, and the methodology for automated assessment.

3.1 Definition of Open Grounded Planning

Specifically, for a given task objective G stemming from any domain, along with conditional constraints C (which may be absent for task without additional constraints), we aim to find a plan P composed of a series of actions $\{s_i\}$, where each action s_i is from an open action set S. In other words, the generated plan P must be grounded onto the action set S which is vast and extendable:

$$P = (s_1, s_2, \cdots, s_n | G, C), s_i \in S, 0 \leq i \leq n$$

where n is the length of P. Table 1 shows the grounded planning process.

Task:
How to Activate the Dark Theme on YouTube
Method:
Using the YouTube App for Android
Action Candidate Set:
* Close the Tool Options window.
* Double click the file.
* Do price forecasting.
* Click on the blue coloured YOUTUBE STUDIO BETA button.
* Open the YouTube app on your iPhone or iPad.
* Launch the YouTube app on your Android device.
* <other steps>...
Steps:
1. Launch the YouTube app on your Android device.
2. Tap on your profile picture.
3. Tap on Settings.
4. Select the General option.
5. Tap on the grey switch, right across Dark theme text.
6. Enjoy YouTube in dark mode

Table 1: An example of an Open Grounded Planning task. LLM needs to select appropriate actions from a complex and huge set of actions to generate a plan to complete the task.

As discussed in Section 2, many explorations into LLM planning are focusing on heuristic planning in which the generated plans cannot be directly used as instructions for downstream control mechanisms, in other words, they are not "grounded". Some previous studies have demonstrated that LLMs can undertake grounded planning tasks in certain fields. However, these applications have often been limited to constrained scenarios and task domains. As the richness of the task domains and actions increases, the model's planning proficiency tends to diminish. LLMs still face challenges in executing grounded planning across open domains, which encompass a wide array of tasks and actions from diverse fields.

3.2 Dataset Construction

LLM's planning capabilities have a variety of application scenarios. We refer to many other works and summarize the three main application areas including daily life, tool usage, and robots. To balance the proportions of data across different categories, we retain a maximum of 500 tasks for each category, forming our evaluation set. All actions related to the original tasks are preserved in the action library as candidate actions.

We split the dataset into two parts. We employ the daily life dataset wikiHow to evaluate the in-domain grounded planning capabilities because this dataset covers a very wide range and the action set for selection is more complex. Additionally, we utilize datasets related to tool use and robots

Category	Eval-Set	Full-Set	Actions	Category	Eval-Set	Full-Set	Actions
wikiHow(Zhang et al., 2020)							
Arts and Entertainment	500	4104	26222	Home and Garden	500	6916	39872
Cars and Other Vehicles	500	1685	10929	Personal Care and Style	500	3888	20786
Computers and Electronics	500	12801	75186	Pets and Animals	500	2282	11056
Education and Communications	500	5485	29856	Philosophy and Religion	500	748	5000
Family Life	500	1532	8634	Relationships	500	1683	8609
Finance and Business	500	4376	24746	Sports and Fitness	500	1898	10916
Food and Entertaining	500	9493	58585	Travel	500	852	5433
Health	500	7918	37364	Work World	500	1088	6618
Hobbies and Crafts	500	7095	47168	Youth	500	1477	8389
Holidays and Traditions	500	904	5658				
Tools				Robot			
APIBank(Li et al., 2023)	263	263	101	SayCan(Ahn et al., 2022)	164	164	97
GPT4Tools(Yang et al., 2023)	500	1750	32	VitualHome(Huang et al., 2022)	500	5088	47522
ToolAlpaca(Tang et al., 2023)	201	201	89				

Table 2: Statistics of the Open Grounded Planning benchmark, marking the quantitative attributes of the in-domain and out-of-domain datasets.

to evaluate the generalization of various models and methods for out-of-domain grounded planning. The statistical information of the evaluation set can be found in Table 2. We also provide the more detailed data processing procedure in Appendix A.

3.2.1 In-Domain Datasets

Wikihow Wikihow is an extensive collection of guides and tutorials, encompassing topics ranging from everyday life skills to more complex subjects[1]. Each guide on WikiHow is presented in a step-by-step manner, making it easy to understand and follow. We gathered the original corpus of WikiHow by referencing Zhang et al. (2020). For each article, we retained only the tasks, methods (if any), and headlines. We eliminated sections containing multiple "parts" as they introduced additional hierarchy. By directly utilizing the original categorization within the WikiHow corpus, we ultimately identified 19 categories, with a total of more than 76,000 tasks. The action libraries for each category are derived from the collective actions of all tasks within the same category, with an average size exceeding 20,000.

3.2.2 Out-of-Domain Datasets

Tools Previous studies have demonstrated the capability of LLMs to utilize tools to accomplish tasks. Effective planning is crucial for tool use, especially when the candidate toolset is extensive. We have collected open-source data relevant to tool usage by LLMs, including contributions from ToolAlpaca (Tang et al., 2023), API-Bank (Li et al., 2023), and GPT4Tools (Yang et al., 2023). These datasets encompass various types of tools and provide standard tool invocation sequences to complete the tasks as well. To maintain consistency with other datasets, we only retain the API names and their corresponding description, while ignoring the parameters.

Robot There exists some research related to grounded planning in robotics(Yoshida et al., 2023; Brohan et al., 2023; Ahn et al., 2022), but there is still a lot of room for development. We have converted datasets proposed in VirtualHome (Huang et al., 2022) and SayCan (Ahn et al., 2022) and merged all executable actions as a candidate action set[2]. It is important to note that complete robot processing involves multiple stages, including visual information processing and action execution. Our dataset, however, only focuses on the planning generation.

3.3 Evaluation

3.3.1 Plan Quality Assessment

In all the datasets we collected, every task has a corresponding golden plan which is provided by the original datasets and presents one of the possible ways to handle the task. Since the solutions to the tasks in our benchmark could be quite diverse, especially when it involves thousands of candidate operations to form a planned execution path, it is unfair to directly judge whether the generated plans match exactly with the golden plans. Instead, we

[1] https://www.wikihow.com/

[2] There seems to be some mismatch in the dataset provided by saycan, we fix it manually. A more detailed procedure is provided in Appendix A.

make the golden plan a reference and compare the plan generated by the model with it from multiple perspectives to judge which plan is better. The specific evaluation criteria are as follows:

Completeness: Examine whether the plan is comprehensive, with a focus on the coherence and logic between steps, and the avoidance of arbitrarily introduced conditions and missing steps.

Feasibility: Assess the practicality of the plan, considering whether each step can be implemented, whether the plan aligns with common sense, adheres to human ethical standards, and avoids excessive redundant steps.

Relevance to the Task: Evaluate the relevance of the plan to the given task, considering the utilization of the provided task conditions and whether it achieves the goal.

We use ChatGPT to evaluate, which is a widely used evaluation method in previous similar work. During one evaluation, the target task to be solved is delivered to the evaluator along with the model generated plan and the golden plan. Then we prompt ChatGPT to list how good the two plans are regarding the evaluation criteria, and ask it to elect a better plan based on its analyses.

Due to many reported issues with ChatGPT as an evaluator, such as position bias, length preference, and style partiality (Koo et al., 2023; Wu and Aji, 2023; Zheng et al., 2023a), we employ various methods to mitigate those biases. We swap the order of the two plans and average the scores to eliminate positional bias. Additionally, we prompt ChatGPT to penalize the score for redundant steps to reduce length preference. We sampled a small dataset for manual evaluation and verified the plausibility of ChatGPT's automatic evaluation. For detailed analysis, please refer to appendix C.

3.3.2 Metrics

To more intuitively compare the performance of various models and methods on open domain planning datasets, we define the following metrics to quantify their performance.

Executability is the proportion of executable cases. Executable cases are actions in the plan that all exist within the given action library.

Quality of the executable plans is evaluated from the dimensions in section 3.3.1. We define win rate as the average of the outcomes of two comparisons involving position swaps. Intuitively, quality assesses how complete the generated plan is.

Overall Pass Rate is the proportion of all generated plans that can be executed while also completing the task. Considering both executability and quality, we choose pass rate as the final evaluation metric to evaluate the overall performance of the model in the entire process. The pass rate is the product of executability and quality.

$$\text{Executability} = \frac{\#\text{executable cases}}{\#\text{all cases}}$$

$$\text{Quality} = \frac{\#\text{win cases}}{\#\text{executable cases}}$$

$$\text{Pass Rate} = \text{Executability} \times \text{Quality}$$

4 Methods

In order to assess the performance of current mainstream models on the Open Grounded Planning task, we endeavored to employ five distinct methods including *Retrieve and Rewrite*, a new framework we proposed, to address this challenge.

Task-Retrieve: We first adopt a simple and intuitive approach by using the task name as the query for action retrieval. Given a task T and an action set A, we retrieve the relevant action list $A' = \text{retrieve}(T, A)$. LLM selects and orders appropriate actions from this list to generate plan $P = \text{select\&sort}(T, A')$.

Plan-Retrieve: Simply searching by task name makes it difficult to recall important steps that are not directly related to the task name. We try to specify the query to improve. We first force LLM to generate an initial plan $P_0 = \{s_1, s_2, \cdots, s_n\}$ for the task, then retrieve related actions based on the generated plan to get action list $A' = \text{retrieve}(P_0, A)$. Finally, we perform similar selection and rearrangement as in task-retrieve. We also provide the initial plan for LLM to refer to generate the final plan $P = \text{select\&sort}'(T, A', P_0)$.

Step-wise Select: In addition to retrieval and rearrangement, we also try the step-wise selection method. We adopt a method similar to ReAct (Yao et al., 2023) to our tasks. Each time, LLM generates a possible next step s_p for the given task T, Then we obtain a candidate action list $A' = \{a_1, a_2, ...\} \subset A$ by retrieving actions based on the generated steps. LLM selects one of the retrieved results as the next step, which means the target plan P is generated step by step, i.e., $P = \{s_1, s_2, ...\} \cup \{a_j\}$ where $a_j \in A'$. Selection iteration repeats until 1) LLM outputs *None* when generating the possible next step, 2) LLM

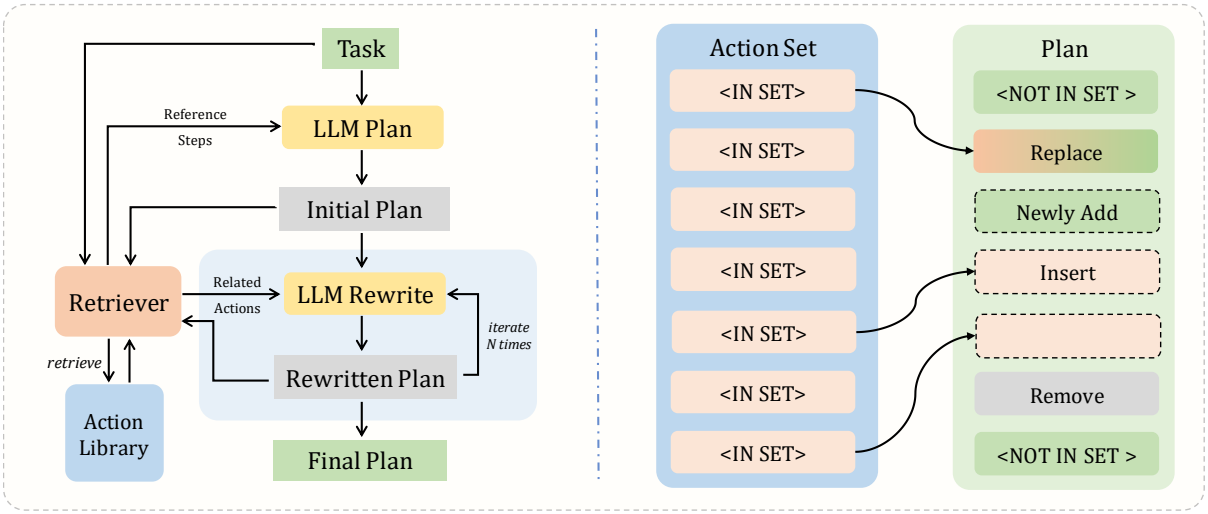


Figure 2: **Left**: The Retrieve and Rewrite framework. **Right**: An illustration of different rewriting operations.

refuses to select from the candidate list, and 3) the maximum number of iterations is reached.

DFS: The *Step-wise Selecting* method suffers from small searching space. ToolLLM (Qin et al., 2023) proposed a DFS-based method to improve. We implement this by simply extending the *Step-wise Selecting* method. During the procedure of model selecting the next step, we allow LLM to abandon the selection if it thinks there exists no suitable choice in the retrieved candidate actions to be the next step of the current plan. In this case, we perform a backtracking.

Retrieve and Rewrite: We realize that methods based on retrieval and rearrangement can consider the overall plan, but may not obtain the optimal choice for each step. The step-wise selection approach enables adjustments during the generation process but does not take into account the plan as a whole. We combine the advantages of both methods and propose a new method named *Retrieve and Rewrite*.Figure 2 illustrates its framework and different rewriting operations.

LLM is first asked to generate an initial plan $P_0 = \{\underline{s}_1, \underline{s}_2, ..., \underline{s}_{n_0}\}$ based on the relevant steps A_0 with the given task T. Different from the *Task-Retrieve* method, P_0 does not have to be composed of the exact steps in A_0. We mark steps not in action set with underline. We perform several iterations to use steps from the action set to rewrite P_0. For iteration $i \geq 1$, we choose some of the steps not in action set for retrieval to retrieve relevant actions $A_i = \{a_{i1}, a_{i2}, \cdots, a_{im_i}\}$, where m_i is the length of candidate action list of iteration i. LLM is allowed to perform various operations when rewriting P_{i-1}, including adding, deleting, and modifying arbitrarily. We only need to ensure using actions *in* the action set to replace as much as possible those *not in* the action set. The plan after rewriting might be $P_i = \{s_1, \underline{s}_2, s_3, \cdots, \underline{s}_{n_i}\}$, where n_i is the new length of current plan P_i. Similarly, iteration stops until all actions are in the action set or iteration reaches the maximum number.

5 Experiment

We systematically assess the capabilities of various LLMs and methods in the Open Grounded Planning task by selecting and comparing mainstream proprietary models, open-source models, and open-source models fine-tuned with a small amount of domain-specific data. For each model, we examine the performance of different methods in Section 4. We test the models' abilities in Open Grounded Planning on both in-domain, the wikiHow dataset, and out-of-domain, the tools and robot datasets.

5.1 Experiment Settings

We chose the proprietary model GPT-3.5[3] and the open-source model Vicuna-7B-v1.5-16k (Zheng et al., 2023b) for experiments. In addition to this, we fine-tuned Llama-2-7B (Touvron et al., 2023) to check the performance of the SFT model. We believe these three models can represent the capabilities of current mainstream models.

We select 200 tasks from each subcategory below wikiHow as the training set. For each setting,

[3]We use *gpt-3.5-turbo-1106* for our experiments.

	In-Domain (wikiHow)			Out-of-Domain (Tools, Robot)				
	Average of All Types			APIBank	GPT4Tools	ToolAlpaca	VirtualHome	SayCan
Method	Executability(%)	Quality(%)	**Pass Rate(%)**	**Pass Rate(%)**	**Pass Rate(%)**	**Pass Rate(%)**	**Pass Rate(%)**	**Pass Rate(%)**
Vicuna-7B-v1.5-16k								
Task-Retrieve	89.60	27.87	24.97	36.50	16.30	19.90	12.40	14.94
Plan-Retrieve	67.77	42.41	28.74	26.05	15.10	9.70	11.90	12.20
Step-wise Select	73.17	12.34	9.03	20.72	15.80	10.20	7.10	1.83
DFS	97.92	7.18	7.03	26.81	20.50	16.17	6.80	2.44
Retrieve and Rewrite	80.75	34.88	28.17	45.44	22.30	23.63	16.00	12.50
GPT-3.5								
Task-Retrieve	95.99	44.43	42.65	37.26	23.20	14.93	26.50	29.27
Plan-Retrieve	69.46	60.15	41.78	30.61	32.30	11.94	37.60	**44.82**
Step-wise Select	93.44	21.21	19.82	32.32	23.50	16.67	30.60	26.83
DFS	98.84	50.76	50.17	35.55	25.00	19.90	32.90	11.28
Retrieve and Rewrite	92.98	58.72	54.60	43.73	26.60	28.86	**47.00**	41.16
LLaMA-2-7B(SFT)								
Task-Retrieve	99.40	47.66	47.37	**58.75**	26.50	**49.00**	37.50	37.50
Plan-Retrieve	99.13	58.21	57.70	47.72	35.00	36.07	42.70	30.79
Step-wise Select	99.82	24.26	24.22	36.12	4.20	21.89	34.10	31.71
DFS	99.09	53.53	53.04	11.40	0.48	2.79	35.64	14.96
Retrieve and Rewrite	98.26	61.58	**60.51**	45.42	**43.70**	45.02	46.80	42.68

Table 3: The average performance of models and methods on in-domain and out-of-domain datasets. The final metric is **Pass Rate**. The best performance score of each dataset is highlighted with bold, while the second-best underlined.

we perform the inference process of GPT-3.5 on it and select those with high quality for training. We mixed it with the Alpaca dataset (Taori et al., 2023) and fine-tuned the model with 3 epochs to improve the generalization ability. We use OpenAI's *text-embedding-ada-002* to generate embedding for each step in all settings. Additional implementation details are in Appendix B and all prompts are in Appendix F.

5.2 In-Domain Results

We measure the performance of each model and method using metrics in Section 3.3.2. We report the average performance on all wikiHow datasets. Results on in-domain datasets are presented in the left part of Table 3, from which we can derive:

SFT model achieves the best performance Compared to Vicuna and GPT-3.5, the SFT model surpasses them in all methods. The trained model can improve the executability of all methods to close to 100% and maintain high quality at the same time.

Different methods have different focuses Although the initial plan of the Plan-retrieve method may cause interference, it can generate a better plan than Task-retrieve. Compared with the restricted search space in the step-wise selection, DFS usually achieves a higher executability and has better quality. Besides, we find pre-planning grants the final plans higher quality. The Retrieve and Rewrite method we proposed surpasses Step-wise select and DFS in terms of rationality and completeness of the final plans due to our pre-planning approach and the subsequent rewriting.

5.3 Out-of-Domain Results

The experiment results on the out-of-domain datasets are presented in the right part of Table 3. Due to space constraints, we provide more detailed OOD dataset results in Appendix E. Apart from the DFS method, LLaMA-2-SFT outperforms GPT-3.5 on most datasets using other methods. This suggests that fine-tuning is beneficial for the model to generalize from in-domain grounded planning to out-of-domain scenarios, primarily by enhancing the model's ability to follow instructions, thereby improving task completion. In the tools dataset, despite our extensive prompt engineering, GPT-3.5 prefers to generate complete steps rather than the provided API names, which results in poor performance.

We observe the performance degradation in SFT model with DFS and find that the average length of the plans generated with DFS on tools dataset is less than 1, while with other methods ranges from 1 to 2, which is also closer to the true length of datasets. We observe extensive backtracking in the generated results,which could be attributed to the higher proportion of rejection in the training data, leading the SFT model to exhibit reduced confidence in its generation. Consequently, this makes LLM more inclined not to select the retrieved re-

sults as the next step when facing OOD data.

Among various methods, Rewrite method consistently maintains strong performance. However, although it performs well in-domain, the generalization performance is sometimes not as good as simple retrieval and rearrangement.

5.4 Error Analysis

Task	How to Build a Small Chick Brooder
Method	Setting up the Brooder
Plan-Retrieve	Set up a brooder. Use absorbent bedding. Provide a warm place. Place any heating apparatus needed for the species that is going to be kept. Monitor the temperature. Set up an attractive feeder. Choose an appropriate hiding spot. Ensure that there is proper ventilation in the stable.
DFS	Set up a brooder. Point a heat lamp on one side of your brooder to keep the chicks warm. Move your chicks to their brooder. Move the chicks to a brooder. Provide food and water for your chicks. Move the chicks to a brooder. Set up a brooder. Move the chicks to a brooder.
Golden plan	Fill the bottom of your brooder with paper towels and pine shavings. Point a heat lamp on one side of your brooder to keep the chicks warm. Provide food and water for your chicks. Put a branch in the brooder to create a roost.
Rewrite	Position the cage away from drafts and direct sunlight. Set up a brooder. Fill the bottom of your brooder with paper towels and pine shavings. Put a branch in the brooder to create a roost. Point a heat lamp on one side of your brooder to keep the chicks warm. Move your chicks to their brooder.

Table 4: Red text indicates steps that are irrelevant to the task and redundant steps, while green text denotes supplementary additional information.

We mainly perform analysis on GPT-3.5 on wikiHow to fairly compare various methods. We analyze from the perspectives of executability and generation quality. As executability shown in table 3, since the output format of the first four methods is usually a sentence or a list, inexecutable plans all come from the hallucination of LLM, which means LLM generate content beyond the given set. However, the output format of the Rewrite method is more complex. We observe that the non-executable plans contains 11.84% format parsing errors. The small format error ratio proves that LLMs have strong instruction following ability, but they still face serious hallucination problem.

To compare the generation quality of different methods, we present cases from the Plan-retrieve, DFS, Rewrite and golden plan from wikiHow in table 4. The plans generated by Plan-retrieve might generate steps that are correct in meaning but irrelevant in detail to the task. This is because the retrieved actions are not always relevant to the task, and LLM, given a one-time, limited selection, may be forced to choose these steps to make the plan complete. Besides being incomplete or irrelevant to the task, DFS also suffers from duplicate steps, despite being provided with previously selected steps. We find that 19.32% of plans generated by DFS contain repetitions of two or more times. Meanwhile, this step by step generation is also less complete, for example, missing information on the soft bedding material. In contrast, the Rewrite method, through iteration and global consideration, can generate more complete plans. Additionally, with a large pool of candidate steps, LLM can even find supplementary information for the task.

5.5 Retrieval Amount Influence

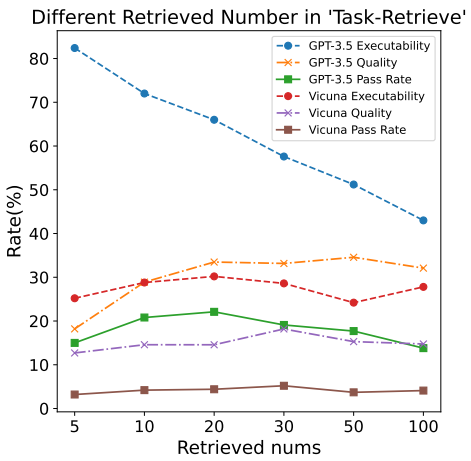


Figure 3: Influence of the number of retrieved actions on the performance of GPT-3.5 and Vicuna conducting Task-Retrieve method.

We find that the number of retrieved actions directly affects the execution rate and quality of the generated plan. We check the result on *wikiHow-Computers and Electronics* for a setup using GPT-3.5 and Vicuna for the task-retrieve method[4]. Figure 3 shows how "Executability", "Quality" and "Pass Rate" change when different numbers of steps are recalled for plan generation. The two solid lines representing "Pass Rate" demonstrate a trend of initial increase followed by a decrease. For GPT-3.5, as the number of recalled actions increases, the proportion of generated plans that

[4]Different from the main experiment, here we only check the output format.

conform to the rules decreases, while the quality of plans lifted. Pass Rate achieves the best performance when recalling 20 actions. Interestingly, we observed that when recalling 5 actions there are still parts of the plan that are not executable. We found that this is because the recalls are too little for LLMs to complete the task using the recall steps. But GPT-3.5 tends to generate complete plans, so steps for supplementing and connecting context are generated. We continue to discuss the results in other settings in Appendix D.

6 Conclusion

In this study, we introduced *Open Grounded Planning* and developed a benchmark comprising datasets from various domains with vast action sets. Extensive experiments revealed significant limitations in the performance of current models and methods in generating grounded plans for these sets. Furthermore, we observed a pronounced challenge in enabling these models and methods to generalize from in-domain scenarios to out-of-domain datasets. Compared to four other methods, our "Retrieve and Rewrite" approach demonstrates a partial resolution to the challenges inherent in open grounded planning. Our work highlights the need for enhancing the capability of models and methods for expansive planning domains and improving the executability and quality of grounded plans, laying a foundation for future research.

Limitation

Our current implementation relies on a two-stage approach of retrieval and generation. We anticipate that an optimized retriever tailored for this task will achieve better performance. Additionally, our current dataset only explores how to ground tasks into a given set of actions. While this is sufficient for many applications, a more challenging extension would be to introduce parameters for each action, meaning the use of a collection of action functions instead of a collection of actions.

Our current method for evaluating the content of the plan involves using ChatGPT as the evaluator, which is to guide ChatGPT to compare and judge the plans through prompting. This method inevitably brings bias and hallucination into the evaluation results. Although we have employed various methods to alleviate their impact and have sampled some cases for manual evaluation to prove the effectiveness of our evaluation method, bias and hallucination persist. In future work, we may introduce more diverse objective evaluation metrics and ways to reduce bias and hallucination.

Acknowledgements

We sincerely thank all anonymous reviewers for their insightful comments and valuable suggestions. This research work is supported by the National Natural Science Foundation of China under Grants no. 62106251, no. 62122077 and no. 62306303, and the Basic Research Program of ISCAS, Grant No. ISCAS-JCZD-202303.

References

Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Daniel Ho, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Eric Jang, Rosario Jauregui Ruano, Kyle Jeffrey, Sally Jesmonth, Nikhil J. Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Kuang-Huei Lee, Sergey Levine, Yao Lu, Linda Luu, Carolina Parada, Peter Pastor, Jornell Quiambao, Kanishka Rao, Jarek Rettinghouse, Diego Reyes, Pierre Sermanet, Nicolas Sievers, Clayton Tan, Alexander Toshev, Vincent Vanhoucke, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Mengyuan Yan, and Andy Zeng. 2022. Do As I Can, Not As I Say: Grounding Language in Robotic Affordances. *ArXiv preprint*, abs/2204.01691.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alexander Herzog, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Tomas Jackson, Sally Jesmonth, Nikhil J. Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Kuang-Huei Lee, Sergey Levine, Yao Lu, Utsav Malla, Deeksha Manjunath, Igor Mordatch, Ofir Nachum, Carolina Parada, Jodilyn Peralta, Emily Perez, Karl Pertsch, Jornell Quiambao, Kanishka Rao, Michael S. Ryoo, Grecia Salazar, Pannag R. Sanketi, Kevin Sayed, Jaspiar Singh, Sumedh Sontakke, Austin Stone, Clayton Tan, Huong T. Tran, Vincent Vanhoucke, Steve Vega, Quan Vuong, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. 2023. RT-1: Robotics Transformer for Real-World Control at Scale. In *Robotics: Science and Systems XIX, Daegu, Republic of Korea, July 10-14, 2023*.

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. 2020. 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*.

Shibo Hao, Yi Gu, Haodi Ma, Joshua Hong, Zhen Wang, Daisy Wang, and Zhiting Hu. 2023. Reasoning with Language Model is Planning with World Model. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 8154–8173, Singapore. Association for Computational Linguistics.

Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. 2022. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 9118–9147. PMLR.

Ryan Koo, Minhwa Lee, Vipul Raheja, Jong Inn Park, Zae Myung Kim, and Dongyeop Kang. 2023. Benchmarking Cognitive Biases in Large Language Models as Evaluators. *ArXiv preprint*, abs/2309.17012.

Minghao Li, Yingxiu Zhao, Bowen Yu, Feifan Song, Hangyu Li, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. 2023. API-Bank: A Comprehensive Benchmark for Tool-Augmented LLMs. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 3102–3116. Association for Computational Linguistics.

Bill Yuchen Lin, Chengsong Huang, Qian Liu, Wenda Gu, Sam Sommerer, and Xiang Ren. 2023. On Grounded Planning for Embodied Tasks with Language Models. In *Thirty-Seventh AAAI Conference on Artificial Intelligence, AAAI 2023, Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence, IAAI 2023, Thirteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2023, Washington, DC, USA, February 7-14, 2023*, pages 13192–13200. AAAI Press.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. 2023. ToolLLM: Facilitating Large Language Models to Master 16000+ Real-world APIs. In *The Twelfth International Conference on Learning Representations*.

Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. 2023. HuggingGPT: Solving AI Tasks with ChatGPT and its Friends in Hugging Face. *ArXiv preprint*, abs/2303.17580.

Chan Hee Song, Brian M. Sadler, Jiaman Wu, Wei-Lun Chao, Clayton Washington, and Yu Su. 2023. LLM-Planner: Few-Shot Grounded Planning for Embodied Agents with Large Language Models. In *IEEE/CVF International Conference on Computer Vision, ICCV 2023, Paris, France, October 1-6, 2023*, pages 2986–2997. IEEE.

Qiaoyu Tang, Ziliang Deng, Hongyu Lin, Xianpei Han, Qiao Liang, Boxi Cao, and Le Sun. 2023. ToolAlpaca: Generalized Tool Learning for Language Models with 3000 Simulated Cases. *ArXiv preprint*, abs/2306.05301.

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

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, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open Foundation and Fine-Tuned Chat Models. *ArXiv preprint*, abs/2307.09288.

Zihao Wang, Shaofei Cai, Guanzhou Chen, Anji Liu, Xiaojian Ma, and Yitao Liang. 2023a. Describe, Explain, Plan and Select: Interactive Planning with Large Language Models Enables Open-World Multi-Task Agents. *ArXiv preprint*, abs/2302.01560.

Zihao Wang, Shaofei Cai, Anji Liu, Yonggang Jin, Jinbing Hou, Bowei Zhang, Haowei Lin, Zhaofeng He, Zilong Zheng, Yaodong Yang, Xiaojian Ma, and Yitao Liang. 2023b. JARVIS-1: Open-World Multi-task Agents with Memory-Augmented Multimodal Language Models. *ArXiv preprint*, abs/2311.05997.

Minghao Wu and Alham Fikri Aji. 2023. Style Over Substance: Evaluation Biases for Large Language Models. *ArXiv preprint*, abs/2307.03025.

Zhenyu Wu, Ziwei Wang, Xiuwei Xu, Jiwen Lu, and Haibin Yan. 2023. Embodied Task Planning with Large Language Models. *ArXiv preprint*, abs/2307.01848.

Rui Yang, Lin Song, Yanwei Li, Sijie Zhao, Yixiao Ge, Xiu Li, and Ying Shan. 2023. GPT4Tools: Teaching Large Language Model to Use Tools via Self-instruction. *ArXiv preprint*, abs/2305.18752.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Takahide Yoshida, Atsushi Masumori, and Takashi Ikegami. 2023. From Text to Motion: Grounding GPT-4 in a Humanoid Robot "Alter3". *ArXiv preprint*, abs/2312.06571.

Quan Yuan, Mehran Kazemi, Xin Xu, Isaac Noble, Vaiva Imbrasaite, and Deepak Ramachandran. 2023. TaskLAMA: Probing the Complex Task Understanding of Language Models. *ArXiv preprint*, abs/2308.15299.

Li Zhang, Qing Lyu, and Chris Callison-Burch. 2020. Reasoning about goals, steps, and temporal ordering with WikiHow. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4630–4639, Online. Association for Computational Linguistics.

Zirui Zhao, Wee Sun Lee, and David Hsu. 2023. Large Language Models as Commonsense Knowledge for Large-Scale Task Planning. *ArXiv preprint*, abs/2305.14078.

Chujie Zheng, Hao Zhou, Fandong Meng, Jie Zhou, and Minlie Huang. 2023a. Large Language Models Are Not Robust Multiple Choice Selectors. *ArXiv preprint*, abs/2309.03882.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023b. Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena. *ArXiv preprint*, abs/2306.05685.

A Dataset Details

For the datasets of WikiHow, Tools, and Robot, although they have different original formats, they can all be transformed into a uniform format with three parts: task name, method, and steps. For WikiHow, each life guide (task) contains a title and several steps, with each step including a headline and a detailed explanation of the step. We retain only the headline part as the step to complete the task. Some guides may have different methods to achieve this. For example, one can create a poster by either hand drawing or using paper cutting. We randomly select one of these methods as the method to accomplish the task. The final example for WikiHow is shown in Table 1.

For Tools scenarios, we extract all the API calls under each task as steps. To maintain consistency with other tasks, we only retain the API name and description without including API parameters, as this would require additional training, and many works have already explored this kind of capability(Qin et al., 2023). We will also set the default method to some tasks to make the results they generate more consistent with requirements. Here is an example from GPT4Tools. Given a task of *Generate a real image of xxx from the sketch image* and a training instance for API call, we will transform it into a process of Sketch Detection On Image -> *Generate Image Condition On Sketch Image*.

```
{
    "title": "Generate a real image of a
         cat sitting on a table next to
        a bowl from the sketch image",
    "method": "One or two steps are
        usually enough to complete the
        task, and there are only a few
        cases where more may be required
        .",
    "steps": [
        "Sketch Detection On Image
            DESCRIPTION: useful when you
             want to generate a scribble
             of the image. like:
            generate a scribble of this
            image, or generate a sketch
            from this image, detect the
            sketch from this image. ",
        "Generate Image Condition On
            Sketch Image DESCRIPTION:
            useful when you want to
            generate a new real image
            from both the user
            description and a scribble
            image or a sketch image. "
    ]
}
```

In the Robot scenario, we can also obtain tasks, methods, and corresponding steps in the same way.

As described in the footnotes, the SayCan repository provides two files: one providing tasks and environment states, and the other providing tasks and corresponding plans, but they do not completely match. Since the tasks in this dataset have similar processes, such as instructing the robot to reach a certain place or pick up an item, we filled in the inconsistent parts based on this style. We also provide an example here.

```
{
    "title": "I'd like a clear soda.",
    "method": "As a robot with only one
        gripper, you are surrounded by a
         far counter, a near counter, a
        table, and a trash can. You are
        located near the table. You can
        only perform one action at a
        time, such as moving or picking
        up and putting down.
        Environmental status:water on
        table, 7up on table",
    "steps": [
        "find a 7up",
        "pick up the 7up",
        "bring it to you",
        "put down the 7up"
    ]
}
```

B Implementation Details

We use FastChat (Zheng et al., 2023b) for training, and the training parameters are consistent with vicuna-7B. For all generation steps, we perform generation with temperature = 1.0. We perform up to five retries per generation to avoid formatting errors. We also performed a rule review on the output of LLM to obtain the best performance of LLM in the open grounding planning task. For example, if LLM is required to choose one of several options as the next step, and it does not output a sentence that meets the requirements, we will regard this output as a failure and regenerate it. This retry will count towards the five retries above. For tools datasets, our output combines API name and API description, and the input format is "{api name} DESCRIPTION: {API description}". Since we only care about choosing the correct step, we accept both API name and "{API name} DESCRIPTION: {anything}" as input.

We simply use OpenAI's *text-embedding-ada-002* for embedding generation in all settings. We used different recall numbers for different methods. For the plan-retrieve method, each generation step recalls the two most relevant choices. For the task-retrieve method, we retrieve the 20 most relevant candidate steps from the task name. Stepwise Selecting and DFS methods are similar in that we both perform recalls of size 5. In the Rewrite method, we will select at most the first three steps that have not been replaced in each round, and dynamically control the recall number of each step to around 10. In all settings, we will first perform deduplication on the recall steps and then hand it over to LLM for other operations.

In addition, since the Stepwise Selecting, DFS, and Rewrite methods will iterate multiple times, we set an upper limit of 20, 30, and 20 iterations for them. These upper bounds are usually sufficient to complete the task, but if the LLM reaches the upper limit of the number of iterations, it means that the generated steps may be incomplete. If the plan complies with the rules, we still think the plan is executable, but incomplete plans will have an impact on the quality of the plan.

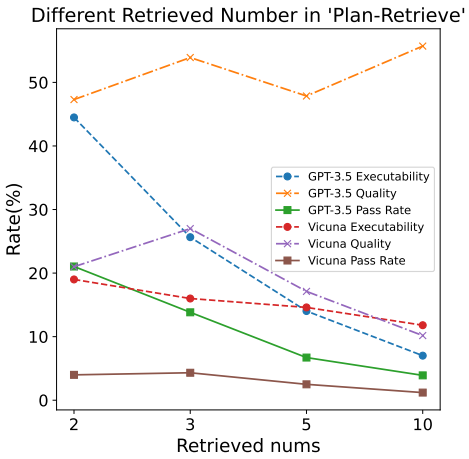
C Evaluation Details

In the evaluation set, we randomly selected a total of 200 cases and conducted human evaluations on the three models and five methods we used in our experiments. The Spearman rank correlation coefficient between the results of human and automated is 80.76%, which indicates that the automated evaluation results using ChatGPT present a significant consistency with those of human evaluation. Therefore, this automated assessment approach is deemed both reasonable and feasible.

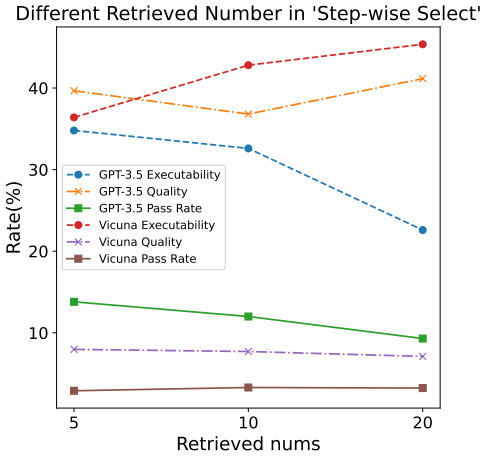
D Different Retrieved Numbers

The experimental results concerning the impact of the retrieved item number are illustrated in Figure 4 for both the Plan-Retrieve and Step-wise Select methods. For the plan-retrieve method, as the number of retrieved items increases, the available actions for the model to select also increase, leading decrease in plan executability, possibly due to excessive choices causing interference for the model, leading to the generation of illusory steps and consequently preventing the generated plan from being fully grounded in actions from the candidate sets. However, the generated quality shows a trend of decline after improvement.

For the select method, vicuna and GPT-3.5 show different properties. As the number of options available for vicuna increases, its executability rate will also increase. This has caused its Pass Rate to also show a slightly upward trend. The results of GPT-3.5 show a downward trend.



(a) plan-retrieve



(b) select

Figure 4: Different Retrieved numbers in plan-retrieve & select

Method	APIBank Executability(%)	Quality(%)	**Pass Rate(%)**
Vicuna-7B-v1.5-16k			
Task-Retrieve	95.06	38.40	36.50
Plan-Retrieve	62.74	41.52	26.05
Step-wise Select	79.47	26.08	20.72
DFS	100.0	26.81	26.81
Retrieve and Rewrite	91.63	49.59	45.44
GPT-3.5			
Task-Retrieve	66.16	56.32	37.26
Plan-Retrieve	49.43	61.92	30.61
Step-wise Select	74.52	43.37	32.32
DFS	100.00	35.55	35.55
Retrieve and Rewrite	92.40	47.33	43.73
LLaMA-2-7B(SFT)			
Task-Retrieve	96.58	60.83	**58.75**
Plan-Retrieve	80.61	59.20	47.72
Step-wise Select	87.83	41.14	36.12
DFS	43.02	26.49	11.40
Retrieve and Rewrite	74.43	61.03	45.42

Table 5: Detailed performance on APIBank

Method	GPT4Tools Executability(%)	Quality(%)	**Pass Rate(%)**
Vicuna-7B-v1.5-16k			
Task-Retrieve	83.20	19.59	16.30
Plan-Retrieve	54.00	27.96	15.10
Step-wise Select	73.00	21.64	15.80
DFS	99.00	20.71	20.50
Retrieve and Rewrite	59.80	37.29	22.30
GPT-3.5			
Task-Retrieve	81.40	28.50	23.20
Plan-Retrieve	67.40	47.92	32.30
Step-wise Select	76.00	30.92	23.50
DFS	100.00	25.00	25.00
Retrieve and Rewrite	90.60	29.36	26.60
LLaMA-2-7B(SFT)			
Task-Retrieve	50.80	52.17	26.50
Plan-Retrieve	61.80	56.63	35.00
Step-wise Select	17.80	23.60	4.20
DFS	4.14	11.60	0.48
Retrieve and Rewrite	89.40	48.88	**43.70**

Table 6: Detailed performance on GPT4Tools

E More Result of OOD Datasets

In this section, we provide more detailed performance results on the five OOD datasets. Tables 5, 6, and 7 show the results on the Tools dataset, while Tables 8 and 9 present the performance of LLMs on the Robot dataset. We can draw some similar conclusions to those found in the In-Domain dataset. For example, the Task-Retrieve method has a higher execution rate, but Plan-Retrieve achieves better plans. Additionally, the performance of methods such as Select and DFS is limited by the model's capabilities, as the model needs to follow the single-step selection instruction and choose the next step reasonably. Existing models often stop too early or too late and lack attention to previous

Method	ToolAlpaca		
	Executability(%)	Quality(%)	**Pass Rate(%)**
Vicuna-7B-v1.5-16k			
Task-Retrieve	50.25	39.60	19.90
Plan-Retrieve	41.79	23.21	9.70
Step-wise Select	53.73	18.98	10.20
DFS	98.01	16.50	16.17
Retrieve and Rewrite	62.19	38.00	23.63
GPT-3.5			
Task-Retrieve	45.77	32.61	14.93
Plan-Retrieve	35.32	33.80	11.94
Step-wise Select	76.12	21.90	16.67
DFS	100.00	19.90	19.90
Retrieve and Rewrite	87.06	33.14	28.86
LLaMA-2-7B(SFT)			
Task-Retrieve	99.50	49.25	**49.00**
Plan-Retrieve	79.10	45.60	36.07
Step-wise Select	67.66	32.35	21.89
DFS	12.62	22.10	2.79
Retrieve and Rewrite	90.55	49.73	45.02

Table 7: Detailed performance on ToolAlpaca

Method	VirtualHome		
	Executability(%)	Quality(%)	**Pass Rate(%)**
Vicuna-7B-v1.5-16k			
Task-Retrieve	74.80	16.58	12.40
Plan-Retrieve	45.80	25.98	11.90
Step-wise Select	65.20	10.89	7.10
DFS	98.60	6.90	6.80
Retrieve and Rewrite	75.20	21.28	16.00
GPT-3.5			
Task-Retrieve	97.40	27.21	26.50
Plan-Retrieve	67.00	56.12	37.60
Step-wise Select	79.00	38.73	30.60
DFS	100.00	32.90	32.90
Retrieve and Rewrite	97.00	48.45	**47.00**
LLaMA-2-7B(SFT)			
Task-Retrieve	99.60	37.65	37.50
Plan-Retrieve	91.80	46.51	42.70
Step-wise Select	100.00	34.10	34.10
DFS	92.40	38.57	35.64
Retrieve and Rewrite	96.20	48.65	46.80

Table 8: Detailed performance on VirtualHome

Method	SayCan		
	Executability(%)	Quality(%)	**Pass Rate(%)**
Vicuna-7B-v1.5-16k			
Task-Retrieve	70.73	21.12	14.94
Plan-Retrieve	56.71	21.51	12.20
Step-wise Select	44.51	4.11	1.83
DFS	95.12	2.56	2.44
Retrieve and Rewrite	80.49	15.53	12.50
GPT-3.5			
Task-Retrieve	91.46	32.00	29.27
Plan-Retrieve	91.46	49.00	**44.82**
Step-wise Select	87.80	30.56	26.83
DFS	100.00	11.28	11.28
Retrieve and Rewrite	99.39	41.41	41.16
LLaMA-2-7B(SFT)			
Task-Retrieve	98.78	37.96	37.50
Plan-Retrieve	73.17	42.08	30.79
Step-wise Select	100.00	31.71	31.71
DFS	91.50	16.35	14.96
Retrieve and Rewrite	92.68	46.05	42.68

Table 9: Detailed performance on SayCan

steps, leading to poorer performance.

F Prompts for All Methods

Our prompt contains two parts: instruction and input. In GPT-3.5, we use instruction for system messages, while in Vicuna and SFT models, since they are not trained on different system messages, we place it at the beginning of user input.

We use 0-shot in most settings and only use 2-shot in a few such as generating initial plans and rewriting. Prompts set to 0-shot often only have formatting requirements. The rest of the operations need to match the characteristics of the dataset or are more complex, so we think they require additional information.

```
<GENERATE FINAL PLAN PROMPT>
INSTRUCTION:
You will be given a task, a method to complete the task and several available actions. If no
method is specified it will be set to "None". Here are a few things you need to keep in mind:
1. You need to generate a plan that satisfies the given tasks and methods.
2. You can only use the steps in <Actions in Library> to complete a given task, even though
the provided steps may not complete the task.
3. You must use the actions in the library exactly. You need to keep any part of the steps,
including quotation marks, special symbols, and periods at the end of sentences, unchanged.
Send your answer in the following format and do nothing else: 1. step1
2. step2
3. step3...

PROMPT:
<Task>: How to Tell the Difference Between a Tortoise, Terrapin and Turtle
<Method>: Observing the Reptile's Behavior
<Actions in library>:
1. Assess what kind of turtle you are dealing with before you start.
2. {other steps}…

RESPONSE:
-------------------------------------------------------------------------------
1. Assess what kind of turtle you are dealing with before you start.
2. {other steps}...
```

Figure 5: task-retrieve prompt.

```
<GENERATE INITIAL PLAN PROMPT>
INSTRUCTION:
You will be given a task and a method to complete the task. If no method is specified it will
be set to "None". You need to generate a plan that satisfies the given tasks and methods. The
plan needs to be a list of several actions and each action should be a complete and short
sentence separated by newlines. Send your answer in the following format and do nothing else:
1. step1
2. step2
3. step3...

PROMPT_EXAMPLE_0:
<Task>: How to Watch Disney Plus on iPhone
<Method>: None
RESPONSE_EXAMPLE_0:
1. Open the Disney+ app.
2. {other steps}…

PROMPT_EXAMPLE_1:
<Task>: How to Improve Your Posture
<Method>: Using Exercise to Improve Your Posture
RESPONSE_EXAMPLE_1:
1. Improve your core muscles with deep abdominal stretching.
2. {other steps}…

PROMPT:
<Task>: How to Love Your Rabbit
<Method>: Handling and Caring for Your Rabbit

RESPONSE:
-------------------------------------------------------------------------------
1.Spend time bonding with your rabbit every day.
2.{other steps}…
```

Figure 6: plan-retrieve prompt to generate initial plan.

```
<GENERATE FINAL PLAN PROMPT>
INSTRUCTION:
You will be given a task, a method to complete the task, an initial plan to follow and a
number of available actions. If no method is specified it will be set to "None". Here are a
few things you need to keep in mind:
1. You need to generate a plan that satisfies the given tasks and methods.
2. The initial plan is a reference only, which means you should not output the steps in the
initial plan directly.
3. You can only use the steps in Actions in Library to complete a given task, even though the
provided steps may not complete the task.
4. You must use the actions in the library exactly. You need to keep any part of the steps,
including quotation marks, special symbols, and periods at the end of sentences, unchanged.
Send your answer in the following format and do nothing else: 1. step1
2. step2
3. step3...

PROMPT:
<Task>: How to Love Your Rabbit
<Method>: Handling and Caring for Your Rabbit
<Initial steps>:
1. Spend time bonding with your rabbit every day.
2. {other steps}…

<Actions in library>:
1. Spend time interacting with your rabbit every day.
2. {other steps}…

RESPONSE:
--------------------------------------------------------------------------------
1. Spend time interacting with your rabbit every day.
2. {other steps}…
```

Figure 7: plan-retrieve prompt to generate final plan.

```
<GENERATE POSSIBLE NEXT STEP PROMPT>
INSTRUCTION:
You will be given a task, a method to complete the task and a current plan. If no method is
specified it will be set to "None". If the current plan is empty, the plan will also be set
to "None". Remember:
1. You need to generate the next step in the plan that needs to meet the given tasks and
methods based on the existing plan. Please note that the step you generate will be added to
the end of the existing steps, and you need to pay attention to maintain the coherence of the
overall steps.
2. There is no going back in your generating process, you cannot try to delete or modify a
previously existing step.
3. If you think the current plan is sufficient for the task, just output "[New step: None]".
You only need to output one step and do nothing else.
Send your answer in the following format: [New step: step] or [New step: None]

PROMPT:
<Task>: How to Love Your Rabbit
<Method>: Handling and Caring for Your Rabbit
<Current plan>:
None

RESPONSE:
--------------------------------------------------------------------------------
[New step: Research the specific needs and behaviors of rabbits to understand how to properly
care for and handle them.]
```

Figure 8: step-wise select prompt to generate possible next step.

```
<SELECT NEXT STEP PROMPT>
INSTRUCTION:
You will be given a task, a method to complete the task, a current plan and several candidate
actions. Candidate actions are called <Actions in Library>. If no method is specified it will
be set to "None". If the current plan is empty, the plan will also be set to "None". Here are
a few things you need to keep in mind:
1. You need to select the next step in the plan from the candidate actions that satisfies the
given task and method based on the currently existing plan. Please note that the step you
select will be added to the end of the existing steps, and you need to pay attention to
maintain the coherence of the overall steps.
2. There is no going back in your generating process.
3. You can only use the steps in <Actions in Library> to complete a given task. If you think
that the provided steps may not accomplish the task, you need to select "[New step: None of
these]". Never use your own steps to complete a task, all output needs to be selected from
the options provided.
4. You must use the actions in the library exactly. You can't just output the sequence number
of an action, you must output the entire sentence. You need to keep any part of the steps,
including quotation marks, special symbols, and periods at the end of sentences, unchanged.
Send your answer in the following format and do nothing else: [New step: None of these] or
[New step: step]

PROMPT:
<Task>: How to Love Your Rabbit
<Method>: Handling and Caring for Your Rabbit
<Current plan>:
None
<Actions in library>:
Research rabbit care.
{other steps}…

RESPONSE:
--------------------------------------------------------------------------------
[New step: Research rabbit care.]
```

Figure 9: step-wise select prompt to select next step.

```
<GENERATE INITIAL STEPS PROMPT>
INSTRUCTION:
You will be given a task and a method to complete the task. If no method is specified it will
be set to "None". Remember:
1. Based on your plan for a given task and method, you need to generate three candidates for
the first step of completing the task with the specified method. These three steps must be
different. I would choose one of the three candidates you generated as an initial step in
planning. You need to pay attention to scalability in this step.
2. There is no going back in your generating process, you cannot try to delete or modify a
previously existing step.
Send your answer in the following format: [1. candidate1
2. candidate2
3. candidate3]

PROMPT:
<Task>: How to Tell the Difference Between a Tortoise, Terrapin and Turtle
<Method>: Observing the Reptile’s Behavior
<Current plan>:
None

RESPONSE:
--------------------------------------------------------------------------------
1. Observe the reptile's feeding behavior, as tortoises are primarily herbivores, while
terrapins and turtles eat a more varied diet.
2. Look for the reptile's swimming habits, as terrapins are generally comfortable in both
freshwater and brackish water, while turtles prefer strictly freshwater habitats.
3. Observe the reptile's sunbathing behavior, as tortoises often bask in the sun for extended
periods, while turtles and terrapins may do so but not as frequently.
```

Figure 10: DFS prompt to generate initial steps.

```
                                    <SELECT NEXT STEP PROMPT>
INSTRUCTION:
You will be given a task, a method to complete the task, a current plan and several candidate
actions. Candidate actions are called <Actions in Library>. If no method is specified it will
be set to "None". If the current plan is empty, the plan will also be set to "None". Here are
a few things you need to keep in mind:
1. You need to select the next step in the plan from the candidate actions that satisfies the
given task and method based on the currently existing plan. Please note that the step you
select will be added to the end of the existing steps, and you need to pay attention to
maintain the coherence of the overall steps.
2. There is no going back in your generating process.
3. You can only use the steps in <Actions in Library> to complete a given task. If you think
that the provided steps may not accomplish the task, you need to select "[New step: None of
these]". Never use your own steps to complete a task, all output needs to be selected from
the options provided.
4. You must use the actions in the library exactly. You can't just output the sequence number
of an action, you must output the entire sentence. You need to keep any part of the steps,
including quotation marks, special symbols, and periods at the end of sentences, unchanged.
Send your answer in the following format and do nothing else: [New step: None of these] or
[New step: step]

PROMPT:
<Task>: How to Tell the Difference Between a Tortoise, Terrapin and Turtle
<Method>: Observing the Reptile's Behavior
<Current plan>:
None
<Actions in library>:
Bathe your tortoise often.
other steps…

RESPONSE:
--------------------------------------------------------------------------------
[New step: Observe what the reptile eats.]

OR:
--------------------------------------------------------------------------------
[New step: None of these]
```

Figure 11: DFS prompt to select next step. The answer can be either new step or None.

```
<EXPAND NEXT STEPS PROMPT>
INSTRUCTION:
You will be given a task, a method to complete the task and a current plan. If no method is
specified it will be set to "None". If the current plan is empty, the plan will also be set
to "None". Remember:
1. You need to generate three candidates for the next step of the current plan based on the
existing plan to satisfy the plan for the given task and method. These three steps must be
different. Note that I will add one of these three candidates you generated to the end of the
existing step. You need to pay attention to maintaining the coherence of the overall steps.
2. There is no going back in your generating process, you cannot try to delete or modify a
previously existing step.
3. If you think the current plan is sufficient for the task, just output "[None]". You only
need to output steps and do nothing else.
Send your answer in the following format: [1. candidate1
2. candidate2
3. candidate3] or [None]

PROMPT:
<Task>: How to Tell the Difference Between a Tortoise, Terrapin and Turtle
<Method>: Observing the Reptile's Behavior
<Current plan>:
1.Observe what the reptile eats.

RESPONSE:
--------------------------------------------------------------------------------
1. Watch the reptile's movement patterns.
2. Observe the reptile's habitat.
3. Note the reptile's swimming behavior.
```

Figure 12: DFS prompt to expand nodes.

```
<GENERATE INITIAL PLAN PROMPT>
INSTRUCTION:
You will be given a task, a method to complete the task and several actions for reference.
These actions are called <References>. If no method is specified, it will be set to "None".
Remember:
1. You need to refer to the content in <References> to generate a plan that can complete the
task in the specified method.
2. The generated plan does not need to use the exact steps in <Reference>. You can generate
any plan as long as it can complete the task in the specified method. In subsequent
operations, I will use other actions in the library to rewrite it, so the plan you generate
needs to be as consistent in style as possible with these actions.
Send your answer in the following format and do nothing else: 1. step1
2. step2
3. step3...

PROMPT:
<Task>: How to Tell the Difference Between a Tortoise, Terrapin and Turtle
<Method>: Observing the Reptile's Behavior
<References>:
1. Assess what kind of turtle you are dealing with before you start.
2. {other steps}…

RESPONSE:
--------------------------------------------------------------------------------
1. Research the characteristics of tortoises, terrapins, and turtles to understand the
differences in their behavior and appearance.
2. {other steps}…
```

Figure 13: rewrite prompt to generate initial plan.

```
                                        <REWRITE PROMPT>
INSTRUCTION:
You will be given a task, a method to complete the task, a current plan and several candidate
actions. Candidate actions are called <Actions in Library>. If no method is specified it will
be set to "None". If the current plan is empty, the plan will also be set to "None".
Use the actions listed below to refine your current steps to complete your task. Actions
marked with <TO BE REPLACED> indicate that the content was not found in the action library,
and actions marked with <IN LIB> indicate that they are in the action library. You need to
analyze which actions in the provided action library can be added to the action list and
replace some or all of the actions marked with <TO BE REPLACED>. We encourage you to add more
<TO BE REPLACED> content to complete these steps.
You can do the following:
* Replace any number of <TO BE REPLACED>-like operations with any number of <IN LIB>
operations.
* Replace any number of <IN LIB> operations with any number of <IN LIB> operations as the
latter are better suited to the task.
* Replace any number of <IN LIB> operations with more general <IN LIB> operations.
* Insert any number of <IN LIB> operations that differ from existing steps.
* Insert any number of <TO BE REPLACED> operations to fill in missing content between steps.
* Remove any number of redundant <IN LIB> operations.
* Remove any number of redundant <TO BE REPLACED> operations.
* Remove any number of overly verbose <IN LIB> operations.
* Compare several similar <IN LIB> operations and select the best one to add to the
operations list.
* Other reasonable actions.
Remember:
1. Your output needs to be a list, and each element in the list needs to start with "{<IN
LIB>" or "{<TO BE REPLACED>" and end with \"}\".
2. The rewritten sentences need to cover roughly the same scope as before the rewrite,
although they may differ in detail.
3. Only actions that are newly added from <Actions in library> can be marked with <IN LIB>,
otherwise they need to maintain their attributes in <Current steps>.
4. You must use the actions in the library exactly. You can't just output the sequence number
of an action, you must output the entire sentence. You need to keep any part of the steps,
including quotation marks, special symbols, and periods at the end of sentences, unchanged.
5. Always output the complete plan, not only newly added or changed steps. If the content
containing <IN LIB> is still needed, you need to output it completely.
Send your answer in the following format and do nothing else: [1. {<IN LIB> step1}
2. {<TO BE REPLACED> step2}
...]

PROMPT_EXAMPLE_0:
<Task>: How to Increase Your Income
<Method>: Cutting Down on Expenses
<Current steps>:
1. {<TO BE REPLACED> Avoid eating out.}
2. {<TO BE REPLACED> Cancel unused subscriptions and memberships.}
3. {<TO BE REPLACED> Bike or walk to work, rather than drive.}
4. {<TO BE REPLACED> Find free or low-cost entertainment options instead of expensive
outings.}
5. {<TO BE REPLACED> Reduce your rent.}

<Actions in library>:
{<IN LIB> Cancel or suspend memberships or subscriptions that you're no longer using, or that
you're using ineffectively.}
{<IN LIB> Select Cancel Subscription.}
{<IN LIB> Cancel your dating profiles and subscriptions.}
{<IN LIB> Click Cancel Subscription.}
{<IN LIB> Click Cancel subscription.}
{<IN LIB> Find fun things to do together as a family that don't cost a lot.}
{<IN LIB> Find alternative or more cost effective ways to spending time on your own.}
{<IN LIB> Pursue less costly hobbies.}
{<IN LIB> Choose affordable activities.}
{<IN LIB> Take advantage of free fun.}
```

Figure 14: rewrite prompt to rewrite.

```
                                    <REWRITE PROMPT>-continue
RESPONSE_EXAMPLE_0:
1. {<TO BE REPLACED>Avoid eating out.}
2. {<IN LIB> Cancel or suspend memberships or subscriptions that you're no longer using, or
that you're using ineffectively.}
3. {<TO BE REPLACED> Bike or walk to work, rather than drive.}
4. {<IN LIB> Choose affordable activities.}
5. {<TO BE REPLACED> Reduce your rent.}

PROMPT_EXAMPLE_1:
<TASK>: How to Make Fried Chicken with Buttermilk and Tarragon
<Method>: None
<Current steps>:
1. {<TO BE REPLACED> Marinate chicken pieces in buttermilk and tarragon for at least 4 hours
or overnight in the refrigerator.}
2. {<TO BE REPLACED> Remove chicken from the buttermilk marinade and let excess drip off.}
3. {<TO BE REPLACED> Dredge the chicken in seasoned flour, ensuring it is evenly coated.}
4. {<TO BE REPLACED> Heat oil in a pan to 350°F (175°C) and carefully place the chicken in
the hot oil.}
5. {<TO BE REPLACED> Fry until golden brown and fully cooked, usually about 15-20 minutes
depending on the size of the chicken pieces.}
6. {<TO BE REPLACED> Once cooked, place the fried chicken on a wire rack to drain excess oil.}
7. {<TO BE REPLACED> Serve and enjoy!}

<Actions in library>:
{<IN LIB> Refrigerate marinated chicken for 4 to 6 hours.}
{<IN LIB> Allow the chicken to marinate overnight.}
{<IN LIB> Shake off the marinade from the chicken pieces.}
{<IN LIB> Remove chicken pieces from the buttermilk mixture and dredge in flour.}
{<IN LIB> Add chicken pieces to seasoned flour and toss to coat.}
{<IN LIB> Heat oil in skillet to 350 °F (177 °C).}
{<IN LIB> Preheat the oil to 350° Fahrenheit or 176° Celsius.}
{<IN LIB> Fry the chicken in a frying pan until crispy and golden in color.}
{<IN LIB> Fry about 2 to 3 minutes or until they are golden brown and crispy.}

RESPONSE_EXAMPLE_1:
1. {<TO BE REPLACED> Mix buttermilk, tarragon, salt and pepper in a large bowl.}
2. {<TO BE REPLACED> Add chicken pieces to mixture to marinate.}
3. {<IN LIB> Refrigerate marinated chicken for 4 to 6 hours.}
4. {<IN LIB> Remove chicken pieces from the buttermilk mixture and dredge in flour.}
5. {<IN LIB> Heat oil in skillet to 350 °F (177 °C).}
6. {<IN LIB> Fry the chicken in a frying pan until crispy and golden in color.}
7. {<TO BE REPLACED> Once cooked, place the fried chicken on a wire rack to drain excess oil.}
8. {<TO BE REPLACED> Serve and enjoy!}

PROMPT:
<Task>: How to Tell the Difference Between a Tortoise, Terrapin and Turtle
<Method>: Observing the Reptile's Behavior
<Current plan>:
1. {<TO BE REPLACED> Research the characteristics of tortoises, terrapins, and turtles to
understand the differences in their behavior and appearance.}
2. other steps…

<Actions in library>:
{<IN LIB> Assess what kind of turtle you are dealing with before you start.}
{<IN LIB> other steps…}

RESPONSE:
--------------------------------------------------------------------------------
1. {<IN LIB> Assess what kind of turtle you are dealing with before you start.}
2. {<TO BE REPLACED> Watch the reptile to observe its movement patterns and habits, paying
attention to its interaction with water and land.}
3. other steps…
```

Figure 15: rewrite prompt to rewrite-continue.

```
<EVALUATION PROMPT>
PROMPT:
For a given task, a possible method to solve the task and two corresponding plans, please
evaluate the plans based on three aspects: completeness, feasibility, and relevance to the
task, with a maximum score of 10 points for each aspect. The detailed scoring criteria for
each aspect are as follows:
- Completeness: Examining whether the plan is comprehensive, with a focus on the coherence
between steps, the presence of necessary steps, and the avoidance of arbitrarily introduced
conditions.
- Feasibility: Assessing the practicality of the plan, considering whether each step can be
implemented, whether the plan aligns with common sense, adheres to human ethical standards,
and avoids excessive redundant steps.
- Relevance to the task: Evaluating the extent to which the plan is related to the given task,
considering the use of provided task conditions and whether it achieves the desired goals of
the task.

Task: {TASK}
Method: {METHOD}

Plan1:
{PLAN1}

Plan2:
{PLAN2}

Now, read the task, method and plans provided, and compare the plans. In the 'Analysis'
section, provide a brief rationale for your comparison in the three aspects. Then, based on
your analyses, choose the better plan (could be chosen from [Plan1, Plan2]):

Output:

<Analysis>
- Completeness:
- Feasibility:
- Relevance:
</Analysis>

<Better Plan> [Plan1, Plan2] </Better Plan>
```

Figure 16: evaluation prompt.