

DO THINK TAGS REALLY HELP LLMs PLAN? A CRITICAL EVALUATION OF REACT-STYLE PROMPTING

Anonymous authors

Paper under double-blind review

ABSTRACT

The reasoning abilities of Large Language Models (LLMs) remain a topic of debate, which are critically tested in sequential decision-making problems. ReAct, a recently popular method has gained popularity for claiming to enhance LLM reasoning abilities while directly prompting them by “*interleaving reasoning trace with action execution*” in text-based planning domains such as AlfWorld and WebShop. However, given the different components of ReAct-style prompting, it remains unclear what the source of improvement in LLM performance is. In this paper, we critically examine the claims of ReAct-style prompting for sequential decision-making problems. By introducing systematic variations to the input prompt, we perform a sensitivity analysis along the original claims of ReAct. Contrary to these claims and common use-cases that utilize ReAct-style prompting, we find that the performance is minimally influenced by the interleaved reasoning trace or by the content of these generated reasoning traces. Instead, the performance of LLMs is primarily driven by the unreasonably high degree of similarity between input example tasks and queries, implicitly forcing the prompt designer to provide instance-specific examples which significantly increases the cognitive burden on the human. Our empirical results, on the same suite of domains as ReAct, show that the perceived reasoning abilities of LLMs stem from the exemplar-query similarity and approximate retrieval rather than any inherent reasoning abilities.

1 INTRODUCTION

Large Language Models (LLMs) have seen rapid advancements specifically in Natural Language Processing and Understanding (NLP & NLU). LLMs have unparalleled capabilities in text generation, summarization, translation, question answering to name a few. (Bubeck et al., 2023). Motivated by these capabilities of LLMs, there has also been a rush to look for other emergent abilities—especially for reasoning and planning. A popular way of enhancing LLM performance on reasoning/planning tasks has been in-context prompting or prompt-engineering (Sahoo et al., 2024) to include instructions (Giray, 2023), syntax structure (Marvin et al., 2023), criticism and plan guidance with verification (Kambhampati et al., 2024) etc. Among these approaches, ReAct (Yao et al., 2022b), presented at ICLR 2023, stands out which claims to improve LLM planning abilities through the use of reasoning traces interleaved with action execution given as plan guidance. While their original study includes multiple domains tasks such as question answering on HotPotQA and FEVER datasets (Yang et al., 2018; Thorne et al., 2018), in this paper we are particularly interested in planning tasks where ReAct claims superior performance.

In our initial experiments with ReAct for planning, we found that the system is overly dependent on a high degree of syntactic similarity between the example prompt and the query, and is extremely brittle to minor perturbations to the input prompt. For example, when provided with an explicit set of examples of *pick-and-place-object* task and asked to plan for a *pick-and-place-two-objects* task, it should be trivial to generalize the solution of the examples to the queried task. Unfortunately, even such a minor variation to the original ReAct agent setup disrupts its performance.

Given the seemingly widespread adoption of ReAct methodology (as of this writing, it has 1,408 citations), the brittleness we witnessed calls for a systematic study of the factors contributing to the performance of ReAct-based LLM Agents. Moreover, recent studies have highlighted similar case studies on the original Chain-of-Thought idea (Stechly et al., 2024a; Sprague et al., 2024). Based on

054 the claims of (Yao et al., 2022b), we isolate three possible reasons for the claimed performance of
 055 ReAct framework: 1) the utility of interleaving reasoning trace during action execution, 2) the utility
 056 of providing plan guidance, and, 3) the significance of example prompt provided to the the LLM.

057 The brittleness of ReAct becomes apparent when considering the variability in prompt designs.
 058 Depending on the domain and task, prompt designers may provide abstract guidance, task-specific
 059 instructions within the same domain, analogical examples, or global reasoning traces. ReAct’s
 060 inability to robustly adapt to such variations underscores its limitations in handling diverse input
 061 prompts, thereby necessitating a closer examination of its design and implementation. In this work,
 062 we systematically evaluate the brittleness of ReAct by studying which potential factors contribute
 063 to its performance. This analysis is conducted through variations in input prompts to understand
 064 how a ReAct LLM Agent responds to (1) where the guidance is provided, (2) the different types
 065 and structure of this guidance, and finally, (3) on varying the resemblance of example prompt to the
 066 queried problem. We investigate the research questions :

067 **RQ1:** Does the agent performance depend on interleaving reasoning trace with action execution?

068 **RQ2:** How does the nature of the reasoning trace or guidance information affect the performance of
 069 LLM Agents?

070 **RQ3:** How does the similarity between the example ⟨problem, solution⟩ and the query ⟨problem, ? ⟩,
 071 which are present in the prompt, affect LLM Agent performance?

072 We conduct extensive experiments on the AlfWorld and WebShop domain using various LLM Models,
 073 including GPT-3.5-turbo, GPT-3.5-instruct, GPT-4-0314(Old Variant), GPT-4-0613 (Latest Variant),
 074 GPT-4o, Claude-Opus and Llama 3.1-8b. Through our comprehensive empirical study, we answer
 075 each of the research questions above. (RQ1) We find that LLM performance in-fact improves when
 076 the reasoning trace is **not** interleaved with action execution. (RQ2) Moreover, providing weaker
 077 guidance or placebo-guidance (where the text provides no information about the task) has comparable
 078 performance to strong reasoning trace-based guidance. Answers to RQ1 and RQ2 suggest that the
 079 source of ReAct LLM agent performance is **not** the interleaving aspect or the content of the reasoning
 080 trace. Finally, in (RQ3), we see that variations to the example prompt such that it belongs to different
 081 task within the same domain, or has a different goal or plan attributes than the queried problem;
 082 causes the performance of ReAct-LLM Agent to plummet.

083 Our findings highlight that the benefits of ReAct-style prompting are present when prompt engineers
 084 can curate highly instance-specific examples. This may not scale for domains with a large number of
 085 problem instance classes, and it places an undue burden on prompt engineers to provide instance specific
 086 examples. Finally, our experiments call into question claims of enhanced “emergent reasoning”
 087 of LLMs with prompt engineering efforts; corroborating contemporary research (Verma et al., 2024;
 088 Valmeekam et al., 2024; Stechly et al., 2024b; Ullman, 2023; Schaeffer et al., 2023; McCoy et al.,
 089 2023; Stechly et al., 2024a; Sprague et al., 2024) questioning reasoning abilities of LLMs.

090 For the rest of the paper, we begin with discuss the relevant literature discussing LLM prompting for
 091 planning problems in Section 2. Next, we provide the preliminaries in Section 3, and discuss our
 092 investigations of ReAct-style prompting in detail in Section 4. We present our empirical results with
 093 a detailed discussion drawing insights from our results in Section 5. Finally, we conclude the work
 094 in Section 6. An appendix with additional experiment details, results, and prompts, has also been
 095 attached, and the code will be released on acceptance.

096 2 RELATED WORK

099 Large Language Models have been shown to be successful in a plethora of natural language tasks
 100 (Kocoń et al., 2023; Gilardi et al., 2023; Zhu et al., 2023; Bubeck et al., 2023; Bhattacharjee et al.,
 101 2024). However, there are two schools of thought when it comes to utilizing off-the-shelf LLMs for
 102 planning and reasoning tasks. Works such as Chain of Thought, ReAct, and others that followed (Wei
 103 et al., 2022; Yao et al., 2023; Long, 2023; Yao et al., 2024; Besta et al., 2024; Fu et al., 2024; Aksitov
 104 et al., 2023), have argued about the reasoning abilities of LLMs by proposing prompting methods. On
 105 the other hand, (Valmeekam et al., 2024; Stechly et al., 2024b) have refuted these claims by showing
 106 the inability of LLMs to solve deterministic planning and classical reasoning problems.

107 In particular, for investigating the use of LLMs in solving decision making problems, AlfWorld
 (Shridhar et al., 2020) is a popular domain that was originally proposed for training text-based

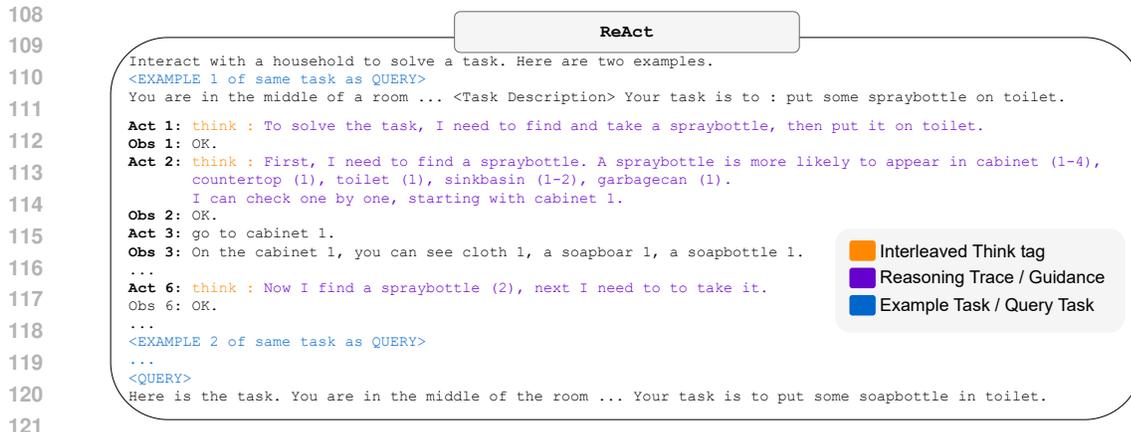


Figure 1: An example of ReAct in AlfWorld. We highlight the main components of ReAct, i.e., Interleaved reasoning and acting, the reasoning trace / plan guidance and the example and query task.

Reinforcement Learning agents. Lately, works such as ReAct, Reflexion, and their other variants (Yao et al., 2022b; Shinn et al., 2023) have argued on the prowess of LLMs’ reasoning abilities on AlfWorld. Furthermore, there have been several extensions to ReAct that boost their generalization abilities across more domains including multi-modal domains (Yang et al., 2023; Castrejon et al., 2024), autonomous vehicles (Cui et al., 2024), table question answering (Zhang et al., 2023), etc. While the effectiveness of ReAct is celebrated across different areas, these works only depend on anthropomorphization of LLMs for using ReAct based prompting with no justification on the source of improvement in performance. This motivates our work in investigating the components of ReAct with respect to sequential decision-making problems and analyzing the role each component plays.

3 PRELIMINARIES

3.1 DOMAINS

AlfWorld: (Shridhar et al., 2020) is a synthetic text-based game built on top of a STRIPS-style PDDL domain description (Fikes & Nilsson, 1971). ReAct (Yao et al., 2022b) defines six tasks (or problem classes) within this domain namely - Put, Clean, Heat, Cool, Examine, and PutTwo. Each problem class consists of several problem instances, such as *put a spraybottle on toilet* (see Fig. 1 is an example instance of Put class. Since AlfWorld is a partially observable environment, each of these problem instances can be solved by navigating and interacting with the environment simulator via text actions. For example, this task can be solved by the following actions- go to cabinet 2, take spraybottle 2 from cabinet 2, go to toilet 1, put spraybottle 2 in/on toilet 1.

WebShop: (Yao et al., 2022a) is an online shopping website environment with 1.18M real-world products and 12K human instructions. The agent is provided with an initial human instruction (for example, “I am looking for a nightstand with drawers. It should have a nickel finish, and priced lower than \$140”). The agent’s task is to crawl the shopping environment using actions such as `search ‘nightstand drawers’`, `choose ‘white buttons’`, `back to search`, etc. For this work, we randomly sample 500 test instructions from the environment and evaluate the success rate of the agent’s task completion.

3.2 REACT

ReAct (Yao et al., 2022b) claims to increase LLM’s performance on text-based planning tasks such as AlfWorld and WebShop primarily by augmenting the original action space of the agent with a *think* action. The *think* action tag provided by ReAct is claimed to comprise of **Reasoning + Action** trace that is provided in the solution for the example problems (exemplars) as part of the prompt. During execution, the LLM is expected to generate a *think* action tag for the queried problem instance that is semantically similar to the one provided for the examples in the prompt.

Location of *THINK* tag In ReAct, the integration of the *think* tag within actions serves to expand the action space. This allows the language model (LLM) agent to execute a *think* action, prompting an ‘OK’ response. Through analysis of example prompts in ReAct experiments, we identify various instances of the `think` action. Typically, it appears after stating the problem instance, reiterating the task, and providing problem-specific guidance. However, the authors offer no structured guidelines for its implementation, placement, or guidance. This observation aligns with feedback from the paper’s reviewers (OpenReview, 2024) citing inconsistencies in the prompting format.

Content of *THINK* tag In ReAct, the *think* action consistently provides the decision-making agent with success-oriented guidance for task completion. For instance, upon encountering a `spraybottle`, the prompt might include: `think: Now I find a spraybottle (2). Next, I need to take it.` This guidance exposes forthcoming actions and sub-tasks for the agent.

Few shot *EXAMPLES* In the `AlfWorld` domain (which is a PDDL domain), ReAct authors (Yao et al., 2022b) classify six problem classes or tasks: `Put`, `Clean`, `Heat`, `Cool`, `Examine`, `PutTwo`. Despite representing different tasks, they share the same environment dynamics and action space, allowing for very similar execution trace. For instance, a `Heat` task might involve `Putting` an item into a microwave. In ReAct experiments, authors provide two example problem-solution pairs (referred to as exemplars in our work) before querying the LLM agent with a problem instance. Authors force ReAct agent to use examples and queries belonging to the same problem class without motivating this design decision. However, the queried problem may differ in objects or locations from the exemplars.

4 A CRITICAL EVALUATION OF REACT-STYLE PROMPTING

We examine the claims of ReAct to understand the performance of ReAct-based LLM agents. It is crucial to assess whether ReAct’s fundamental claims hold, particularly in planning. As outlined in Section 3, ReAct comprises three main components: interleaving the `think` tag with actions, plan guidance after the `think` tag, and the selection of exemplar problems for LLM prompts. We perform a sensitivity analysis by proposing alternatives along these three dimensions. The subsequent sections explore the design of exemplar prompt variations to investigate our research questions concerning the claims of ReAct. Each variation modifies the base ReAct prompt, and we use the `AlfWorld` domain as the running example for discussing these variations in the following sub-sections. We curate variations for `WebShop` domain in a similar manner. Readers can find the variations in the attached supplementary materials.

4.1 RQ1 : INTERLEAVING *thinking* WITH ACTING

Does the agent performance depend on interleaving reasoning trace with action execution?

To answer this research question, we propose collating the guidance information contained within the multiple *think* tags present in the examples of the input prompt into a single *think* tag appended after the example problem is specified. This approach can be interpreted as Chain-of-Thought (Kojima et al., 2022; Wei et al., 2022), where guidance information is generated before action execution.

Variation 1: Exemplar-based CoT `AlfWorld` is a partially observable environment where an agent can only observe objects after reaching that location. Hence, we remove specific location and object identifiers to modify the *think* actions that are originally interleaved with other actions in the environment. For example, we rewrite it as: `think: Now Once I find a spraybottle (2), next I need to take it..` Finally, we append all the *think* actions together at the beginning of the example problem. **Intuition:** Problem-specific guidance for a sequential decision-making agent can be given step-by-step (as in ReAct) or all at once.

Variation 2: Anonymized Exemplar-CoT We take one step further and modify the *think* tag to remove references to specific locations and objects, making it more general. For example, `think: To solve the task, I need to find and take a spraybottle the object, then put it on toilet the desired location.` Similarly, in `WebShop` we anonymize specific item options as desired options. **Intuition:** Exemplars

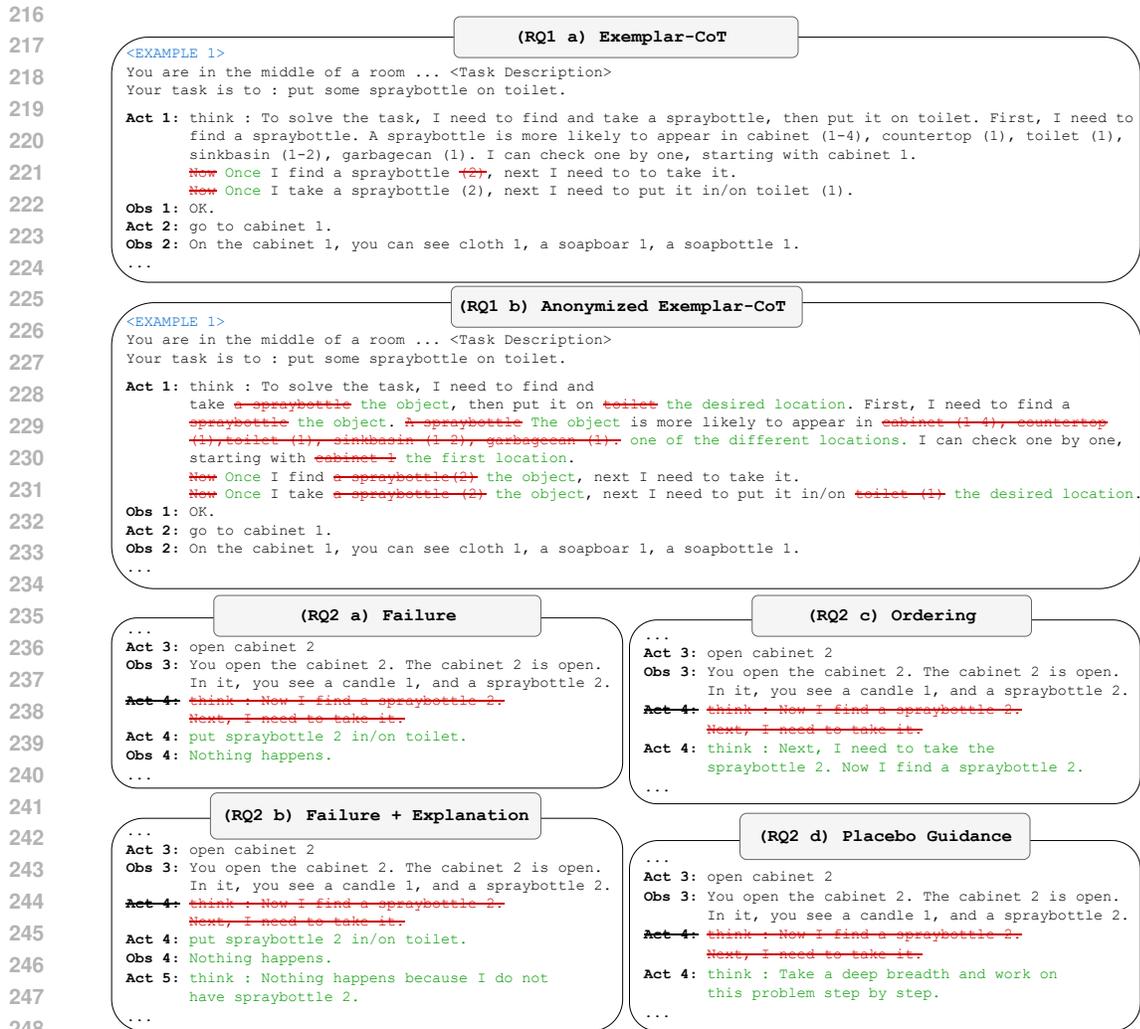


Figure 2: Example of prompt variations considered for RQ1 and RQ2.

can be made more general by providing abstract guidance and exploiting LLMs ability to identify necessary semantic entity relations.

4.2 RQ2 : PLAN GUIDANCE FOLLOWING *think* TAG

How does the nature of the reasoning trace or guidance information affect the performance of LLM?

ReAct claims to use reasoning trace as the guidance information following the *think* tag. For instance, in ReAct () thoughts are to (1) decompose the goal (2) track subgoal completion (3) determine the next subgoal and (4) reason via common-sense where to find and object and what to do with it. It is, however, unclear what is the motivation to use these as the reasoning trace. The potential anthropomorphization of large language models (LLMs) may suggest that their thought processes are similar to the abstract plans humans make, and that they must be prompted in the same manner. However, it is unclear why this assumption should hold true. Alternatives can be, we can prompt the LLM to reflect on past failures and provide possible explanations (hindsight-guidance) or We can substitute task-relevant guidance with placebo-guidance by using "magic incantations".

Variation 1: Failure From the example prompts used in ReAct, we note that none of the examples for any task consist of invalid actions. We inject two invalid actions in the execution trace : the first that attempts to execute the action pertinent to the task (such as `put spraybottle 2 in/on`

toilet) when not possible and, second, executes some other invalid action. We include the expected simulator response, `Nothing happens.`, when invalid actions are taken. **Intuition:** Reasoning trace can be about *what to do* such as subgoals of the future, or *what not to do* such as mistakes in hindsight. This should be weaker guidance than in base ReAct as the exemplars do not point out what to do next.

Variation 2: Failure + Explanation We place *think* actions after invalid actions injected in Failure Variation which consist of explanations for the failure such as `think: Nothing happens because I do not have a spraybottle 2.` **Intuition:** We can augment pointing out mistakes in hindsight with explanations to avoid similar failures. This is stronger guidance signal than Failure, however, the exemplars still not provide information on what to do next.

Variation 3: Guidance Ordering LLMs are known to be susceptible to minor syntactic perturbations to inputs. We test whether it is true for guidance information given as prompt as well. We identify chain of subtasks in a reasoning trace $S_1 \rightarrow S_2 \cdots S_n$ and reverse it to be $S_n \rightarrow S_{n-1} \cdots S_1$. For instance, `think: Now I find a spraybottle (2). Next, I need to take it.` becomes `think: Next, I need to take the spraybottle (2). Now I find a spraybottle (2).` **Intuition:** LLM agent should be invariant to the syntax of reasoning trace if the semantic information is preserved. This does not change the reasoning trace from the perspective of information content.

Variation 4: Placebo Guidance It is unclear to what extent LLM agent uses the supposed helpful thoughts for the decision making task. In this variation we replace *think* tag guidance with a placebo thought that does not contain any task relevant information, but has been widely used as prompt engineering trick (Kojima et al., 2022). **Intuition:** According to claims of ReAct, we expect the performance to get worse when the guidance does not have any information useful for task success.

4.3 RQ3 : SIMILARITY BETWEEN *EXAMPLES* AND *QUERY*

How does the similarity between the example (problem, solution) and the query (problem, ?), which are present in the prompt, affect LLM Agent performance?

RQ3 investigates the role of example similarity to the query in LLM agent’s performance. Establishing problem similarity can be challenging, especially where minor variations to the problem can have varied interpretations (such as an analogy to a different task altogether). Our work explores this challenge in a systematic way. During example prompt construction, prompt designers may use synonyms to refer to objects (*Domain*), come up with examples where the agent task is the same as query but the goals are different (*Instance*), or provide optimal solutions as the examples (*Optimal*) preventing LLM to obtain information regarding exploration strategy. Furthermore, given that the domain has the same underlying action dynamics and that the tasks reuse several actions, prompt designers may choose to provide query specific example prompts (as in base ReAct), provide one of a different task and one of the same task (*One*), provide both examples to be of a different task (*Both*), or take an exhaustive approach and provide one example of all tasks (*All*).

Variation 1: Synonyms - (Domain) For this variation, we replace the object and location names in the example prompts with their synonyms. For example, `spraybottle` \rightarrow `aerosolbottle`, `cabinet` \rightarrow `cupboard`, and, `microwave` \rightarrow `oven`. We make 36 such changes to object and location names across all the examples. Note that the object names / location are unchanged for the problem query and subsequent interaction with the simulator. **Intuition:** Exemplar guidance maybe specified with alternate synonymous object and location names. Reasoning agents should be invariant to variable name substitution for closed world dynamics such as PDDL based *AlfWorld*.

Variation 2: Problem Instance-level - Instance We inject instance-level changes to the examples provided in the prompts. Recall that we are updating the base ReAct’s prompts, where the exemplar tasks are same as the query. We change the goal location in exemplar problem to ensure that it does not match with any of the goal locations in query problem. Moreover, we add repetitive yet futile actions in the exemplar execution trace which does not effect the solution. **Intuition:** Ensuring a different goal location in exemplar from the queried problem is a natural usecase. Moreover, exemplars may contain arbitrary exploration strategies such as action repetition (Sharma et al., 2017). By ReAct’s claims, LLM agent performance should not be affected.

Table 1: Average Success % of LLM for RQ1 and RQ2 on six AlfWorld tasks.

Model / Prompt	Act	ReAct	RQ1		RQ2			
			CoT	Anon. CoT	Placebo	Order	Failure	Explanation
GPT-3.5-Turbo	34.3	27.6	46.6	41	30	28.3	43.3	41.6
GPT-3.5-Instruct	44	50.7	61.9	50.7	41	42.5	47	44.7
GPT-4-0314 (Old)	-	23.3	43.3	33.3	36.6	30	50	36.6
GPT-4-0613 (Latest)	70.0	26.7	40.0	26.6	36.6	30	60	36.6
Claude-Opus	43.3	56.6	50	46.6	30	50	53.3	30

Table 2: Average Success % of LLM for RQ1 and RQ2 on WebShop tasks.

Model / Prompt	Act	ReAct	RQ1		RQ2		
			CoT	Anon. CoT	Placebo	Failure	Explanation
GPT-3.5-Turbo	1.12	1.04	2.20	1.88	1.52	3.48	3.48
GPT-3.5-Instruct	7.24	7.16	7.52	6.12	7.40	7.20	7.24
GPT-4-0613 (Latest)	8	4	8	8	6	8	8
GPT-4o	4.64	2.24	4.68	4.52	4.08	4.68	4.68
Claude-Opus	4	4	4	2	4	2	4
LLAMA-3.1-8B	1.44	3.16	3.28	3.92	2.04	1.20	2.16

Variation 3: Problem Level - Both, One, All Recall that the environment dynamics for all the tasks are the same. In fact, several tasks subsume the use of our tasks such as `Heat` requires the agent to `Put` an food in the microwave. In general, all the tasks share a large portion of actions (such as exploring cabinets and locations, picking objects etc.). Motivated by how tight relationship of these tasks we come up with three variations. First, `One`, uses one exemplar of an arbitrarily picked task and the other exemplar of the same task as the query. Second, `Both`, uses both exemplars from an arbitrarily picked task. Finally, `All`, uses a total of six exemplars (this is the only variation where we provide more than the standard two examples as in `ReAct`) corresponding to each task under consideration. Remember, this includes the query task which is always present at the end in the input prompt. **Intuition:** With a very similar action execution trace (such as exploration, picking and placing objects) across tasks, and shared dynamics, LLM agent should be minimally affected by the use of exemplars of a different task.

Variation 4: Exploration Strategy - Optimal As noted before, `ReAct` does not explain the choice of exemplars used. An important ingredient to the exemplars is the exploration strategy used. In this variation we provide exemplars which serendipitously take the optimal actions (as if the environment were fully observable) and therefore the example plan is the shortest possible. **Intuition:** Exploration strategy exposed in exemplars (that too for the same problem task) should not impact `ReAct`'s performance if the LLM agent is reasoning instead of retrieval (or pattern matching).

5 RESULTS

In the following sub-sections, we will answer our three RQs using the proposed prompt variations along three dimensions, i.e., the location of the `think` tag, the content of the `think` tag, and the similarity between exemplars and queried problems. All the variations modify the base `ReAct` prompts and we do not present a cross between variations unless otherwise noted. While the original `ReAct` experiments were carried out on PaLM (currently decommissioned), we reproduce their results with newer set of models. We use GPT-3.5-Turbo, GPT-3.5-Instruct, GPT-4, GPT-4o, and Claude-Opus, which are all newer models than those benchmarked in `ReAct` (Yao et al., 2022b). Note, that despite using newer models, our results shed doubts on the reproducibility and consistency across models of the original paper's results. As noted, we use the setup in (Yao et al., 2022b) for all our experiments. In AlfWorld, GPT3.5(Turbo, Instruct) results are on 134 instances across six tasks, GPT-4/Claude-Opus on 60 instances (10 for each task) due to cost considerations. In WebShop, GPT3.5(Turbo, Instruct), GPT-4o, LLAMA-3.1-8B results are on 500 samples, GPT-4/Claude-Opus are on 50 instances due to cost considerations.

Table 3: Average Success % of LLM for RQ3 on six AlfWorld tasks. OC: Out of context limit

Model / Prompt	Act	ReAct	RQ3					
			Domain	Instance	Optimal	All	One	Both
GPT-3.5-Turbo	34.3	27.6	1.6	30	20.1	32	28.3	1.6
GPT-3.5-Instruct	44	50.7	47.6	42.5	39.5	OC	17.9	5.2
GPT-4-0314 (Old)	–	23.3	13.3	23.3	50	23.3	16.6	0
GPT-4-0613 (Latest)	70.0	26.7	10.0	20.0	53.3	23.3	20	3.3
Claude-Opus	43.3	56.6	50	46.6	43.3	50	60	6.6

5.1 UTILITY OF INTERLEAVING REASONING TRACE WITH ACTION EXECUTION

Given the claims of ReAct, a practitioner’s expectation is that all the rows would be red as the performance would drop by changing the location of the think tag. From Table 1(RQ1) note that the exemplar CoT and the anonymized exemplar CoT performs significantly better than base ReAct for all GPT-X family of models. Moreover, the performance dips slightly for Claude-Opus along these variations. From Fig. 3 (larger area represents worse performance), we observe that base ReAct consistently performs worse in most of the tasks. This refutes ReAct’s first claim on the importance of interleaving reasoning trace generation with action execution. Even in the case of the Claude where there is a slight dip in performance, the models seems to be performing at reasonably high success rate which questions the importance of interleaved reasoning and action execution. We omit LLAMA-3.1-8B and GPT-4o (See B.4) for AlfWorld as they achieve zero performance over baselines and all the variations. From Table 2, we find a similar pattern: CoT and Anon. CoT variants perform closely or better than the baseline ReAct. A surprising result consistent in both the domains is the performance of Act baseline (where think tags are absent and actions are generated directly). Act baseline is weaker only for two models GPT-3.5-Instruct, Claude-Opus for both the domains, which further questions the utility of using ReAct style paradigm in the first place.

5.2 UTILITY OF GUIDANCE INFORMATION FOLLOWING think TAG

Recall that reasoning trace guidance pertains to the prospective actions or behaviors an agent should execute (foresight guidance). This type of guidance is more informative compared to other variations, such as hindsight guidance, which focuses on past errors without providing future solution steps, and placebo guidance, which is entirely unrelated to the task. ReAct claims that reasoning trace is crucial for LLM agent performance, which would predict a decline in performance with hindsight guidance and a collapse with placebo guidance. Therefore, a practitioner would expect all the rows to be a dark shade of red. In contrast, our findings in Table 1 indicate that hindsight guidance (Failure, Explanation) actually improves the performance of the GPT family of models. The Claude-Opus model’s performance remains stable with hindsight (Failure) guidance and declines with placebo guidance. Figure 4 illustrates these models’ performance across six AlfWorld tasks and variations, showing that the performance of LLMs either improved or remained consistent when provided with weaker or irrelevant guidance information. This refutes ReAct’s claim that task-specific reasoning trace is the source of LLM agent performance. Our argument that LLM agent’s performance is only slightly affected by the reasoning trace explains the indifference to ordering perturbation as well. If the LLM is not utilizing the reasoning trace for decision making, change in ordering would not affect the agent’s performance. Our arguments hold for the Webshop domain as well, where all of the variants perform closely or better than the baseline ReAct. Finally, contrary to the general perception that better GPT models would improve over reasoning, we find that GPT-4-(Old)’s performance is the worst among GPT-X family further highlighting the brittleness of claims of ReAct. GPT-4-(Latest) performs similarly to GPT-4-(Old), except for the Act baseline which again shows the futility of ReAct prompting.

5.3 UTILITY OF EXEMPLAR SIMILARITY TO QUERY TASK

Intuitively, the similarity of Domain examples is closest with base ReAct, followed by Instance and Optimal variations. Finally, All contains an overload of information followed by One and Both which has the same action space but uses different tasks as exemplars. Recall that AlfWorld being a PDDL domain has a shared environment dynamics across all tasks with upto 80% of actions

shared across execution traces. While ReAct does not investigate impact of varied exemplars, given the popular usage one expects LLMs to be robust to such changes especially in a common-sense household domain. Table 3 shows the severe brittleness of ReAct based LLM agent to even minor variations (such as `Domain`, `Instance`). Specifically, performance of GPT-3.5-Turbo and GPT-4 plummets for `Domain`. Claude-Opus which was more robust in RQ1, RQ2, is also impacted severely by `Domain`, `Instance` variations. Furthermore, when we do not expose the exploration strategy and only provide Optimal exemplars, the performance of LLM agents further drops (except in GPT4).

Overloading the LLMs with more exemplars `All` does not impact its performance. We posit, this is because the query-task exemplar is still part of the large input prompt. Among the two exemplars, as provided in ReAct, when one of them is of a different task (`One`) then the performance significantly reduces for LLMs. When both of the exemplars are of a different task then the performance collapses to single digit success rates for all the models. This is a key result of this work highlighting the severe dependence of LLMs on the similarity of the exemplars to the query task. Through sensitivity analysis using our RQ3 variations we could find parts of the input (the task similarity of the exemplar with query) which is the source of ReAct performance. Essentially, the LLM is mimicking / performing approximate retrieval from the context presented to it. Moreover, our results corroborate the line of research that questions the inability of LLMs to reason or plan (Verma et al., 2024; Valmeekam et al., 2024; Stechly et al., 2024b; Ullman, 2023; Schaeffer et al., 2023; McCoy et al., 2023; Stechly et al., 2024a; Sprague et al., 2024).

The reported success-rate from the ReAct paper Yao et al. (2022b) on the WebShop domain is 40%. Due to the absence of the exact queries used in the paper, we randomly sampled queries from the WebShop dataset comprising 12K records. This approach possibly resulted in the decoupling of any relationship between the exemplars and the queries. Referring to Table 2, it is evident that the performance of the WebShop ReAct agent significantly declined, reaching single digit percentages (as well as other variants). This mirrors the trends observed in the `Both` variant of the Alfworld in Table 3, further supporting our arguments.

Unrolling and Subtask Similarity We perform additional experiments where the query task is to essentially repeat the task in the exemplar (`Unrolling`). For instance, in Alfworld, the exemplar is `Put` and the query is `PutTwo` to put two objects at given location. In this case, the LLM has to unroll the given advice and repeat exemplar task execution to solve the query. The success rate of GPT-3.5-Instruct (the best performing GPT model in our experiments) drops down from 52% to 9%. Similarly, we experiment with a `Subtask Similarity` variation where the exemplar task subsumes execution of the query task. For instance, the `Heat` task requires the agent to pick and place object in the microwave (which is an instantiation of `Put` task). One would expect that `Heat` is a good exemplar for `Put`, however, the performance of GPT-3.5-instruct model goes from 18% to 0% in this case. These results further underscore the brittleness and the need for instance-specific exemplars in ReAct.

Thought operationalization ability of LLMs Given the free form nature of thought generation and arbitrary nature of thought (about subtask, common-sense next steps etc.), checking whether the generated thoughts are in-fact reasonable is a challenging problem. For completeness, we find that 40% of the times after generation of a `think` tag, subsequent environment action taken by the LLM was invalid (for GPT-3.5-instruct) in Alfworld. It is much higher (80% for GPT-3.5-Turbo, 90% for Claude-Haiku) for weaker LLM models. This further highlights the inability of LLMs to operationalize its generated thought as also seen in (Roy et al., 2024). From manual inspection we find that the typical thoughts would enlist all possible locations as next locations to visit for most of the tasks. As demonstrated in Section 5.2, the performance of LLMs actually decreases when provided with foresight guidance, as seen with the base ReAct model. A detailed investigation into the validity of the generated reasoning traces is beyond the scope of this work and is suggested as future research.

5.4 DISCUSSION

In this sub-section, we aim to draw insights from our experiments across the three RQs which can be further extended to understanding the limitations of LLMs for planning problems. Specifically, we discuss a) the pitfalls of using ReAct-style prompting for planning domains which could further be ex-

acerbated by approaches that build on top of ReAct framework, and b) scalability and generalizability issues as observed across multiple LLMs.

Pitfalls of ReAct-Style Prompting: Recall, that ReAct claims an improved performance for text-based planning domains, namely - AlfWorld and WebShop, where the presence of a *think* tag acts as guidance for the LLM to generate the next set of actions during the LLM-environment interaction. Through our sensitivity analysis, we dissect each component of ReAct-style prompting in a critical effort to understand the factor that leads to the observed success rates in these domains. With variations on the placement (RQ1) and content (RQ2) of the *think* tag, we eliminate it as the primary cause of any improvement. Furthermore, slight variations in exemplar tasks (RQ3) lead to a stark decline in success rate, clearly indicating the dependence of performance on the highly curated instance-specific examples by domain experts. While newer research in the art of prompting has pointed out the impact of well-curated examples, our work specifically highlights exemplar-query similarity as the cause of ReAct’s performance and rejects contemporary belief that the heavy-lifting of LLM reasoning & planning is done through the *think* tag.

Relevance of ReAct to Newer LLMs: ReAct uses the `Act` baseline in their to showcase improvements due to the presence of the proposed *think* tag. For AlfWorld, ReAct reports 45% success rate for `Act` baseline and 71% success rate for ReAct prompting using the PaLM model. For WebShop, ReAct reports 30.1% success rate for `Act` baseline and 40% success rate for ReAct prompting on PaLM. However, we note from our results on both domains that the `Act` baseline performs much better than ReAct for several LLMs, which questions on the compatibility of ReAct to newer-age LLMs. ReAct performs worse with newer models as compared to the results they report on PaLM, which is currently decommissioned. This observation also questions the contemporary belief that such prompting strategies are generalizable throughout different LLM families, including newer models.

We re-iterate our key result, given any LLM model, the success rates plummet with our RQ3 variations showing a consistent pattern of dependence on the provided examples irrespective of the LLM. Moreover, the performance of all the LLMs remain quite high (if not better) when we vary the location and content of the *think* tags. This highlights the need for higher rigor in agentic LLM experimentation and in-depth evaluation seeking source of improvements. Finally, we highlight our previous discussion on *unrolling*, *subtask-similarity* (discussing the brittleness of perceived reasoning abilities of LLMs) and the inability of LLMs to perform reliable *thought operationalization* as key limitations which exist despite ReAct style prompting.

6 CONCLUSION

ReAct based prompt engineering methods have been claimed to improve planning abilities of Large Language Models. In this study, we critically examine ReAct along three dimensions, informed by its claims and our hypotheses regarding its performance sources. Contrary to ReAct’s claims, our findings reveal that its performance is **neither** due to interleaving reasoning trace and guidance information generation with action execution, **nor** due to the specific nature of the guidance information. Instead, we identify that the true source of LLM performance in sequential decision-making tasks, such as AlfWorld, is the high degree of similarity between exemplar problems (few-shot) and the query task. We also showed that ReAct is susceptible to trivial variations in exemplar prompts (such as with the use of synonyms, or *Unrolling* and *Subtask Similarity* cases). Our findings caution against an uncritical adoption of ReAct-style frameworks for their putative abilities to enhance performance in domains requiring planning.

REFERENCES

- Renat Aksitov, Sobhan Miryoosefi, Zonglin Li, Daliang Li, Sheila Babayan, Kavya Koppurapu, Zachary Fisher, Ruiqi Guo, Sushant Prakash, Pranesh Srinivasan, et al. Rest meets react: Self-improvement for multi-step reasoning llm agent. [arXiv preprint arXiv:2312.10003](https://arxiv.org/abs/2312.10003), 2023.
- Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, et al. Graph of thoughts:

- 540 Solving elaborate problems with large language models. In Proceedings of the AAAI Conference
541 on Artificial Intelligence, volume 38, pp. 17682–17690, 2024.
- 542
- 543 Amrita Bhattacharjee, Raha Moraffah, Joshua Garland, and Huan Liu. Towards llm-guided causal
544 explainability for black-box text classifiers. 2024.
- 545 Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrike, Eric Horvitz, Ece Kamar,
546 Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. Sparks of artificial general intelligence:
547 Early experiments with gpt-4. arXiv preprint arXiv:2303.12712, 2023.
- 548
- 549 Lluís Castrejon, Thomas Mensink, Howard Zhou, Vittorio Ferrari, Andre Araujo, and Jasper Uijlings.
550 Hammr: Hierarchical multimodal react agents for generic vqa. arXiv preprint arXiv:2404.05465,
551 2024.
- 552 Can Cui, Yunsheng Ma, Xu Cao, Wenqian Ye, and Ziran Wang. Receive, reason, and react: Drive
553 as you say, with large language models in autonomous vehicles. IEEE Intelligent Transportation
554 Systems Magazine, 2024.
- 555
- 556 Richard E Fikes and Nils J Nilsson. Strips: A new approach to the application of theorem proving to
557 problem solving. Artificial intelligence, 2(3-4):189–208, 1971.
- 558 Dayuan Fu, Jianzhao Huang, Siyuan Lu, Guanting Dong, Yejie Wang, Keqing He, and Weiran Xu. Pre-
559 act: Predicting future in react enhances agent’s planning ability. arXiv preprint arXiv:2402.11534,
560 2024.
- 561
- 562 Fabrizio Gilardi, Meysam Alizadeh, and Maël Kubli. Chatgpt outperforms crowd workers for
563 text-annotation tasks. Proceedings of the National Academy of Sciences, 120(30):e2305016120,
564 2023.
- 565 Louie Giray. Prompt engineering with chatgpt: a guide for academic writers. Annals of biomedical
566 engineering, 51(12):2629–2633, 2023.
- 567
- 568 Subbarao Kambhampati, Karthik Valmeekam, Lin Guan, Kaya Stechly, Mudit Verma, Siddhant
569 Bhambri, Lucas Saldyt, and Anil Murthy. Llms can’t plan, but can help planning in llm-modulo
570 frameworks. arXiv preprint arXiv:2402.01817, 2024.
- 571 Jan Kocoń, Igor Cichecki, Oliwier Kaszyca, Mateusz Kochanek, Dominika Szydło, Joanna Baran,
572 Julita Bielaniewicz, Marcin Gruza, Arkadiusz Janz, Kamil Kanclerz, et al. Chatgpt: Jack of all
573 trades, master of none. Information Fusion, 99:101861, 2023.
- 574
- 575 Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large
576 language models are zero-shot reasoners. Advances in neural information processing systems, 35:
577 22199–22213, 2022.
- 578 Jieyi Long. Large language model guided tree-of-thought. arXiv preprint arXiv:2305.08291, 2023.
- 579
- 580 Ggaliwango Marvin, Nakayiza Hellen, Daudi Jjingo, and Joyce Nakatumba-Nabende. Prompt
581 engineering in large language models. In International Conference on Data Intelligence and
582 Cognitive Informatics, pp. 387–402. Springer, 2023.
- 583
- 584 R Thomas McCoy, Shunyu Yao, Dan Friedman, Matthew Hardy, and Thomas L Griffiths. Embers
585 of autoregression: Understanding large language models through the problem they are trained to
586 solve. arXiv preprint arXiv:2309.13638, 2023.
- 587
- 588 OpenReview. ReAct: Synergizing Reasoning and Acting in Language Models. [https://
openreview.net/forum?id=WE_vluYUL-X](https://openreview.net/forum?id=WE_vluYUL-X), 2024.
- 589
- 590 Shamik Roy, Sailik Sengupta, Daniele Bonadiman, Saab Mansour, and Arshit Gupta. Flap: Flow
591 adhering planning with constrained decoding in llms. arXiv preprint arXiv:2403.05766, 2024.
- 592
- 593 Pranab Sahoo, Ayush Kumar Singh, Sriparna Saha, Vinija Jain, Samrat Mondal, and Aman Chadha.
A systematic survey of prompt engineering in large language models: Techniques and applications.
arXiv preprint arXiv:2402.07927, 2024.

- 594 Rylan Schaeffer, Kateryna Pistunova, Samar Khanna, Sarthak Consul, and Sanmi Koyejo. Invalid
595 logic, equivalent gains: The bizarreness of reasoning in language model prompting. [arXiv preprint](#)
596 [arXiv:2307.10573](#), 2023.
- 597 Sahil Sharma, Aravind Srinivas, and Balaraman Ravindran. Learning to repeat: Fine grained action
598 repetition for deep reinforcement learning. [arXiv preprint arXiv:1702.06054](#), 2017.
- 600 Noah Shinn, Beck Labash, and Ashwin Gopinath. Reflexion: an autonomous agent with dynamic
601 memory and self-reflection. [arXiv preprint arXiv:2303.11366](#), 2023.
- 602 Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew
603 Hausknecht. Alfworld: Aligning text and embodied environments for interactive learning. [arXiv](#)
604 [preprint arXiv:2010.03768](#), 2020.
- 606 Zayne Sprague, Fangcong Yin, Juan Diego Rodriguez, Dongwei Jiang, Many Wadhwa, Prasann
607 Singhal, Xinyu Zhao, Xi Ye, Kyle Mahowald, and Greg Durrett. To cot or not to cot? chain-
608 of-thought helps mainly on math and symbolic reasoning. [arXiv preprint arXiv:2409.12183](#),
609 2024.
- 610 Kaya Stechly, Karthik Valmeekam, and Subbarao Kambhampati. Chain of thoughtlessness: An
611 analysis of cot in planning. [arXiv preprint arXiv:2405.04776](#), 2024a.
- 612 Kaya Stechly, Karthik Valmeekam, and Subbarao Kambhampati. On the self-verification limitations
613 of large language models on reasoning and planning tasks. [arXiv preprint arXiv:2402.08115](#),
614 2024b.
- 616 James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. Fever: a large-scale
617 dataset for fact extraction and verification. [arXiv preprint arXiv:1803.05355](#), 2018.
- 618 Tomer Ullman. Large language models fail on trivial alterations to theory-of-mind tasks. [arXiv](#)
619 [preprint arXiv:2302.08399](#), 2023.
- 621 Karthik Valmeekam, Matthew Marquez, Sarath Sreedharan, and Subbarao Kambhampati. On the
622 planning abilities of large language models-a critical investigation. [Advances in Neural Information](#)
623 [Processing Systems](#), 36, 2024.
- 624 Mudit Verma, Siddhant Bhambri, and Subbarao Kambhampati. Theory of mind abilities of large
625 language models in human-robot interaction: An illusion? In [Companion of the 2024 ACM/IEEE](#)
626 [International Conference on Human-Robot Interaction](#), pp. 36–45, 2024.
- 628 Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny
629 Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. [Advances in](#)
630 [neural information processing systems](#), 35:24824–24837, 2022.
- 631 Zhengyuan Yang, Linjie Li, Jianfeng Wang, Kevin Lin, Ehsan Azarnasab, Faisal Ahmed, Zicheng Liu,
632 Ce Liu, Michael Zeng, and Lijuan Wang. Mm-react: Prompting chatgpt for multimodal reasoning
633 and action. [arXiv preprint arXiv:2303.11381](#), 2023.
- 634 Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W Cohen, Ruslan Salakhutdinov,
635 and Christopher D Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question
636 answering. [arXiv preprint arXiv:1809.09600](#), 2018.
- 637 Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. Webshop: Towards scalable real-
638 world web interaction with grounded language agents. [Advances in Neural Information Processing](#)
639 [Systems](#), 35:20744–20757, 2022a.
- 640 Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan
641 Cao. React: Synergizing reasoning and acting in language models. In [The Eleventh International](#)
642 [Conference on Learning Representations](#), 2022b.
- 643 Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan.
644 Tree of thoughts: Deliberate problem solving with large language models. [Advances in Neural](#)
645 [Information Processing Systems](#), 36, 2024.

648 Yao Yao, Zuchao Li, and Hai Zhao. Beyond chain-of-thought, effective graph-of-thought reasoning
649 in large language models. [arXiv preprint arXiv:2305.16582](#), 2023.

650

651 Yunjia Zhang, Jordan Henkel, Avriella Floratou, Joyce Cahoon, Shaleen Deep, and Jignesh M Patel.
652 Reactable: Enhancing react for table question answering. [arXiv preprint arXiv:2310.00815](#), 2023.

653

654 Yiming Zhu, Peixian Zhang, Ehsan-Ul Haq, Pan Hui, and Gareth Tyson. Can chatgpt reproduce
655 human-generated labels? a study of social computing tasks. [arXiv preprint arXiv:2304.10145](#),
656 2023.

657

658

659

660

661

662

663

664

665

666

667

668

669

670

671

672

673

674

675

676

677

678

679

680

681

682

683

684

685

686

687

688

689

690

691

692

693

694

695

696

697

698

699

700

701

A RESOURCES USED

In this work we leverage OpenAI API and Claude API for prompting the Language Models. We use gpt-4-0613 for GPT4, gpt-3.5-turbo-0125, gpt-3.5-turbo-instruct, claude-3-opus-20240229, claude-3-sonnet-20240229 and claude-3-haiku-20240307 for AlfWorld in April-May 2024. Results using GPT-4-0613 (for both AlfWorld and WebShop) and all the results on Webshop were completed in Sep 2024. As an estimate, for AlfWorld, ReAct and corresponding experiments use approximately 14M input tokens (due to repeated prompting after each action execution) and 150K output tokens for 134 problem instances as used by ReAct.

B ADDITIONAL CONSIDERATIONS

B.1 FINE-GRAINED PERFORMANCE ON EACH TASK ON ALFWORLD

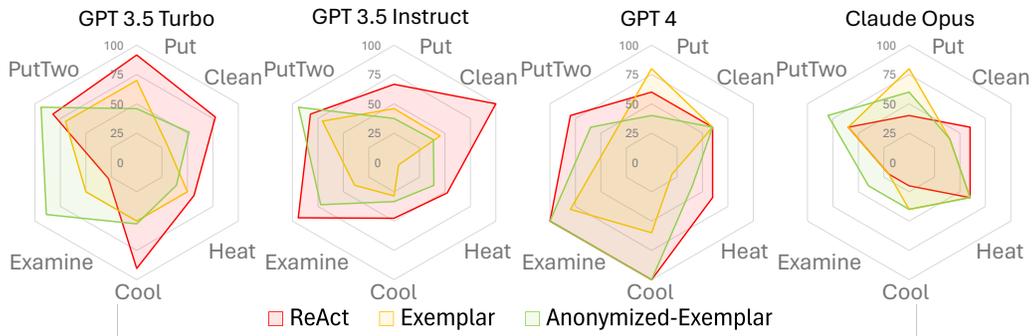


Figure 3: The radar chart shows the **failure rates** of various LLMs with different ReAct-based prompt settings for RQ1 (Base React, Global, Anonymized) across six Alfworld tasks (hexagon vertices). Higher values / Larger shaded region indicate worse performance.

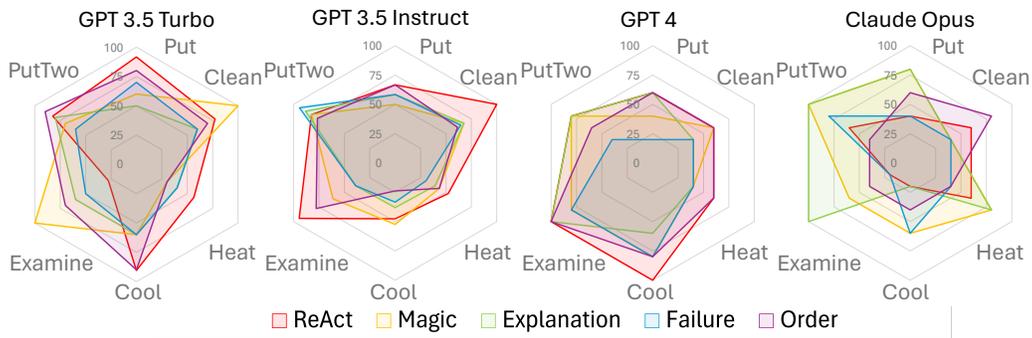


Figure 4: The radar chart shows the **failure rates** of various LLMs with different ReAct-based prompt settings for RQ2 (Base React, Magic, Failure, Failure+Explanation, Ordering) across six Alfworld tasks (hexagon vertices). Higher values / Larger shaded region indicate worse performance.

Figures 3, 4 and 5 shows a radar chart highlighting the failure cases of various LLMs with different ReAct based prompting variations for RQ1, RQ2 and RQ3 respectively on AlfWorld domain. Note that GPT-4 in the figures refer to GPT-4-0314 (Old Variant).

B.2 FAILURE RATES

We report failure rates in the radar chart as in Figs. 3, 4, 5 and 6 instead of success rates. We attempted to visualize the severe brittleness given by the larger area of the shaded region. Since, for various of

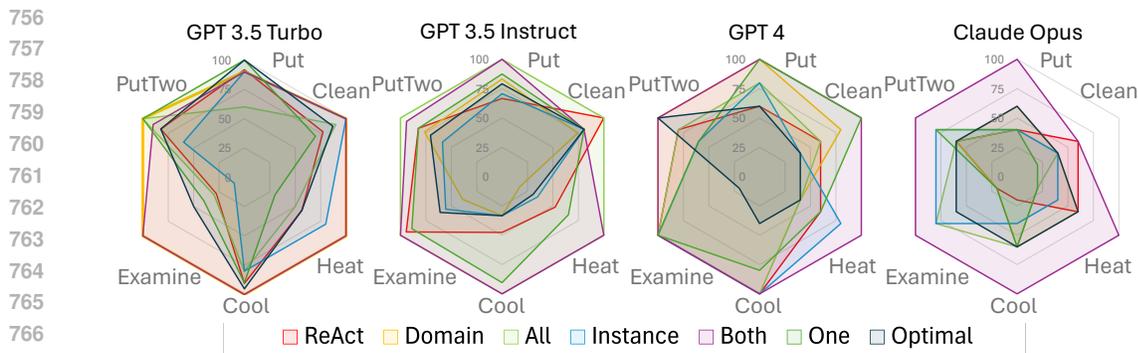


Figure 5: The radar chart shows the **failure rates** of various LLMs with different ReAct-based prompt settings for RQ3 (Base React, Domain, Instance, All, Both, One) across six Alfvorld tasks (hexagon vertices). Higher values / Larger shaded region indicate worse performance.

our RQ variations the LLMs performance was very low, we decided to report failure rate given as $(100 - \text{Success Rate } \%)$ instead.

B.3 PERFORMANCE OF CLAUDE-HAIKU

We skip on mentioning the performance of Claude-Haiku, since it was not able to generate syntactically correct actions for any of the instances. We found that following our instruction to generate specific actions as in the exemplar was difficult. We improved the prompt to have specific instructions for generating actions (See D.3) but it did not yield any improvements for Claude-Haiku. However, the instruction did help with Claude-Sonnet and Claude-Opus. We find that Claude-Sonnet follows a similar pattern as GPT-3.5-Instruct as presented in our results, and decided to focus ourselves on the strongest/largest Claude model (Claude-Opus) for our evaluation.

B.4 PERFORMANCE OF GPT-4O AND LLAMA-3.1 IN ALFWORLD

In our experiments, we found that GPT-4o obtained zero success-rate across all variations (including Act and ReAct and our proposed variations). Upon closer inspection, we found that GPT-4o requires significant effort in instruction tuning for Alfvorld, specifically, it would start emitting justifications for why a previously taken action was unsuccessful rather than generate syntactically accepted `think` tags and environment actions. Even for action generation, GPT-4o would pre-pend the actions with justifications, thereby expecting the users to write specific parsers. While stronger parsers maybe possible to implement, we hold GPT-4o to the same expectations as other models (which do not get the benefit of stronger parsers) and report our findings on a consistent evaluation.

LLAMA-3.1-8B obtains zero success-rate across all variations in Alfvorld as well. We observe that the LLAMA model would generate incorrect actions and would repeat those actions exhausting the iteration budget.

B.5 WHY IS GPT-O1 MISSING IN OUR EVALUATION BED?

We take up a large set of large language models in our evaluation and perform extensive evaluations for each of those. This is already in contrast with the baseline ReAct work which only showcased results on a single PaLM model. Our variations are agnostic to the underlying language model and code-base implements the standard `text-in-text-out` interface. However, ReAct-based agents are costly to evaluate. As of this submission, GPT-o1 incurs higher input/output token costs and also charges for generated thought tokens and would require a substantially larger budget to investigate than we can support.

B.6 EXTENSION TO OTHER MODELS

We are in the process of experimenting with GPT-4o and Google Gemini models, APIs for which were released in May 2024 which does not allow enough time for thorough and verified evaluation before the conference submission. For completeness, however, we will experiment with these APIs as they become accessible and append our results.

B.7 MAIN RESULTS ON EXEMPLAR CoT VARIANT

While this work does not investigate effectiveness of exemplar Chain of Thought as presented in RQ1, we do however test the main results of the work with Exemplar CoT to identify whether our findings hold true there as well. That is, we test RQ3-Both, RQ3-One. For GPT-3.5-Turbo we find that the average performance drops from 46.6% (RQ3-Exemplar CoT) as in Table 1 to 28.3% in One and 10.4% in Both variation cases, and remains at 40.3% for All variation.

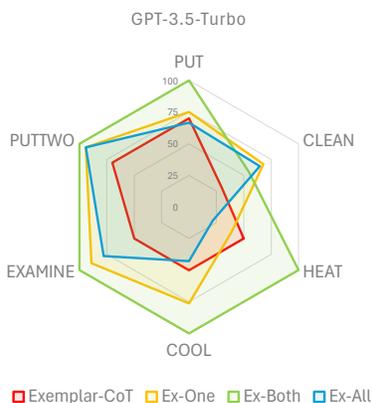


Figure 6: The radar chart shows the failure rates of GPT-3.5-Turbo with our RQ3 variants on ReAct across six Alfworld tasks. Higher values / Larger shaded region indicate worse performance.

C EXPERIMENT DESIGN

Each of the variations proposed along RQ1, RQ2 and RQ3 modifies the few-shot examples only. Remaining aspects such as the query problem or the interaction with the simulator is directly inherited from the ReAct code-base Yao et al. (2022b) at publicly available at <https://github.com/ysymyth/ReAct>. Our code can be found in the attached supplementary material.

Except All RQ3 variation, all other settings use the standard two exemplars for prompting the LLM. Depending on the variation we change the content of the exemplar. Full prompts can be found in the attached supplementary code.

C.1 RUNNING THE EXPERIMENTS

In our experiments, according to the variation style we take the exemplar prompts and use the same exemplar prompts across the instances of the query task. Other than RQ3-Both, One we use the exemplar of the same task as the query as done in ReAct (and still find brittleness of ReAct). For RQ3 - Both, One we use exactly two exemplars but of a different task than query. Finally, RQ3-All is the only variation that provides six exemplars (instead of two) and we force the exemplar of the query-task to be appended at the end in the prompt. This was the best performing prompting strategy (on GPT-3.5-Turbo) amongst when the query-task exemplar was placed at the beginning, at position 4 (middle) and at the end.

864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917

C.2 HYPERPARAMETERS

We use $temperature = \tau = 0$ for all of the GPT and Claude models and set `max-tokens = 100` which is borrowed from ReAct’s hyperparameters. Rest of the parameters are kept to be default as specified in the respective model’s API documentation.

D EXAMPLE PROMPTS

The full list of curated variations can be found in the supplementary materials. However, for completeness we reference the prompt used for base ReAct (as in (Yao et al., 2022b)) and our variations along RQ1, RQ2 and RQ3 for the `Put` task.

D.1 SYNONYM SUBSTITUTION MAPPING FOR `DOMAIN`

We make the following substitutions to the object names / locations in the exemplar prompt in the `Domain` variation. Note that these substitutions are done only to the exemplar, and the query problem and subsequent interaction with the `AlfWorld` simulator uses the original vocabulary mapping.

```
spraybottle -> aerosolbottle
cabinet -> cupboard
countertop -> worktop
sinkbasin -> sinkbowl
toilet -> lavatory
toiletpaperhanger -> toiletpaperholder
towelholder -> towelrack

microwave -> oven
shelf -> rack
drawer -> deskdrawer
stoveburner -> hob
diningtable -> table
garbagecan -> trashbin

fridge -> refrigerator
peppershaker -> pepperpot
room -> livingroom
bread -> breadloaf
pan -> fryingpan
pot -> saucepan
book -> notebook

creditcard -> amexcard
mirror -> lookingglass
dresser -> chestofdrawers
sofa -> couch
cellphone -> mobilephone
coffeemachine -> coffeemaker
knife -> kitchenknife
spatula -> turner
soapbottle -> liquidsoapdispenser
saltshaker -> saltpot
statue -> sculpture
vase -> flowerpot
dish sponge -> spongewipe
desk lamp -> table lamp
sidetable -> nightstand
```

918 D.2 FOR ALL, BOTH, ONE

919
920
921
922 **All**: We take the exemplar prompt for each task and concatenate it together.

923
924 **Both** : We use the following mapping generated arbitrarily, to replace the exemplar prompt for the
925 query task.

```
926  
927  
928  
929 VARIATION_MAPPING = {  
930     'put' : 'examine',  
931     'clean' : 'cool',  
932     'heat' : 'put',  
933     'cool' : 'puttwo',  
934     'examine' : 'clean',  
935     'puttwo' : 'heat'  
936 }  
937  
938  
939
```

940 **One** : We use the first example from the above mapping, and the second example from the same task
941 as the query.

942 943 944 945 946 947 948 D.3 BUILDING THE INPUT PROMPT

949
950
951
952 We inherit ReAct's way of stitching together the exemplars and the query using the following prompt
953 template as :

```
954  
955 Interact with a household to solve a task. Here are two  
956 examples.' + eg1 + eg2+ INSTRUCTION + Here is the task. (TASK)´
```

957 In the case of GPT models INSTRUCTION is an empty string. Claude Models gave 0% success
958 rate (regardless of the model size, i.e. Haiku, Sonnet and Opus) with the original ReAct prompt
959 design. We identified that Claude required an explicit instruction to generate only the actions for the
960 simulator. We added a task independent INSTRUCTION prompt as :

```
961 INSTRUCTION = " Only provide the one action at a time. Be  
962 concise, and do not provide any extra information. Always start  
963 with the action. For example, 'pick up apple' is correct, but 'I  
964 want to pick up the apple' is not."
```

965 Interestingly, GPT-3-Turbo performance dropped to 1% (averaged across 6 independent runs over 134
966 examples) with the inclusion of this INSTRUCTION tag. This further highlights how brittle prompt
967 engineering is, and cautions us against misplacing our trust with ReAct in enhancing reasoning
968 abilities of LLMs.

969
970 Note that, we did experiment with both prompt variations (with our explicit instruction and without)
971 for Claude models and picked the best-performing variant (which had the instructions) for our
reported results.

D.4 RQ1

Prompt 1: rq1-anon-exemplar

You are in the middle of a room. Looking quickly around you, you see a cabinet 4, a cabinet 3, a cabinet 2, a cabinet 1, a coffeemachine 1, a countertop 1, a diningtable 3, a diningtable 2, a diningtable 1, a drawer 1, a fridge 1, a garbagecan 1, a microwave 1, a sidetable 1, a sinkbasin 1, a stoveburner 4, a stoveburner 3, a stoveburner 2, a stoveburner 1, and a toaster 1.

Your task is to: find some apple and put it in sidetable.

> think: To solve the task, I need to find and take the object, then put it on the desired location. First I need to find the object. The object is more likely to appear in one of the different locations. I can check one by one, starting with the first location. Once I find the object, next I need to take it. Once I take the object, next I need to put it in/on the desired location.

OK.

> go to fridge 1

The fridge 1 is closed.

> open fridge 1

You open the fridge 1. The fridge 1 is open. In it, you see a lettuce 2, a mug 2, a potato 2, and a tomato 1.

> go to diningtable 1

On the diningtable 1, you see nothing.

> go to diningtable 2

On the diningtable 2, you see a pineapple 1.

> go to diningtable 3

On the diningtable 3, you see a bowl 2, a bowl 1, a bread 2, and a bread 1.

> go to sidetable 1

On the sidetable 1, you see a cup 1, a lettuce 1, a peppershaker 3, a potato 1, and a saltshaker 1.

> go to countertop 1

On the countertop 1, you see a butterknife 1, a dish sponge 1, and a soapbottle 1.

> go to sinkbasin 1

On the sinkbasin 1, you see a cup 2, a egg 1, a mug 1, and a potato 3.

> go to garbagecan 1

On the garbagecan 1, you see a apple 3, and a egg 3.

> take apple 3 from garbagecan 1

You pick up the apple 3 from the garbagecan 1.

> go to sidetable 1

On the sidetable 1, you see a cup 1, a lettuce 1, a peppershaker 3, a potato 1, and a saltshaker 1.

> put apple 3 in/on sidetable 1

You put the apple 3 in/on the sidetable 1.

Prompt 2: rq1-exemplar

You are in the middle of a room. Looking quickly around you, you see a cabinet 4, a cabinet 3, a cabinet 2, a cabinet 1, a coffeemachine 1, a countertop 1, a diningtable 3, a diningtable 2, a diningtable 1, a drawer 1, a fridge 1, a garbagecan 1, a microwave 1, a sidetable 1, a sinkbasin 1, a stoveburner 4, a stoveburner 3, a stoveburner 2, a stoveburner 1, and a toaster 1.

Your task is to: find some apple and put it in sidetable.

> think: To solve the task, I need to find and take an apple, then put it in sidetable. First I need to find an apple. An apple is more likely to appear in fridges (1), diningtables (1-3), sidetables (1), countertops (1), sinkbasins (1), garbagecan (1). I can check one by one, starting with fridge 1. Once I find an apple, next I need to take it. Once I find an apple, next I need to put it in/on sidetable.

OK.

> go to fridge 1

The fridge 1 is closed.

> open fridge 1

You open the fridge 1. The fridge 1 is open. In it, you see a lettuce 2, a mug 2, a potato 2, and a tomato 1.

> go to diningtable 1

On the diningtable 1, you see nothing.

> go to diningtable 2

On the diningtable 2, you see a pineapple 1.

> go to diningtable 3

On the diningtable 3, you see a bowl 2, a bowl 1, a bread 2, and a bread 1.

> go to sidetable 1

On the sidetable 1, you see a cup 1, a lettuce 1, a peppershaker 3, a potato 1, and a saltshaker 1.

> go to countertop 1

On the countertop 1, you see a butterknife 1, a dish sponge 1, and a soapbottle 1.

> go to sinkbasin 1

On the sinkbasin 1, you see a cup 2, a egg 1, a mug 1, and a potato 3.

> go to garbagecan 1

On the garbagecan 1, you see a apple 3, and a egg 3.

> take apple 3 from garbagecan 1

You pick up the apple 3 from the garbagecan 1.

> go to sidetable 1

On the sidetable 1, you see a cup 1, a lettuce 1, a peppershaker 3, a potato 1, and a saltshaker 1.

> put apple 3 in/on sidetable 1

You put the apple 3 in/on the sidetable 1.

1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079

Prompt 3: base-react

You are in the middle of a room. Looking quickly around you, you see a cabinet 4, a cabinet 3, a cabinet 2, a cabinet 1, a coffeemachine 1, a countertop 1, a diningtable 3, a diningtable 2, a diningtable 1, a drawer 1, a fridge 1, a garbagecan 1, a microwave 1, a sidetable 1, a sinkbasin 1, a stoveburner 4, a stoveburner 3, a stoveburner 2, a stoveburner 1, and a toaster 1.

Your task is to: find some apple and put it in sidetable.

> think: First I need to find an apple. An apple is more likely to appear in fridges (1), diningtables (1-3), sidetables (1), countertops (1), sinkbasins (1), garbagecan (1). I can check one by one, starting with fridge 1.

OK.

> go to fridge 1

The fridge 1 is closed.

> open fridge 1

You open the fridge 1. The fridge 1 is open. In it, you see a lettuce 2, a mug 2, a potato 2, and a tomato 1.

> go to diningtable 1

On the diningtable 1, you see nothing.

> go to diningtable 2

On the diningtable 2, you see a pineapple 1.

> go to diningtable 3

On the diningtable 3, you see a bowl 2, a bowl 1, a bread 2, and a bread 1.

> go to sidetable 1

On the sidetable 1, you see a cup 1, a lettuce 1, a peppershaker 3, a potato 1, and a saltshaker 1.

> go to countertop 1

On the countertop 1, you see a butterknife 1, a dish sponge 1, and a soapbottle 1.

> go to sinkbasin 1

On the sinkbasin 1, you see a cup 2, a egg 1, a mug 1, and a potato 3.

> go to garbagecan 1

On the garbagecan 1, you see a apple 3, and a egg 3.

> take apple 3 from garbagecan 1

You pick up the apple 3 from the garbagecan 1.

> think: Now I find a apple (3). Next, I need to put it in/on sidetable 1.

OK.

> go to sidetable 1

On the sidetable 1, you see a cup 1, a lettuce 1, a peppershaker 3, a potato 1, and a saltshaker 1.

> put apple 3 in/on sidetable 1

You put the apple 3 in/on the sidetable 1.

1080 D.5 RQ2

1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133

Prompt 4: rq2-Explanation

You are in the middle of a room. Looking quickly around you, you see a cabinet 4, a cabinet 3, a cabinet 2, a cabinet 1, a coffeemachine 1, a countertop 1, a diningtable 3, a diningtable 2, a diningtable 1, a drawer 1, a fridge 1, a garbagecan 1, a microwave 1, a sidetable 1, a sinkbasin 1, a stoveburner 4, a stoveburner 3, a stoveburner 2, a stoveburner 1, and a toaster 1.

Your task is to: find some apple and put it in sidetable.

> go to fridge 1
The fridge 1 is closed.
> close cabinet 2.
Nothing Happens.
> think: Nothing happens because you are not at cabinet 2.
OK.
> open fridge 1
You open the fridge 1. The fridge 1 is open. In it, you see a lettuce 2, a mug 2, a potato 2, and a tomato 1.
> go to diningtable 1
On the diningtable 1, you see nothing.
> go to diningtable 2
On the diningtable 2, you see a pineapple 1.
> go to diningtable 3
On the diningtable 3, you see a bowl 2, a bowl 1, a bread 2, and a bread 1.
> go to sidetable 1
On the sidetable 1, you see a cup 1, a lettuce 1, a peppershaker 3, a potato 1, and a saltshaker 1.
> go to countertop 1
On the countertop 1, you see a butterknife 1, a dish sponge 1, and a soapbottle 1.
> go to sinkbasin 1
On the sinkbasin 1, you see a cup 2, a egg 1, a mug 1, and a potato 3.
> go to garbagecan 1
On the garbagecan 1, you see a apple 3, and a egg 3.
> put apple 3 in/on sidetable.
Nothing Happens.
> think: Nothing happens because I do not have apple 3.
OK.
> take apple 3 from garbagecan 1
You pick up the apple 3 from the garbagecan 1.
> go to sidetable 1
On the sidetable 1, you see a cup 1, a lettuce 1, a peppershaker 3, a potato 1, and a saltshaker 1.
> put apple 3 in/on sidetable 1
You put the apple 3 in/on the sidetable 1.

1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187

Prompt 5: rq2-ordering

You are in the middle of a room. Looking quickly around you, you see a cabinet 4, a cabinet 3, a cabinet 2, a cabinet 1, a coffeemachine 1, a countertop 1, a diningtable 3, a diningtable 2, a diningtable 1, a drawer 1, a fridge 1, a garbagecan 1, a microwave 1, a sidetable 1, a sinkbasin 1, a stoveburner 4, a stoveburner 3, a stoveburner 2, a stoveburner 1, and a toaster 1.
Your task is to: find some apple and put it in sidetable.
> think: To solve the task, I need to put an apple in sidetable, for which I need to find and take an apple.
OK.
> think: An apple is more likely to appear in sidetables (1), fridges (1), countertops (1), diningtables (1-3), garbagecan (1), sinkbasins (1). First I need to find an apple. I can check one by one, starting with fridge 1.
OK.
> go to fridge 1
The fridge 1 is closed.
> open fridge 1
You open the fridge 1. The fridge 1 is open. In it, you see a lettuce 2, a mug 2, a potato 2, and a tomato 1.
> go to diningtable 1
On the diningtable 1, you see nothing.
> go to diningtable 2
On the diningtable 2, you see a pineapple 1.
> go to diningtable 3
On the diningtable 3, you see a bowl 2, a bowl 1, a bread 2, and a bread 1.
> go to sidetable 1
On the sidetable 1, you see a cup 1, a lettuce 1, a peppershaker 3, a potato 1, and a saltshaker 1.
> go to countertop 1
On the countertop 1, you see a butterknife 1, a dish sponge 1, and a soapbottle 1.
> go to sinkbasin 1
On the sinkbasin 1, you see a cup 2, a egg 1, a mug 1, and a potato 3.
> go to garbagecan 1
On the garbagecan 1, you see a apple 3, and a egg 3.
> think: Next, I need to take the apple. Now I find an apple (3).
OK.
> take apple 3 from garbagecan 1
You pick up the apple 3 from the garbagecan 1.
> think: Next, I need to put the apple in/on sidetable 1. Now I find an apple (3).
OK.
> go to sidetable 1
On the sidetable 1, you see a cup 1, a lettuce 1, a peppershaker 3, a potato 1, and a saltshaker 1.
> put apple 3 in/on sidetable 1
You put the apple 3 in/on the sidetable 1.

Prompt 6: rq2-Placebo

You are in the middle of a room. Looking quickly around you, you see a cabinet 4, a cabinet 3, a cabinet 2, a cabinet 1, a coffeemachine 1, a countertop 1, a diningtable 3, a diningtable 2, a diningtable 1, a drawer 1, a fridge 1, a garbagecan 1, a microwave 1, a sidetable 1, a sinkbasin 1, a stoveburner 4, a stoveburner 3, a stoveburner 2, a stoveburner 1, and a toaster 1.
Your task is to: find some apple and put it in sidetable.
> think: Take a deep breath and work on this problem step-by-step.
OK.
> go to fridge 1
The fridge 1 is closed.
> open fridge 1
You open the fridge 1. The fridge 1 is open. In it, you see a lettuce 2, a mug 2, a potato 2, and a tomato 1.
> go to diningtable 1
On the diningtable 1, you see nothing.
> go to diningtable 2
On the diningtable 2, you see a pineapple 1.
> go to diningtable 3
On the diningtable 3, you see a bowl 2, a bowl 1, a bread 2, and a bread 1.
> go to sidetable 1
On the sidetable 1, you see a cup 1, a lettuce 1, a peppershaker 3, a potato 1, and a saltshaker 1.
> go to countertop 1
On the countertop 1, you see a butterknife 1, a dish sponge 1, and a soapbottle 1.
> go to sinkbasin 1
On the sinkbasin 1, you see a cup 2, a egg 1, a mug 1, and a potato 3.
> go to garbagecan 1
On the garbagecan 1, you see a apple 3, and a egg 3.
> think: Take a deep breath and work on this problem step-by-step.
OK.
> take apple 3 from garbagecan 1
You pick up the apple 3 from the garbagecan 1.
> think: Take a deep breath and work on this problem step-by-step.
OK.
> go to sidetable 1
On the sidetable 1, you see a cup 1, a lettuce 1, a peppershaker 3, a potato 1, and a saltshaker 1.
> put apple 3 in/on sidetable 1
You put the apple 3 in/on the sidetable 1.

1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241

Prompt 7: rq2-Failure

You are in the middle of a room. Looking quickly around you, you see a cabinet 4, a cabinet 3, a cabinet 2, a cabinet 1, a coffeemachine 1, a countertop 1, a diningtable 3, a diningtable 2, a diningtable 1, a drawer 1, a fridge 1, a garbagecan 1, a microwave 1, a sidetable 1, a sinkbasin 1, a stoveburner 4, a stoveburner 3, a stoveburner 2, a stoveburner 1, and a toaster 1.
Your task is to: find some apple and put it in sidetable.
> go to fridge 1
The fridge 1 is closed.
> close cabinet 2.
Nothing Happens.
> open fridge 1
You open the fridge 1. The fridge 1 is open. In it, you see a lettuce 2, a mug 2, a potato 2, and a tomato 1.
> go to diningtable 1
On the diningtable 1, you see nothing.
> go to diningtable 2
On the diningtable 2, you see a pineapple 1.
> go to diningtable 3
On the diningtable 3, you see a bowl 2, a bowl 1, a bread 2, and a bread 1.
> go to sidetable 1
On the sidetable 1, you see a cup 1, a lettuce 1, a peppershaker 3, a potato 1, and a saltshaker 1.
> go to countertop 1
On the countertop 1, you see a butterknife 1, a dish sponge 1, and a soapbottle 1.
> go to sinkbasin 1
On the sinkbasin 1, you see a cup 2, a egg 1, a mug 1, and a potato 3.
> go to garbagecan 1
On the garbagecan 1, you see a apple 3, and a egg 3.
> put apple 3 in/on sidetable.
Nothing Happens.
> take apple 3 from garbagecan 1
You pick up the apple 3 from the garbagecan 1.
> go to sidetable 1
On the sidetable 1, you see a cup 1, a lettuce 1, a peppershaker 3, a potato 1, and a saltshaker 1.
> put apple 3 in/on sidetable 1
You put the apple 3 in/on the sidetable 1.

1242 D.6 RQ3

1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295

Prompt 8: rq3-Domain

You are in the middle of a livingroom. Looking quickly around you, you see a cupboard 4, a cupboard 3, a cupboard 2, a cupboard 1, a coffeemaker 1, a worktop 1, a table 3, a table 2, a table 1, a deskdrawer 1, a refrigerator 1, a trashbin 1, an oven 1, a nightstand 1, a sinkbowl 1, a hob 4, a hob 3, a hob 2, a hob 1, and a toaster 1.

Your task is to: find some apple and put it in nightstand.

> think: To solve the task, I need to find and take an apple, then put it in nightstand.

OK.

> think: First I need to find an apple. An apple is more likely to appear in refrigerators (1), tables (1-3), nightstands (1), worktops (1), sinkbowls (1), trashbin (1). I can check one by one, starting with refrigerator 1.

OK.

> go to refrigerator 1

The refrigerator 1 is closed.

> oballpointpen refrigerator 1

You oballpointpen the refrigerator 1. The refrigerator 1 is oballpointpen. In it, you see a lettuce 2, a mug 2, a saucepanato 2, and a tomato 1.

> go to table 1

On the table 1, you see nothing.

> go to table 2

On the table 2, you see a pineapple 1.

> go to table 3

On the table 3, you see a bowl 2, a bowl 1, a breadloaf 2, and a breadloaf 1.

> go to nightstand 1

On the nightstand 1, you see a cup 1, a lettuce 1, a peppersaucepan 3, a saucepanato 1, and a saltpot 1.

> go to worktop 1

On the worktop 1, you see a butterknife 1, a spongewipe 1, and a liquidsoapdispenser 1.

> go to sinkbowl 1

On the sinkbowl 1, you see a cup 2, an egg 1, a mug 1, and a saucepanato 3.

> go to trashbin 1

On the trashbin 1, you see an apple 3, and an egg 3.

> think: Now I find an apple (3). Next, I need to take it.

> take apple 3 from trashbin 1

You pick up the apple 3 from the trashbin 1.

> think: Now I find an apple (3). Next, I need to put it in/on nightstand 1.

OK.

> go to nightstand 1

On the nightstand 1, you see a cup 1, a lettuce 1, a peppersaucepan 3, a saucepanato 1, and a saltpot 1.

> put apple 3 in/on nightstand 1

You put the apple 3 in/on the nightstand 1.

1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349

Prompt 9: rq3-Instance

You are in the middle of a room. Looking quickly around you, you see a cabinet 4, a cabinet 3, a cabinet 2, a cabinet 1, a coffeemachine 1, a countertop 1, a diningtable 3, a diningtable 2, a diningtable 1, a drawer 1, a fridge 1, a garbagecan 1, a microwave 1, a sidetable 1, a sinkbasin 1, a stoveburner 4, a stoveburner 3, a stoveburner 2, a stoveburner 1, and a toaster 1.
Your task is to: find some apple and put it in sidetable.
> think: To solve the task, I need to find and take an apple, then put it in sidetable.
OK.
> think: First I need to find an apple. An apple is more likely to appear in fridges (1), diningtables (1-3), sidetables (1), countertops (1), sinkbasins (1), garbagecan (1). I can check one by one, starting with fridge 1.
OK.
> go to fridge 1
The fridge 1 is closed.
> open fridge 1
You open the fridge 1. The fridge 1 is open. In it, you see a lettuce 2, a mug 2, a potato 2, and a tomato 1.
> go to diningtable 1
On the diningtable 1, you see nothing.
> go to diningtable 2
On the diningtable 2, you see a pineapple 1.
> go to diningtable 3
On the diningtable 3, you see a bowl 2, a bowl 1, a bread 2, and a bread 1.
> go to sidetable 1
On the sidetable 1, you see a cup 1, a lettuce 1, a peppershaker 3, a potato 1, and a saltshaker 1.
> go to countertop 1
On the countertop 1, you see a butterknife 1, a dish sponge 1, and a soapbottle 1.
> go to sinkbasin 1
On the sinkbasin 1, you see a cup 2, a egg 1, a mug 1, and a potato 3.
> go to diningtable 2
On the diningtable 2, you see a pineapple 1.
> go to diningtable 3
On the diningtable 3, you see a bowl 2, a bowl 1, a bread 2, and a bread 1.
> go to sidetable 1
On the sidetable 1, you see a cup 1, a lettuce 1, a peppershaker 3, a potato 1, and a saltshaker 1.
> go to countertop 1
On the countertop 1, you see a butterknife 1, a dish sponge 1, and a soapbottle 1.
> go to sinkbasin 1
On the sinkbasin 1, you see a cup 2, a egg 1, a mug 1, and a potato 3.
> go to garbagecan 1
On the garbagecan 1, you see a apple 3, and a egg 3.
> think: Now I find an apple (3). Next, I need to take it.
> take apple 3 from garbagecan 1
You pick up the apple 3 from the garbagecan 1.
> think: Now I find an apple (3). Next, I need to put it in/on sidetable 1.
OK.
> go to sidetable 1
On the sidetable 1, you see a cup 1, a lettuce 1, a peppershaker 3, a potato 1, and a saltshaker 1.
> put apple 3 in/on sidetable 1
You put the apple 3 in/on the sidetable 1.

Prompt 10: rq3-Optimal

You are in the middle of a room. Looking quickly around you, you see a cabinet 4, a cabinet 3, a cabinet 2, a cabinet 1, a coffeemachine 1, a countertop 1, a diningtable 3, a diningtable 2, a diningtable 1, a drawer 1, a fridge 1, a garbagecan 1, a microwave 1, a sidetable 1, a sinkbasin 1, a stoveburner 4, a stoveburner 3, a stoveburner 2, a stoveburner 1, and a toaster 1.
Your task is to: find some apple and put it in sidetable.
> think: To solve the task, I need to find and take an apple, then put it in sidetable.
OK.
> think: First I need to find an apple. An apple is more likely to appear in fridges (1), diningtables (1-3), sidetables (1), countertops (1), sinkbasins (1), garbagecan (1). I can check one by one, starting with garbagecan 1.
OK.
> go to garbagecan 1
On the garbagecan 1, you see a apple 3, and a egg 3.
> think: Now I find an apple (3). Next, I need to take it.
> take apple 3 from garbagecan 1
You pick up the apple 3 from the garbagecan 1.
> think: Now I find an apple (3). Next, I need to put it in/on sidetable 1.
OK.
> go to sidetable 1
On the sidetable 1, you see a cup 1, a lettuce 1, a peppershaker 3, a potato 1, and a saltshaker 1.
> put apple 3 in/on sidetable 1
You put the apple 3 in/on the sidetable 1.