Language Models can Solve Computer Tasks

Geunwoo Kim [1] **Pierre Baldi** [1] **Stephen McAleer** [2]

Abstract

Agents capable of carrying out general tasks on a computer can improve efficiency and productivity by automating repetitive tasks and assisting in complex problem-solving. Ideally, such agents should be able to solve new computer tasks presented to them through natural language commands. However, previous approaches to this problem require large amounts of expert demonstrations and task-specific reward functions, both of which are impractical for new tasks. In this work, we show that a pre-trained large language model (LLM) agent can execute computer tasks guided by natural language using a simple prompting scheme where the agent **R**ecursively **C**riticizes and **I**mproves its output (RCI). The RCI approach significantly outperforms existing LLM methods for automating computer tasks and surpasses supervised learning (SL) and reinforcement learning (RL) approaches on the MiniWoB++ benchmark. We compare multiple LLMs and find that RCI with the InstructGPT-3+RLHF LLM is state-of-the-art on MiniWoB++, using only a handful of demonstrations per task rather than tens of thousands, and without a task-specific reward function. Furthermore, we demonstrate RCI prompting's effectiveness in enhancing LLMs' reasoning abilities on a suite of natural language reasoning tasks, outperforming chain of thought (CoT) prompting. We find that RCI combined with CoT performs better than either separately. Our code can be found here: https://github.com/posgnu/rci-agent.

1. Introduction

A long-standing goal in artificial intelligence has been to create generally-intelligent agents that can accomplish cognitive tasks as well as humans. Such agents should be able to solve any computer task a human can by communicating via natural language. By automating repetitive tasks and providing assistance in complex problem-solving, generally-intelligent virtual agents may radically increase productivity.

Recently, large language models (LLMs) have shown remarkable in-context learning capabilities across a variety of domains and tasks (Dai et al., 2022; von Oswald et al., 2022; Brown et al., 2020; Du et al., 2022; Hoffmann et al., 2022; Smith et al., 2022; Chowdhery et al., 2022; OpenAI, 2023; Bubeck et al., 2023). Although LLMs can impressively manipulate text and can use high-level API tools (Schick et al., 2023; Paranjape et al., 2023; Mialon et al., 2023), previous approaches to using LLMs that directly take keyboard and mouse actions on computers have had difficulty compared to imitation learning and reinforcement learning approaches (Gur et al., 2022). LLMs that take keyboard and mouse actions on computers face a number of obstacles, such as ensuring that generated actions are task-appropriate (task grounding), feasible in the agent's current state (state grounding), and admissible to be executed (agent grounding).

The previous best-performing approaches for taking actions on computers have not used LLMs. Instead, they have trained networks from scratch to predict actions given prompts and screenshots or DOM information, either via supervised learning (SL) from expert demonstrations, reinforcement learning (RL) on a handcrafted reward signal, or both (SL+RL) (Humphreys et al., 2022). Although SL+RL works well on a number of individual computer tasks, since it requires expert data and a reward function for every task, it has not been shown to generalize to novel tasks in a few-shot setting.

In this work, we show that a pre-trained LLM agent can successfully execute computer tasks guided by natural language. Our method employs a simple prompting scheme, which we call Recursive Criticism and Improvement (RCI), that significantly outperforms existing LLM methods for automating computer tasks. RCI works by first having the LLM generate an output based on zero-shot prompting. Then, RCI prompts the LLM to identify problems with the given output. After the LLM has identified problems with the output, RCI prompts the LLM to generate an updated output.

[1]University of California, Irvine [2]Carnegie Mellon University. Correspondence to: Stephen McAleer <smcaleer@cs.cmu.edu>.

AI & HCI Workshop at the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. 2023.

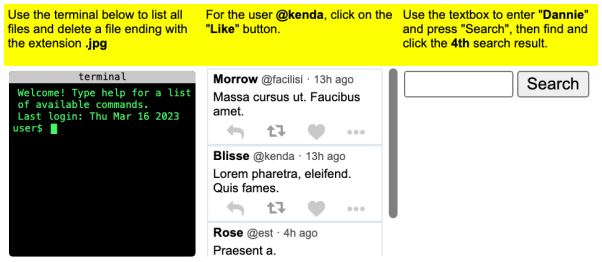


Figure 1: MiniWoB++ environment. Every task contains a natural language prompt in yellow. The agent then uses keyboard strokes and mouse clicks to accomplish the task.

When applying RCI to computer tasks, we improve task grounding, state grounding, and agent grounding sequentially. Firstly, task grounding prompts the Language Model (LLM) with the task text, instructing it to generate a high-level plan. Secondly, state grounding connects high-level concepts derived from the task grounding step with actual HTML elements present in the current state, subsequently outputting the appropriate action. Finally, agent grounding ensures the correct formatting of the action output obtained from the state grounding step. RCI is applied to each of these three steps; however, we find that critiquing the state-grounding step is only necessary once.

We evaluate the RCI approach on the MiniWoB++ benchmark (Shi et al., 2017), and show it surpasses existing SL, RL, and LLM approaches. Furthermore, it proves itself to state-of-the-art compared to existing methods, using only a small number of demonstrations per task instead of tens of thousands, and without relying on a task-specific reward function. This significant reduction in required demonstrations and the elimination of task-specific reward functions make our method more practical and accessible for new tasks. Furthermore, as the capabilities of LLMs continue to improve, one can expect the performance of our method to improve as well.

In addition to its success in automating computer tasks, we also showcase the effectiveness of RCI prompting in enhancing the reasoning abilities of LLMs on a suite of natural language reasoning tasks. Our method achieves a significant performance increase over zero-shot prompting and slightly improves upon chain-of-thought (Wei et al., 2022b) (CoT) prompting. Interestingly, RCI and CoT have a synergistic effect, and their combination outperforms all other methods.

In summary, our work presents a new powerful and practical approach to enabling LLM agents to execute computer tasks guided by natural language. The RCI prompting scheme not only outperforms previous methods in computer tasks, but also improves reasoning abilities for LLMs more broadly, making it a significant contribution in the development of intelligent agents.

2. Methods

2.1. RCI Prompting

The self-critiquing ability of LLMs has demonstrated that LLMs can find errors in their own output by themselves (Saunders et al., 2022; Ganguli et al., 2023; Bai et al., 2022). In light of this, we introduce a simple reasoning architecture called RCI prompting, where we prompt LLMs to find problems in their output and improve the output based on what they find. This architecture is designed to further enhance the reasoning ability of LLMs by inserting a critique step before generating the final answer. Figure 2 compares example traces of RCI prompting and baseline prompting methods on GSM8K dataset where language models should answer grade school math problems. While baselines elicit answers with a single step of prompting, RCI consists of two steps: criticize the previous answer (*e.g.,* "Review your previous answer and find problems with your answer") and improve the answer based on the critique (*e.g.,* "Based on the problems you found, improve your answer"). In this way, RCI prompting finds errors (*e.g.,* the overall sum of money only considered Valerie and her brother) in the previous answer and generates an improved answer (*e.g.,* money from Valerie's mother is included in the total) conditioned on the critique. The iterative process of RCI can be continued until specific conditions are satisfied, which could include receiving feedback from the environment, reaching the maximum predetermined number of iterations, or adhering to certain heuristics. We define two approaches for achieving RCI: explicit RCI and implicit RCI. Explicit RCI includes the critique in the prompt to generate improved output and implicit RCI updates the previous output directly without sampling a critique explicitly. Examples of explicit RCI and implicit RCI applied to computer tasks are illustrated in Figure 3 where the action plan is improved based on an explicit critique while actions are updated without an explicit critique. This process is described in more detail in the following section.

2.2. RCI for Computer Tasks

In this section we describe the application of RCI to computer tasks via a decomposition of action selection into three reasoning steps: task grounding, state grounding, and agent grounding. The first step, task grounding, involves generating a plan for task-solving and conditioning actions on this plan, with RCI being used to improve the plan's success rate. The state grounding subsection discusses the importance of grounding actions in the environment for language-based agents and how implicit RCI is used to refine task-grounded actions to be feasible in the current state. Lastly, the agent grounding step focuses on ensuring that actions are admissi-

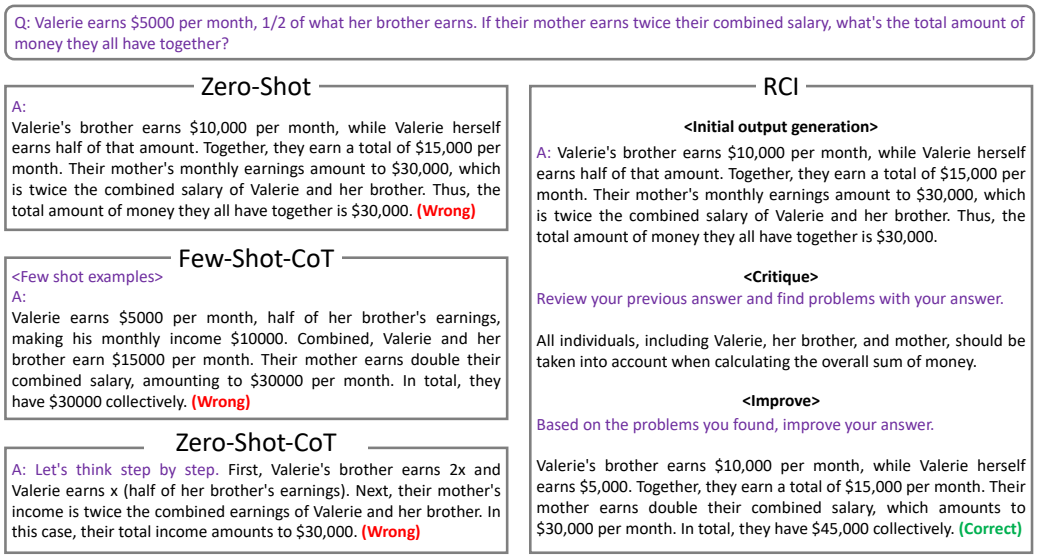


Figure 2: Illustrative examples of explicit RCI prompting and baseline prompting approaches on the GSM8K dataset. RCI prompting effectively addresses logical errors that arise in the baseline prompting approaches. Prompts text is displayed in violet color.

ble for the computer agent by employing implicit RCI and conditioning agent-grounded actions on the current state, task, and other grounded actions, with a loop count set to optimize performance.

2.2.1. Problem Setting

We assume that we are given an instruction-following computer agent that can execute a set of admissible actions given some natural language instructions. An instruction that is not part of the admissible actions will be ignored. At every step, we receive a high-level natural language task prompt and a state of the environment. Given the current state and task, we sample the most probable action from LLMs. The generated natural language action is then fed into the computer agent. Sampling the actions in a fully generative manner presents a challenge, as the actions must consider the given task, feasibility in the current state, and admissibility for the computer agent simultaneously. Therefore, we propose decomposing this action sampling into three reasoning steps each of which considers task grounding, state grounding, and agent grounding. Task grounding improves actions to be more effective in solving the given task, state grounding ensures the feasibility of actions in the current state, and agent grounding considers the executability of actions given the specification of the computer agent. We first sample a step-by-step plan to solve the given task which improves the task grounding. Next, the task-grounded action is sampled conditioned on the current state, task, and the generated plan. The state-grounded actions is generated conditioned on the task-grounded action. If the task-grounded action is not executable by the computer agent, the agent-grounded action is sampled. For each sampling of grounded action, we use RCI prompting to make LLM consider some specific information for grounding.

2.2.2. Grounding Language Model in Computer Tasks

Task grounding. In the action sampling process, the first step involves generating a plan of actionable steps for task solving from LLMs. Subsequently, actions are sampled from the same LLMs, taking into account the present state, task, and generated plan. The benefits of conditioning on the plan for improved grounding of actions are twofold. First, it enables LLMs to identify the stage of task solving at which the agent is located, serving as a memory module. Second, we can perform explicit RCI on the generated plan to further improve the plan's success rate. Although the number of explicit RCI loops can be arbitrary, we observe that a single pass of explicit RCI suffices for most of MiniWoB++ tasks.

State grounding. In language-based agents, grounding actions in the environment is a crucial step to enable real-world task performance. The aim of this phase is to enhance the task-grounded actions to be feasible in the current state. Although the actions generated in the preceding phase may align with the task, they may lack the specificity required to be executed in the current context. For example, if the assigned task is to forward an email from Bob to Alice and the action obtained from the task grounding phase is to click on an email from Bob in the email inbox, it is necessary to establish a connection between the abstract concept of "email from Bob" and the concrete element, such as the

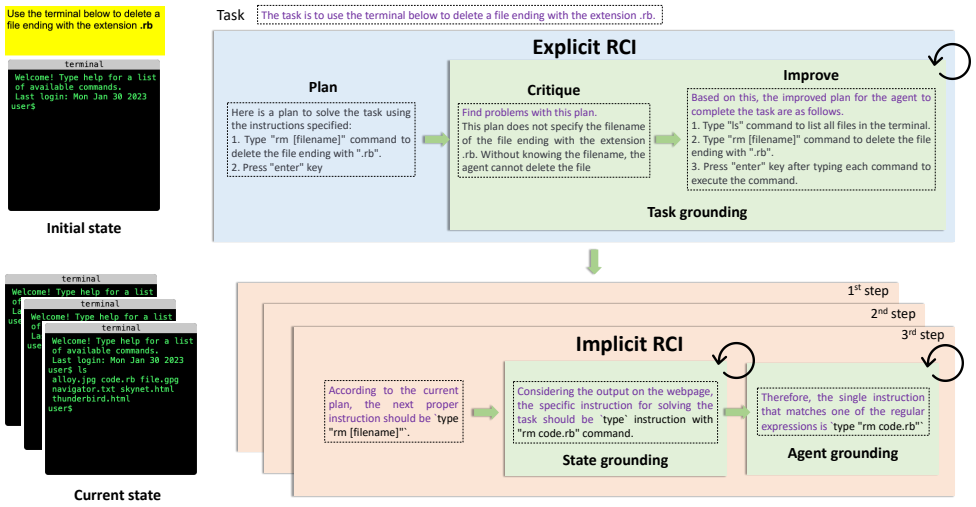


Figure 3: An illustrative execution trace of the agent for terminal tasks with RCI prompting. The language model generates a step-by-step plan for the high-level task described in natural language, which in this case involves using the terminal to delete a file ending with ".rb". We then run an explicit RCI on this plan, where we sample an improved plan based on the critique and the previous plan, resulting in an improvement in the task-grounding of the plan. For each step, we first sample the task-grounded action that follows the improved plan, and then the implicit RCI updates the task-grounded actions sequentially to provide state-grounding and agent-grounding. Finally, the agent-grounded action is executed by the instruction-following agent on the environment. The prompts are highlighted, and the remaining text shows the outputs generated by the language model.

email heading, in the current webpage state represented by HTML. To achieve this goal, we perform the implicit RCI and prompt the LLMs to consider the current state, which subsequently outputs refined state-grounded actions. Moreover, the state-grounded action is additionally conditioned on the task-grounded action. We avoid repeating the implicit RCI cycle more than once as it does not impact the success rate based on our observations.

Agent grounding. To ensure the successful integration of language-based methodologies in decision-making processes, it is imperative to establish a scalable framework that guarantees the admissibility of actions derived from the language model. While the preceding steps of sampling produce a state-grounded action that is both feasible and grounded in the task, it may not be executable by the agent due to issues such as improper formatting. To address this, Implicit RCI is employed, whereby an agent-grounded action is sampled conditioned on the current state, task, task-grounded action, and state-grounded action. The LLMs are prompted to consider specifications of the computer agent. The implicit RCI is repeatedly run until the resulting action is executable, with a maximum loop count set to limit the number of iterations. Empirical analysis on MiniWoB++ tasks suggests that setting the loop count to 3 yields optimal performance.

3. Evaluation

3.1. Reasoning tasks

In our grounding enhancement process, RCI prompts the LLM to criticize its prior output, considering the given context (*e.g.,* current task, state, and agent), which ultimately leads to improved output. We first demonstrate the effectiveness of RCI prompts in augmenting the reasoning capabilities of LLMs across a range of reasoning benchmarks. We compare RCI to Chain-of-Thought (CoT) prompting, a state-of-the-art method recognized for its effectiveness in reasoning tasks.

Specifically, we compare our approach with Few-Shot-CoT (Wei et al., 2022b) where a few chain-of-thought demonstrations are given as examples in prompting, and Zero-Shot-CoT (Kojima et al., 2022) that elicit multiple reasoning steps by simply adding "Let's think step by step" to the prompt. Following Kojima et al. (Kojima et al., 2022), our evaluation is conducted with 8 datasets from two categories of reasoning: arithmetic and commonsense. Please refer to Appendix C.2 for a comprehensive depiction of the datasets. We use the same experimental setting with their answer extraction method except that we use InstructGPT-3 + RLHF as the underlying language model. We use the same prompts that CoT uses and we also use the answer cleansing approach used in CoT, but we only used answer extraction prompting in zero-shot CoT experiments. We also use the

	Arithmetic						Common Sense	
	GSM8K	MultiArith	AddSub	SVAMP	SingleEq	AQuA	CommonSenseQA	StrategyQA
Zero-Shot	78.35	96.06	85.83	78.35	91.34	55.91	53.15	51.57
Zero-Shot + RCI	**85.43**	**97.64**	**89.76**	**84.65**	**94.49**	**67.32**	**68.11**	**61.81**

Table 1: RCI prompting increases the reasoning capability of LLMs on all of eight reasoning benchmarks.

same few-shot examples that were introduced in (Wei et al., 2022b) to evaluate Few-Shot CoT's performance on five arithmetic reasoning tasks.

Comparison with Zero-Shot. RCI prompting is better at solving reasoning tasks compared to zero-shot prompting. Table 1 summarizes the accuracy of our approach (Zero-Shot + RCI) and standard zero-shot prompting for each reasoning benchmark. Zero-Shot + RCI substantially outperforms the standard prompting in all benchmarks including arithmetic (GSM8K, MultiArith, AddSub, AQUA, SVAMP, SingleEq) and common sense (CommonSenseQA, StrategyQA) tasks. RCI prompting even achieves score gains from two arithmetic reasoning tasks (SingleEq and AddSub), which do not require multi-step reasoning. This distinguishes our RCI prompting from the previous CoT prompting methods (Wei et al., 2022b; Kojima et al., 2022) that are not useful in simple reasoning tasks. It is also worth noting that RCI prompting achieves a significant performance gain in commonsense reasoning tasks (CommonSenseQA and StrategyQA). While Wei et al. (Wei et al., 2022b) reported that only a substantially large PaLM (540B) model can benefit from Few-Shot-CoT, RCI prompting can provide performance gain even with a smaller InstructGPT-3 + RLHF (175B) model.

Comparison with Chain-of-Thought. The performance results of RCI and CoT baselines on arithmetic reasoning tasks are summarized in Table 2. Notably, Zero-Shot + RCI outperforms Zero-Shot CoT and Few-Shot CoT without any CoT prompting in four tasks except *MultiArith*. In *MultiArith* tasks, where most of the standard prompting's answers are correct (96.06%), RCI prompting does not yield significant performance gains. RCI prompting has a synergistic collaborative impact on the two CoT baselines. Namely, Zero-Shot CoT + RCI and Few-Shot CoT + RCI attain the highest scores on four out of the five tasks. These findings suggest a promising avenue for future research: combining RCI with other prompting methods for CoT, such as self-consistency (Saunders et al., 2022).

	GSM8K	MultiArith	AddSub	SVAMP	SingleEq
Zero-Shot	78.35	96.06	85.83	78.35	91.34
Zero-Shot + RCI	85.43	97.64	89.76	84.65	**94.49**
Zero-Shot CoT	82.28	96.85	83.86	79.92	89.37
Zero-Shot CoT + RCI	**86.22**	97.24	89.88	85.83	90.94
Few-Shot CoT	80.31	98.82	89.37	83.46	91.73
Few-Shot CoT + RCI	84.25	**99.21**	**90.55**	**87.40**	93.70

Table 2: Chain-of-Thought prompting exhibits a synergistic effect when coupled with RCI prompting in arithmetic reasoning tasks.

3.2. Computer tasks

3.2.1. Setup

MiniWoB++ benchmark suite. The miniwob++ task suite is selected as the main benchmark to evaluate our computer agent. MiniWoB++ (Liu et al., 2018), an extension of MiniWoB (Shi et al., 2017), is a web-based simulation environment that offers a diverse range of computer tasks, from simple button-clicking to complex compositional tasks requiring advanced reasoning, such as solving math problems. Its shared action space, including keyboard and mouse, and a common state space centered around HTML code enables our proposed agent to be thoroughly evaluated in ample tasks. Additionally, the varying levels of complexity between tasks enable a systematic evaluation of our work. The action space consists of two operations each of wich controls the keyboard and mouse. The first action enables typing of arbitrary characters or special keys such as Backspace and Enter. The second action involves moving and clicking the mouse, allowing the agent to interact with visible HTML elements on a webpage. All actions can be executed through natural language instructions defined by regular expressions that are presented within the initial prompts provided to the LLMs. The regular expressions employed in our evaluation are presented in Appendix D. Our action space definition is similar to previous works, such as (Gur et al., 2019; Jia et al., 2019; Liu et al., 2018), in which clicking actions directly interact with HTML elements. However, for typing actions, we extend beyond simple form-filling by using keyboard-based actions and excluding dictionary-based typing actions (Humphreys et al., 2022). Our approach, therefore, has a better generalization capability for diverse computer tasks. The state space of our agent consists solely of HTML code.

Model choices. For the purpose of evaluating the effectiveness of RCI prompting, multiple language models are used in our experiments. Specifically, we employ three models, namely, GPT-3 (*davinci*) (Brown et al., 2020), InstructGPT-3 (*text-davinci-002*) (Ouyang et al., 2022; Wei et al., 2022a;

Sanh et al., 2022), and InstructGPT-3 + RLHF (*text-davinci-003*, *gpt-3.5-turbo*, *gpt-4*) (Ouyang et al., 2022). Unless otherwise specified, we primarily evaluate our computer agent with the InstructGPT-3 + RLHF model. Additionally, we use GPT-3 and InstructGPT-3 models for ablation studies. Furthermore, the InstructGPT-3 + RLHF model is employed to carry out diverse reasoning tasks. All the models were obtained through the OpenAI API, and further details can be found in Appendix C.1.

Evaluated tasks. We employ a set of 55 tasks to enable fair comparisons with baselines, as previous works are only evaluated on a subset of tasks consistently. Furthermore, to assess the performance of models on challenging tasks, we have selected tasks that involve free-form language typing actions, which have been reported to have an almost-zero success rate in previous works (*e.g.,* terminal). Notably, certain commonly evaluated tasks in prior works are excluded due to the excessive length of HTML code for some UI components, which are described in Appendix C.3.

Metrics Consistent with prior studies, our main evaluation criterion is the success rate, which measures the ability of our agent to actually complete the assigned task. This rate is calculated as the proportion of successful episodes, which are defined as those in which the agent receives a positive reward. We identified two modes of failure: the production of unexecutable actions and task failure. When the agent generates an unexecutable action following the implicit RCI step, it fails immediately. Moreover, an episode is considered unsuccessful when the agent, despite effectively executing the plan generated, is unable to accomplish the task and thus receives no reward.

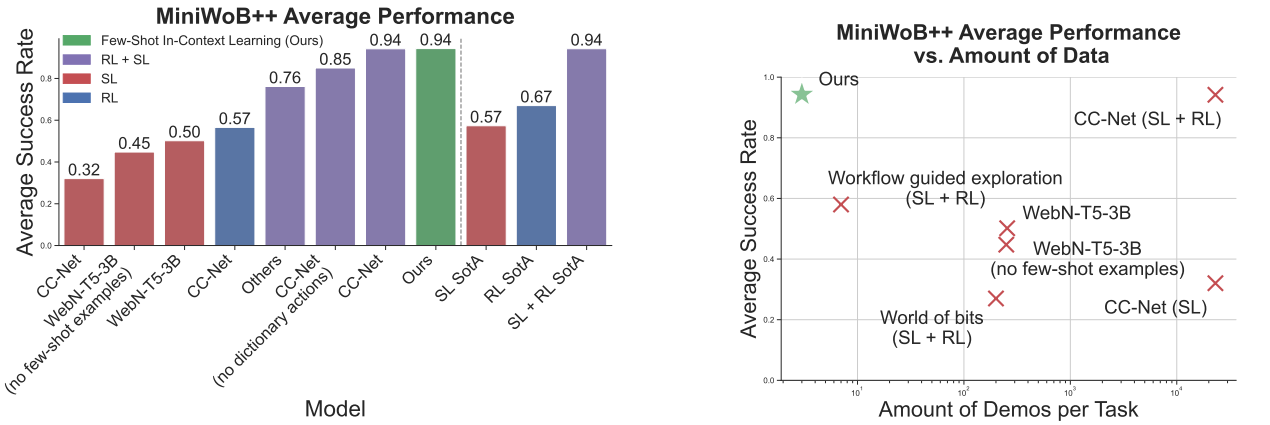
3.2.2. Outperforming baselines on MiniWoB++ task suite

We present Figure 4(a) which summarizes the average success rate of our agent and baseline models over the MiniWoB++ benchmark. The results demonstrate significant outperformance of our approach over supervised learning models. Specifically, we observe a 41% higher score than the *WebN-T5-3B*, which employs a finetuned large language model with 12K expert demonstration data. Our approach also outperforms reinforcement learning approaches that require an order of magnitude more interactions with the environment. Among all the baselines, our approach achieves the second highest score. The sole model that surpasses our agent is the *CC-Net*, which involves co-training of reinforcement learning and imitation learning. However, a direct comparison with *CC-Net* is not possible since it uses dictionary-based typing actions. In other words, *CC-Net* selects text from a predefined list for typing actions in some tasks, while our approach is fully generative. Thus, *CC-Net (without dictionary-based action)* in Figure 4(a) serves as our appropriate comparison and we outperform it by 6%. The performance data for *CC-Net (with no dictionary-based action)* is obtained from the ablation study section in their paper (Humphreys et al., 2022).

Another comparative analysis is performed to evaluate the performance of our agent in contrast to the state-of-the-art agents in three categories, namely supervised learning, reinforcement learning, and a combination of both. To facilitate a fair comparison, we specifically isolate LLM-based state-of-the-art approaches, which share similarities with our approach to solving computer tasks. The best performance achieved by each category is then aggregated, and the outcomes are presented as SotA in Figure 4(a). Our findings reveal that our agent surpasses SotA in supervised learning by 34% and in reinforcement learning by 24%. Notably, our proposed RCI prompting method outperforms the SotA LLM approach, even when the latter employs both finetuning and few-shot examples in prompts. This outcome highlights the effectiveness of our approach in extracting vital knowledge for computer tasks from language models. However, our agent underperforms in comparison to SotA in the combination of supervised and reinforcement learning, which employs significantly more expert data and online interactions. We also provide task-level performance comparisons in Figure 10, where tasks are arranged in ascending order based on the difference between our agent's performance and the baseline. We observed three main failure modes of our agent: (i) underperformance in tasks that require long-horizon planning (*e.g.,* guess-number, search-engine, use-spinner), (ii) difficulty in selecting appropriate actions for tasks that require multi-step reasoning (*e.g.,* tic-tac-toe, use-autocomplete), and (iii) lower scores in tasks that rely on visual rendering of HTML code to solve the task (*e.g.,* count-shape). These failures are explained in more detail in Appendix F.

3.2.3. Lowest sample complexity

Figure 4(b) provides a comparative analysis of the total number of samples used in several models and their mean performance. We begin by discussing *CC-Net* (Humphreys et al., 2022) model, which employs 2.4 million expert demonstrations (equivalent to 6,300 hours) collected from 77 human participants across 104 tasks for behavior cloning. This amounts to an average of 23,076 demonstrations per task. In contrast, the *WebN-T5-3B* (Gur et al., 2022) model uses 12,000 expert demonstrations to fine-tune its pre-trained T5 model. Rather than directly updating model parameters with demonstration data, our approach involves integrating two to three demonstrations into the prompt for in-context learning, which biases the model output without any parameter updates. This approach allows our agent to generalize to unseen tasks with only a handful of demonstrations. Our



(a) (b)

Figure 4: (a) Average performance comparison with baselines. Our agent with RCI prompting achieves state-of-the-art performance in MiniWoB++ environment. The tasks that were included in the averaging process are indicated in Table 17. (b) Relationship between performance and amount of expert training data. Our agent displays comparable performance to the current state-of-the-art scores on the MiniWoB++ benchmark, despite using the least amount of data.

results show that our agent achieved a higher success rate than all baselines, requiring 120x fewer samples than *WebN-T5-3B* and 11,000x fewer samples than *CC-Net*. Given the challenges of obtaining expert demonstrations for computer tasks, our findings demonstrate the practicality of our approach in automating such tasks.

3.2.4. Ablating the Groundings

This section examines the impact of grounding improvement on task success rates. We conduct ablations to isolate the contributions of task, state, and agent grounding improvements by eliminating RCI prompting at each stage. We categorize tasks by three different difficulty levels to provide a more detailed understanding of the effects of grounding improvements across a diverse range of tasks. We conducted a task grounding ablation by eliminating the plan sampling stage. This modification entails generating actions directly from the state, without the need for conditioning on a step-by-step plan. State grounding is evaluated by directly applying the agent-grounding update to task-grounded actions. Lastly, we ablate the implicit RCI of the agent grounding by letting the state-grounded action be the final output of the agent. Figure 5 illustrates the performance degradation resulting from each ablation of grounding. Our results indicate that each grounding contribution is essential to solving computer tasks, with each contributing almost equally to the overall success rate. The reason for this is partially due to the fact that the three methods of improving grounding are not mutually exclusive, but rather complementary, with one enhancement in grounding contributing to multiple action groundings. Examples of cross-grounding improvement are provided in Appendix E. Moreover, it has been observed that state grounding plays a crucial role in enabling an agent to use relevant information during episodes, particularly in scenarios where the initial state does not offer sufficient information to accomplish the task, such as *terminal* task. Interestingly, task grounding significantly improves the success rate when a task requires a long-horizon action plan, such as the *click checkboxes large* task. We also observe that agent grounding significantly enhances the feasibility of actions. Notably, in simpler tasks, the success rate decreases by 60% in contrast to the baseline without the agent grounding. This finding is of particular significance as it distinguishes our work from prior investigations (Ahn et al., 2022; Huang et al., 2022a), which employ additional trained model components. In contrast, our study solely relies on the reasoning ability of language models.

3.2.5. Ablating the Language Model

The performance of our agent is contingent on the quality of the underlying pre-trained language models used, so enhancing language models can lead to an improvement in the agent's performance. In this section, we present a comparison of the agent's performance using three distinct language models: GPT-3, InstructGPT-3, and InstructGPT-3 + RLHF. Our objective is to investigate the relationship between LLMs' capability and their ability to solve MiniWoB++ tasks. The experimental setting employed in Section 3.2.4 is replicated in this study. Figure 6 depicts the average success rate of three language models on tasks of varying difficulty levels. Our results reveal that LLMs struggle to effectively complete tasks without instruction fine-tuning. This may be attributed to the absence of intricate prompt engineering, as our observations have indicated that GPT-3 displays suffi-

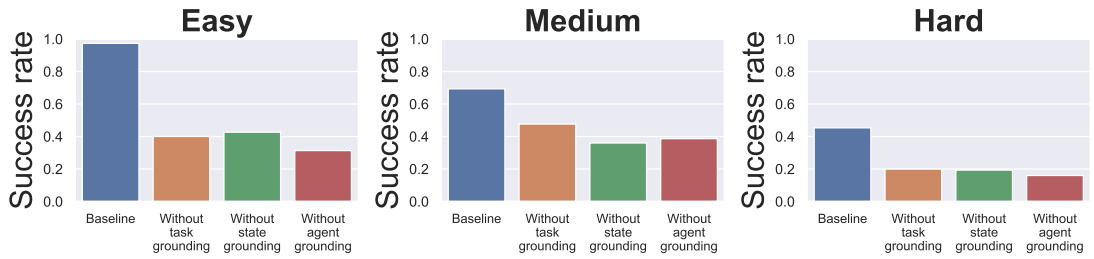


Figure 5: Ablation analysis on the different types of grounding across tasks with varying degrees of difficulty. The experimental design employs the use of *InstructGPT-3 + RLHF* model.

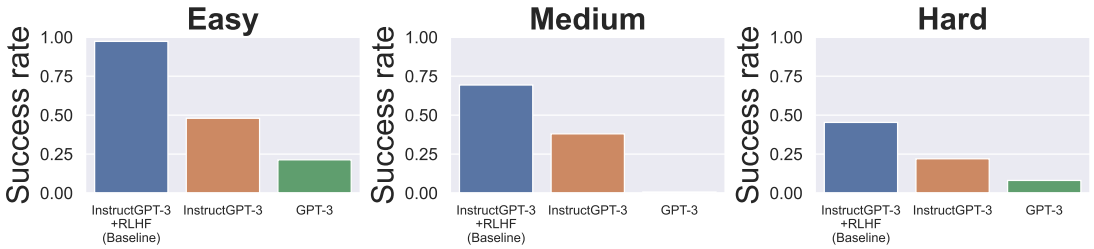


Figure 6: Ablation analysis on the different language models across tasks of varying degrees of difficulty.

cient competence in comprehending HTML code, regular expressions, and engaging in reasoning.

4. Discussion

This work is part of a growing literature showing that LLMs might be all you need for hard decision-making problems (Yang et al., 2023). In contrast to imitation learning and reinforcement learning approaches, LLMs can solve novel tasks in a zero-shot or few-shot manner, and don't require task-dependent expert data or a reward function. Furthermore, we expect that as the capabilities of LLMs and foundation models increase, our method will naturally improve as well. However, we find that current capabilities of LLMs aren't as powerful as task-dependent SL+RL approaches on some computer tasks. Also, RCI is more expensive to run compared to approaches that just sample once from the LLM. There are many avenues for future research in increasing the capacity of LLMs in decision-making tasks. First, our experiments use LLMs on HTML code, but ideally methods based on multimodal foundation models (Driess et al., 2023; Reed et al., 2022; Alayrac et al., 2022; OpenAI, 2023) will be able to take actions based on text, images, audio, and video as input (Baker et al., 2022; Fan et al., 2022; Nottingham et al., 2023; Wang et al., 2023b). Second, the results presented in this paper all use pre-trained LLMs. We expect the performance of our method to increase when using LLMs fine-tuned to solve computer tasks.

Importantly, current LLMs are poor at reasoning tasks, such as playing tic-tac-toe, because they do not think ahead. Although RCI improves reasoning capabilities in LLMs, there exist much work to be done on increasing the reasoning capabilities in LLMs. This will be crucial to accomplish hard cognitive tasks on computers that require thinking ahead. Similar to other prompting-based approaches for reasoning in LLMs, RCI can be viewed as using the LLM's output to write to an external memory, which is later retrieved to choose an action. LLMs with memory have been demonstrated to be computationally universal (Schuurmans, 2023), meaning that in principle all that is needed to run arbitrary programs is the right prompt. Since RCI represents a basic version of this powerful framework, we anticipate the development of more advanced RCI variations in the future. There is a vast array of potential methods that repeatedly feed the output of particular prompts into the LLM. For example, multiple different LLMs can simulate the information exchange between team members in an organization. This would enable the merging of diverse perspectives to tackle complex problems. In such a context, incorporating game theory and multi-agent systems research could significantly enhance the overall performance. Reinforcement learning could be used to discover effective structures involving loops and prompts (Zhang et al., 2023), either through human feedback or a given reward function. This optimization process can be further refined by exploring the space of potential loop and prompt structures, identifying those that yield the best results, and fine-tuning the model accordingly (Yang et al., 2022).

References

Ahn, M., Brohan, A., Brown, N., Chebotar, Y., Cortes, O., David, B., Finn, C., Gopalakrishnan, K., Hausman, K., Herzog, A., et al. Do as I can, not as I say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022.

Alayrac, J.-B., Donahue, J., Luc, P., Miech, A., Barr, I., Hasson, Y., Lenc, K., Mensch, A., Millican, K., Reynolds, M., et al. Flamingo: a visual language model for few-shot learning. *Advances in Neural Information Processing Systems*, 35:23716–23736, 2022.

Bai, Y., Kadavath, S., Kundu, S., Askell, A., Kernion, J., Jones, A., Chen, A., Goldie, A., Mirhoseini, A., McKinnon, C., et al. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*, 2022.

Baker, B., Akkaya, I., Zhokov, P., Huizinga, J., Tang, J., Ecoffet, A., Houghton, B., Sampedro, R., and Clune, J. Video pretraining (vpt): Learning to act by watching unlabeled online videos. *Advances in Neural Information Processing Systems*, 35:24639–24654, 2022.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33: 1877–1901, 2020.

Bubeck, S., Chandrasekaran, V., Eldan, R., Gehrke, J., Horvitz, E., Kamar, E., Lee, P., Lee, Y. T., Li, Y., Lundberg, S., et al. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*, 2023.

Carta, T., Romac, C., Wolf, T., Lamprier, S., Sigaud, O., and Oudeyer, P.-Y. Grounding large language models in interactive environments with online reinforcement learning. *arXiv preprint arXiv:2302.02662*, 2023.

Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H. W., Sutton, C., Gehrmann, S., et al. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*, 2022.

Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Creswell, A. and Shanahan, M. Faithful reasoning using large language models. *arXiv preprint arXiv:2208.14271*, 2022.

Creswell, A., Shanahan, M., and Higgins, I. Selection-inference: Exploiting large language models for interpretable logical reasoning. *arXiv preprint arXiv:2205.09712*, 2022.

Dai, D., Sun, Y., Dong, L., Hao, Y., Sui, Z., and Wei, F. Why can GPT learn in-context? Language models secretly perform gradient descent as meta optimizers. *arXiv preprint arXiv:2212.10559*, 2022.

Dasgupta, I., Kaeser-Chen, C., Marino, K., Ahuja, A., Babayan, S., Hill, F., and Fergus, R. Collaborating with language models for embodied reasoning. In *Second Workshop on Language and Reinforcement Learning*, 2022.

Dohan, D., Xu, W., Lewkowycz, A., Austin, J., Bieber, D., Lopes, R. G., Wu, Y., Michalewski, H., Saurous, R. A., Sohl-Dickstein, J., et al. Language model cascades. *arXiv preprint arXiv:2207.10342*, 2022.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. An image is worth 16x16 words: Transformers for image recognition at scale. *International Conference on Learning Representations*, 2020.

Driess, D., Xia, F., Sajjadi, M. S., Lynch, C., Chowdhery, A., Ichter, B., Wahid, A., Tompson, J., Vuong, Q., Yu, T., et al. PaLM-E: An embodied multimodal language model. *arXiv preprint arXiv:2303.03378*, 2023.

Du, N., Huang, Y., Dai, A. M., Tong, S., Lepikhin, D., Xu, Y., Krikun, M., Zhou, Y., Yu, A. W., Firat, O., et al. GLaM: Efficient scaling of language models with mixture-of-experts. In *International Conference on Machine Learning*, pp. 5547–5569. PMLR, 2022.

Fan, L., Wang, G., Jiang, Y., Mandlekar, A., Yang, Y., Zhu, H., Tang, A., Huang, D.-A., Zhu, Y., and Anandkumar, A. Minedojo: Building open-ended embodied agents with internet-scale knowledge. In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2022.

Furuta, H., Nachum, O., Lee, K.-H., Matsuo, Y., Gu, S. S., and Gur, I. Instruction-finetuned foundation models for multimodal web navigation. In *Workshop on Reincarnating Reinforcement Learning at ICLR*, 2023.

Ganguli, D., Askell, A., Schiefer, N., Liao, T., Lukošiūtė, K., Chen, A., Goldie, A., Mirhoseini, A., Olsson, C., Hernandez, D., et al. The capacity for moral self-correction in large language models. *arXiv preprint arXiv:2302.07459*, 2023.

Geva, M., Khashabi, D., Segal, E., Khot, T., Roth, D., and Berant, J. Did aristotle use a laptop? a question answering

benchmark with implicit reasoning strategies. *Transactions of the Association for Computational Linguistics*, 9: 346–361, 2021.

Glaese, A., McAleese, N., Trębacz, M., Aslanides, J., Firoiu, V., Ewalds, T., Rauh, M., Weidinger, L., Chadwick, M., Thacker, P., et al. Improving alignment of dialogue agents via targeted human judgements. *arXiv preprint arXiv:2209.14375*, 2022.

Gur, I., Rueckert, U., Faust, A., and Hakkani-Tur, D. Learning to navigate the web. In *International Conference on Learning Representations*, 2019.

Gur, I., Jaques, N., Miao, Y., Choi, J., Tiwari, M., Lee, H., and Faust, A. Environment generation for zero-shot compositional reinforcement learning. *Advances in Neural Information Processing Systems*, 34:4157–4169, 2021.

Gur, I., Nachum, O., Miao, Y., Safdari, M., Huang, A., Chowdhery, A., Narang, S., Fiedel, N., and Faust, A. Understanding HTML with large language models. *arXiv preprint arXiv:2210.03945*, 2022.

Hoffmann, J., Borgeaud, S., Mensch, A., Buchatskaya, E., Cai, T., Rutherford, E., de Las Casas, D., Hendricks, L. A., Welbl, J., Clark, A., et al. An empirical analysis of compute-optimal large language model training. *Advances in Neural Information Processing Systems*, 35: 30016–30030, 2022.

Hosseini, M. J., Hajishirzi, H., Etzioni, O., and Kushman, N. Learning to solve arithmetic word problems with verb categorization. In *EMNLP*, pp. 523–533, 2014.

Huang, W., Abbeel, P., Pathak, D., and Mordatch, I. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In *International Conference on Machine Learning*, pp. 9118–9147. PMLR, 2022a.

Huang, W., Xia, F., Xiao, T., Chan, H., Liang, J., Florence, P., Zeng, A., Tompson, J., Mordatch, I., Chebotar, Y., et al. Inner monologue: Embodied reasoning through planning with language models. In *6th Annual Conference on Robot Learning*, 2022b.

Humphreys, P. C., Raposo, D., Pohlen, T., Thornton, G., Chhaparia, R., Muldal, A., Abramson, J., Georgiev, P., Santoro, A., and Lillicrap, T. A data-driven approach for learning to control computers. In *International Conference on Machine Learning*, pp. 9466–9482. PMLR, 2022.

Iki, T. and Aizawa, A. Do BERTs learn to use browser user interface? Exploring multi-step tasks with unified vision-and-language berts. *arXiv preprint arXiv:2203.07828*, 2022.

Jia, S., Kiros, J. R., and Ba, J. DOM-Q-NET: Grounded RL on structured language. In *International Conference on Learning Representations*, 2019.

Kojima, T., Gu, S. S., Reid, M., Matsuo, Y., and Iwasawa, Y. Large language models are zero-shot reasoners. In *Advances in Neural Information Processing Systems*, 2022.

Koncel-Kedziorski, R., Hajishirzi, H., Sabharwal, A., Etzioni, O., and Ang, S. D. Parsing algebraic word problems into equations. *Transactions of the Association for Computational Linguistics*, 3:585–597, 2015.

Ling, W., Yogatama, D., Dyer, C., and Blunsom, P. Program induction by rationale generation: Learning to solve and explain algebraic word problems. *Proceedings of ACL*, 2017.

Liu, E. Z., Guu, K., Pasupat, P., Shi, T., and Liang, P. Reinforcement learning on web interfaces using workflow-guided exploration. In *International Conference on Learning Representations*, 2018.

Liu, R., Wei, J., Gu, S. S., Wu, T.-Y., Vosoughi, S., Cui, C., Zhou, D., and Dai, A. M. Mind's eye: Grounded language model reasoning through simulation. In *International Conference on Learning Representations*, 2023.

Madaan, A. and Yazdanbakhsh, A. Text and patterns: For effective chain of thought, it takes two to tango. *arXiv preprint arXiv:2209.07686*, 2022.

Madaan, A., Tandon, N., Gupta, P., Hallinan, S., Gao, L., Wiegreffe, S., Alon, U., Dziri, N., Prabhumoye, S., Yang, Y., et al. Self-refine: Iterative refinement with self-feedback. *arXiv preprint arXiv:2303.17651*, 2023.

Menick, J., Trebacz, M., Mikulik, V., Aslanides, J., Song, F., Chadwick, M., Glaese, M., Young, S., Campbell-Gillingham, L., Irving, G., et al. Teaching language models to support answers with verified quotes. *arXiv preprint arXiv:2203.11147*, 2022.

Mialon, G., Dessì, R., Lomeli, M., Nalmpantis, C., Pasunuru, R., Raileanu, R., Rozière, B., Schick, T., Dwivedi-Yu, J., Celikyilmaz, A., et al. Augmented language models: a survey. *arXiv preprint arXiv:2302.07842*, 2023.

Nakano, R., Hilton, J., Balaji, S., Wu, J., Ouyang, L., Kim, C., Hesse, C., Jain, S., Kosaraju, V., Saunders, W., et al. WebGPT: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*, 2021.

Nogueira, R. and Cho, K. End-to-end goal-driven web navigation. *Advances in Neural Information Processing Systems*, 29, 2016.

Nottingham, K., Ammanabrolu, P., Suhr, A., Choi, Y., Hajishirzi, H., Singh, S., and Fox, R. Do embodied agents dream of pixelated sheep?: Embodied decision making using language guided world modelling. *arXiv preprint arXiv:2301.12050*, 2023.

Nye, M., Andreassen, A. J., Gur-Ari, G., Michalewski, H., Austin, J., Bieber, D., Dohan, D., Lewkowycz, A., Bosma, M., Luan, D., et al. Show your work: Scratchpads for intermediate computation with language models. In *Deep Learning for Code Workshop at ICLR*, 2022.

OpenAI. Gpt-4 technical report, 2023.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744, 2022.

Paranjape, B., Lundberg, S., Singh, S., Hajishirzi, H., Zettlemoyer, L., and Ribeiro, M. T. ART: Automatic multi-step reasoning and tool-use for large language models. *arXiv preprint arXiv:2303.09014*, 2023.

Pasupat, P. and Liang, P. Zero-shot entity extraction from web pages. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 391–401, 2014.

Pasupat, P., Jiang, T.-S., Liu, E., Guu, K., and Liang, P. Mapping natural language commands to web elements. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 4970–4976, 2018.

Patel, A., Bhattamishra, S., and Goyal, N. Are nlp models really able to solve simple math word problems? *Proceedings of NAACL*, 2021.

Press, O., Zhang, M., Min, S., Schmidt, L., Smith, N. A., and Lewis, M. Measuring and narrowing the compositionality gap in language models. *arXiv preprint arXiv:2210.03350*, 2022a.

Press, O., Zhang, M., Min, S., Schmidt, L., Smith, N. A., and Lewis, M. Measuring and narrowing the compositionality gap in language models. *arXiv preprint arXiv:2210.03350*, 2022b.

Raman, S. S., Cohen, V., Rosen, E., Idrees, I., Paulius, D., and Tellex, S. Planning with large language models via corrective re-prompting. *Foundation Models for Decision Making workshop at NeurIPS*, 2022.

Reed, S., Zolna, K., Parisotto, E., Colmenarejo, S. G., Novikov, A., Barth-maron, G., Giménez, M., Sulsky, Y., Kay, J., Springenberg, J. T., et al. A generalist agent. *Transactions on Machine Learning Research*, 2022.

Roy, S. and Roth, D. Solving general arithmetic word problems. *EMNLP*, 2016.

Sanh, V., Webson, A., Raffel, C., Bach, S., Sutawika, L., Alyafeai, Z., Chaffin, A., Stiegler, A., Raja, A., Dey, M., et al. Multitask prompted training enables zero-shot task generalization. In *International Conference on Learning Representations*, 2022.

Saunders, W., Yeh, C., Wu, J., Bills, S., Ouyang, L., Ward, J., and Leike, J. Self-critiquing models for assisting human evaluators. *arXiv preprint arXiv:2206.05802*, 2022.

Schick, T., Dwivedi-Yu, J., Dessì, R., Raileanu, R., Lomeli, M., Zettlemoyer, L., Cancedda, N., and Scialom, T. Toolformer: Language models can teach themselves to use tools. *arXiv preprint arXiv:2302.04761*, 2023.

Schuurmans, D. Memory augmented large language models are computationally universal. *arXiv preprint arXiv:2301.04589*, 2023.

Shi, T., Karpathy, A., Fan, L., Hernandez, J., and Liang, P. World of bits: An open-domain platform for web-based agents. In *International Conference on Machine Learning*, pp. 3135–3144. PMLR, 2017.

Shinn, N., Labash, B., and Gopinath, A. Reflexion: an autonomous agent with dynamic memory and self-reflection. *arXiv preprint arXiv:2303.11366*, 2023.

Shridhar, M., Manuelli, L., and Fox, D. CLIPort: What and where pathways for robotic manipulation. In *Conference on Robot Learning*, pp. 894–906. PMLR, 2022.

Smith, S., Patwary, M., Norick, B., LeGresley, P., Rajbhandari, S., Casper, J., Liu, Z., Prabhumoye, S., Zerveas, G., Korthikanti, V., et al. Using deepspeed and megatron to train megatron-turing NLG 530b, a large-scale generative language model. *arXiv preprint arXiv:2201.11990*, 2022.

Srivastava, S., Polozov, O., Jojic, N., and Meek, C. Learning web-based procedures by reasoning over explanations and demonstrations in context. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 7652–7662, 2020.

Sun, Z., Wang, X., Tay, Y., Yang, Y., and Zhou, D. Recitation-augmented language models. In *International Conference on Learning Representations*, 2023.

Talmor, A., Herzig, J., Lourie, N., and Berant, J. Commonsenseqa: A question answering challenge targeting commonsense knowledge. *Proceedings of NAACL-HLT*, 2019.

Thoppilan, R., De Freitas, D., Hall, J., Shazeer, N., Kulshreshtha, A., Cheng, H.-T., Jin, A., Bos, T., Baker, L.,

Du, Y., et al. LaMDA: Language models for dialog applications. *arXiv preprint arXiv:2201.08239*, 2022.

von Oswald, J., Niklasson, E., Randazzo, E., Sacramento, J., Mordvintsev, A., Zhmoginov, A., and Vladymyrov, M. Transformers learn in-context by gradient descent. *arXiv preprint arXiv:2212.07677*, 2022.

Wang, X., Wei, J., Schuurmans, D., Le, Q., Chi, E., and Zhou, D. Self-consistency improves chain of thought reasoning in language models. In *International Conference on Learning Representations*, 2023a.

Wang, Z., Cai, S., Liu, A., Ma, X., and Liang, Y. Describe, explain, plan and select: Interactive planning with large language models enables open-world multi-task agents. *arXiv preprint arXiv:2302.01560*, 2023b.

Wei, J., Bosma, M., Zhao, V., Guu, K., Yu, A. W., Lester, B., Du, N., Dai, A. M., and Le, Q. V. Finetuned language models are zero-shot learners. In *International Conference on Learning Representations*, 2022a.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E. H., Le, Q. V., Zhou, D., et al. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, 2022b.

Welleck, S., Lu, X., West, P., Brahman, F., Shen, T., Khashabi, D., and Choi, Y. Generating sequences by learning to self-correct. *arXiv preprint arXiv:2211.00053*, 2022.

Yang, M. S., Schuurmans, D., Abbeel, P., and Nachum, O. Chain of thought imitation with procedure cloning. *Advances in Neural Information Processing Systems*, 35: 36366–36381, 2022.

Yang, S., Nachum, O., Du, Y., Wei, J., Abbeel, P., and Schuurmans, D. Foundation models for decision making: Problems, methods, and opportunities. *arXiv preprint arXiv:2303.04129*, 2023.

Yao, S., Chen, H., Yang, J., and Narasimhan, K. R. Webshop: Towards scalable real-world web interaction with grounded language agents. In *Advances in Neural Information Processing Systems*, 2022.

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., and Cao, Y. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023.

Zelikman, E., Wu, Y., Mu, J., and Goodman, N. STaR: Bootstrapping reasoning with reasoning. *Advances in Neural Information Processing Systems*, 35:15476–15488, 2022.

Zeng, A., Wong, A., Welker, S., Choromanski, K., Tombari, F., Purohit, A., Ryoo, M., Sindhwani, V., Lee, J., Vanhoucke, V., et al. Socratic models: Composing zero-shot multimodal reasoning with language. In *International Conference on Learning Representations*, 2023.

Zhang, T., Wang, X., Zhou, D., Schuurmans, D., and Gonzalez, J. E. TEMPERA: Test-time prompt editing via reinforcement learning. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=gSHyqBijPFO.

Zhou, D., Schärli, N., Hou, L., Wei, J., Scales, N., Wang, X., Schuurmans, D., Cui, C., Bousquet, O., Le, Q. V., and Chi, E. H. Least-to-most prompting enables complex reasoning in large language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=WZH7099tgfM.