

RAT: Retrieval Augmented Thoughts Elicit Context-Aware Reasoning in Long-Horizon Generation

Zihao Wang¹, Anji Liu², Haowei Lin¹, Jiaqi Li³, Xiaojian Ma³ and Yitao Liang¹

¹Peking University, ²University of California, Los Angeles, ³Beijing Institute for General Artificial Intelligence

We explore how iterative revising a chain of thoughts with the help of information retrieval significantly improves large language models' reasoning and generation ability in long-horizon generation tasks, while hugely mitigating hallucination. In particular, the proposed method — *retrieval-augmented thoughts* (RAT) — revises each thought step one by one with retrieved information relevant to the task query, the current and the past thought steps, after the initial zero-shot CoT is generated. Applying RAT to GPT-3.5, GPT-4, and CodeLLaMA-7b substantially improves their performances on various long-horizon generation tasks; on average of relatively increasing rating scores by 13.63% on code generation, 16.96% on mathematical reasoning, 19.2% on creative writing, and 42.78% on embodied task planning. The demo page can be found in <https://craftjarvis.github.io/RAT>.

1. Introduction

Large Language Models (LLMs) have achieved fruitful progress on various natural language reasoning tasks (Brown et al., 2020; Wang et al., 2023a; Wei et al., 2022; Yao et al., 2022; Zhou et al., 2023), especially when combining large-scale models (OpenAI, 2023; Team, 2022) with sophisticated prompting strategies, notably chain-of-thought (CoT) prompting (Kojima et al., 2022; Wei et al., 2022). However, there have been increasing concerns about the factual correctness of LLMs reasoning, citing the possible hallucinations in model responses (Rawte et al., 2023) or the intermediate reasoning paths, *i.e.* CoTs (Dhuliawala et al., 2023). This issue becomes more significant when it comes to zero-shot CoT prompting, aka. “let’s think step-by-step” (Kojima et al., 2022) and long-horizon generation tasks that require multi-step and context-aware reasoning, including code generation, task planning, mathematical reasoning, *etc.* Factually valid intermediate thoughts could be critical to the successful completion of these tasks.

Several prompting techniques have been proposed to mitigate this issue, one promising direction, Retrieval Augmented Generation (RAG) (Lewis et al., 2020b) seeks insights from human reasoning (Holyoak and Morrison, 2012),

and utilizes retrieved information to facilitate more factually grounded reasoning. In this paper, we explore how to synergize RAG with sophisticated long-horizon reasoning. Our intuition is that the hallucination within the intermediate reasoning process could be alleviated through the help of outside knowledge. The resulting prompting strategy, *retrieval-augmented thoughts* (RAT), is illustrated in Figure 1. Our strategy comprises two key ideas. Firstly, the initial zero-shot CoT produced by LLMs along with the original task prompt will be used as queries to retrieve the information that could help revise the possibly flawed CoT. Secondly, instead of retrieving and revising with the full CoT and producing the final response at once, we devise a progressive approach, where LLMs produce the response step-by-step following the CoT (a series of sub-tasks), and only the current thought step will be revised based on the information retrieved with task prompt, the current and the past CoTs. This strategy can be an analogy to the human reasoning process: we utilize outside knowledge to adjust our step-by-step thinking during complex long-horizon problem-solving (Holyoak and Morrison, 2012). A comparison of RAT and counterparts can be found in Figure 2.

We evaluate RAT on a wide collection of challenging long-horizon tasks, including code gen-

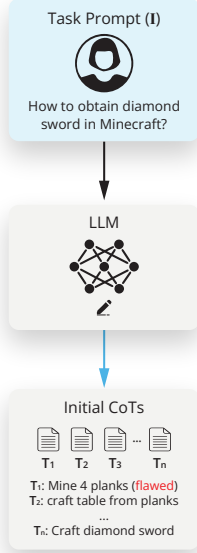
Step 0

Draft initial step-by-step zero-shot CoTs based on the task prompt.

A task prompt is given by a human user.

LLM makes zero-shot step-by-step reasoning based on the prompt.

This initial zero-shot CoT answer may be flawed.

Step 1 - Step n

Retrieve relevant information and iteratively revise each CoT with all previous generations in context.

Retrieve with the task prompt and previous generated CoTs.

LLM revises the i -th steps in thought chains ($T_{1:i-1}^*$, T_i) based on the retrieved content.

The thought chain ($T_{1:i-1}^*$, T_i) is replaced with the revised generation $T_{1:i}^*$.

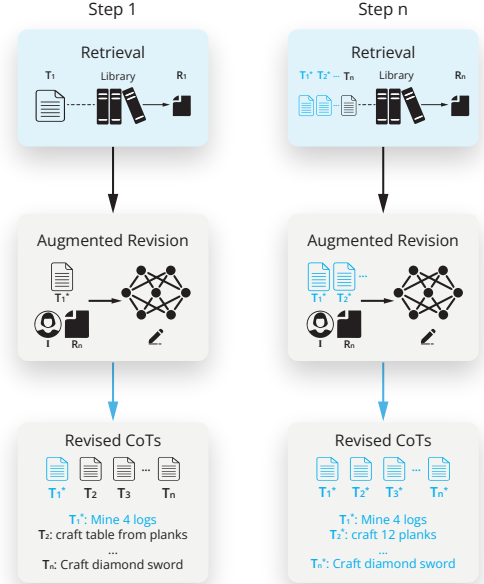


Figure 1 | Pipeline of Retrieval Augmented Thoughts (RAT). Given a task prompt (denoted as I in the figure), RAT starts from initial step-by-step thoughts (T_1, T_2, \dots, T_n) produced by an LLM in zero-shot (“let’s think step by step”). Some thought steps (such as T_1 in the figure) may be **flawed** due to hallucination. RAT iteratively revises each thought step ($T_1^*, T_2^*, \dots, T_{i-1}^*, T_i$) using RAG from an external knowledge base (denoted as *Library*). Detailed prompting strategy can be found in subsection 2.2.

eration, mathematical reasoning, embodied task planning, and creative writing. We employ several LLMs of varied scales: GPT-3.5 (Brown et al., 2020), GPT-4 (OpenAI, 2023), CodeLLaMA-7b (Rozière et al., 2023). The results indicate that combining RAT with these LLMs elicits strong advantages over vanilla CoT prompting and RAG approaches. In particular, we observe new state-of-the-art level of performances across our selection of tasks: 1) code generation: HumanEval (+20.94%), HumanEval+ (+18.89%), MBPP (+14.83%), MBPP+ (+1.86%); 2) mathematical reasoning problems: GSM8K (+8.36%), and GSMHard (+31.37%); 3) Minecraft task planning (2.96 times on executability and +51.94% on plausibility); 4) creative writing (+19.19% on human score). Our additional ablation studies further confirm the crucial roles played by the two key ingredients of RAT: revising CoT using RAG and progressive revision & generation. This work reveals how can LLMs revise their reasoning process in a zero-shot fashion with the help of outside knowledge, just as what humans do.

2. Retrieval Augmented Thoughts

Our goal is to support long-horizon reasoning and generation while mitigating hallucination when using LLMs. To have satisfying performance on long-horizon tasks, two ingredients are indispensable. Firstly, access to factual information can be facilitated by retrieval. Secondly, appropriate intermediate steps that outline a scratchpad to finish complex tasks, can be facilitated by CoT. Yet, a naive combination of the two would not necessarily yield improvements. Two questions still persist: (1) what is relevant information to retrieve; (2) how to effectively correct reasoning steps with relevant factual information. To better appreciate our method and why our method can address these two questions, we first provide a brief preliminary introduction of RAG and CoT.

2.1. Preliminary

Retrieval-Augmented Generation (RAG) targets the problem of generating fictitious facts by pro-

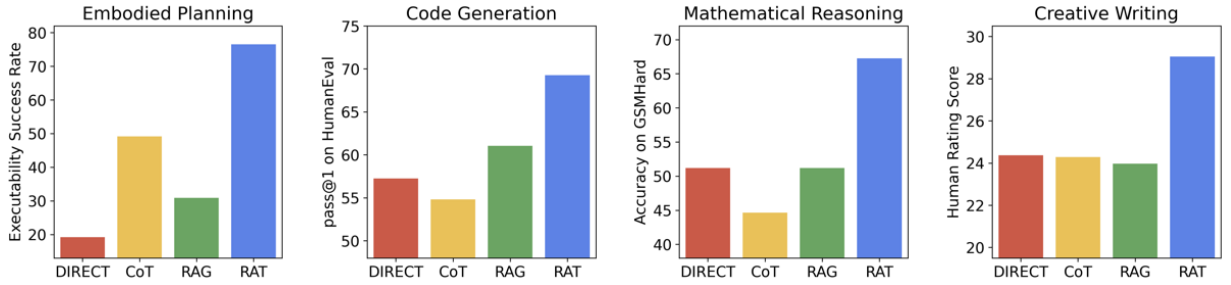
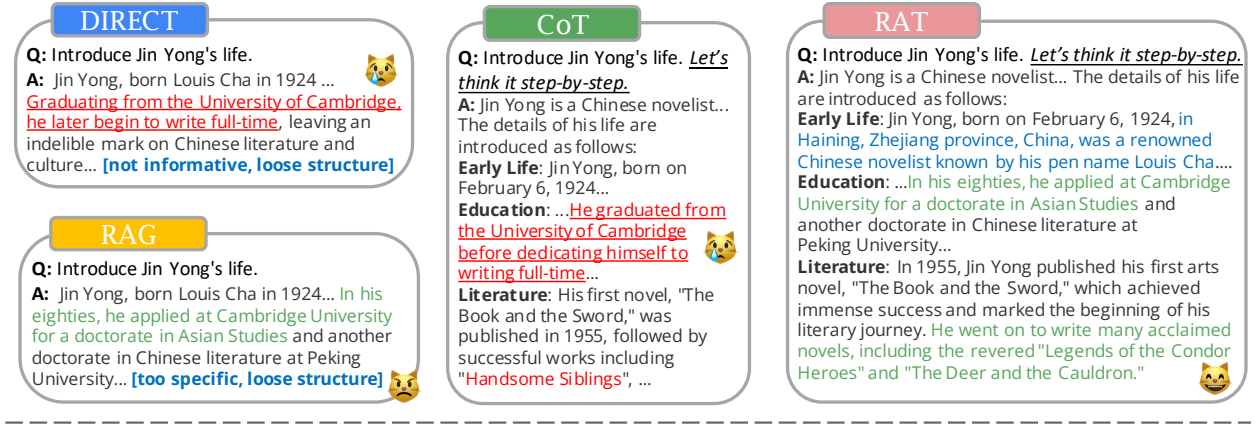


Figure 2 | **Top:** An example of different LLM reasoning methods on creative generation tasks. Red text indicates errors or illusions in the text generated by LLM, while green text represents correct generation. Methods without RAG often generate incorrect information with hallucination, classical RAG is highly related to retrieved content with a loose structure, and RAT-generated texts perform best in terms of accuracy and completeness. **Bottom:** The quantitative performance comparison for different LLM reasoning methods on complex embodied planning, mathematical reasoning, code generation, and creative generation tasks. Our RAT outperforms all the baselines on all tasks.

viding LLMs with relevant text extracted from trusted sources. It is primarily used in question-answering (QA) tasks (Lewis et al., 2020b). Specifically, given a set of n candidate documents $R := \{R_i\}_{i=1}^n$, RAG aims to retrieve the most relevant ones w.r.t. a query Q , which can be the question/task prompt itself or relevant information generated by LLMs. To achieve this, RAG first extracts semantic-aware embeddings of the documents $r_i := \text{emb}(R_i) \in \mathbb{R}^K$ (K is the size of the embedding) as well as the query $q := \text{emb}(Q) \in \mathbb{R}^K$. $\text{emb}(\cdot)$ can be implemented with various text embedding models, such as Sentence-BERT (Reimers and Gurevych, 2019). The relevance between the query and a document is measured by their cosine similarity:

$$\text{sim}(Q, R_i) = \frac{q \cdot r_i}{\|q\| \|r_i\|}.$$

Based on their relevance, the top-ranked k documents are then fed into the prompt for LLMs to generate the final answer. With such rich and

factual contexts, RAG mitigates the hallucination of LLMs. However, complex reasoning tasks (e.g., those requiring multi-step reasoning) can be difficult to translate into effective search queries, leading to challenges in finding relevant documents and making RAG less applicable. Traditionally, RAG retrieves all relevant information at once. Yet, it overlooks the fact that it is difficult to predict what "facts" or information is required in the subsequent reasoning and generation steps. The task prompt itself is hardly sufficient to provide enough clues for this.

Chain of Thoughts (CoT) prompting is designed to enhance the performance of LLMs under tasks that require complex reasoning steps (Wei et al., 2022), such as multi-step math word problems. Specifically, instead of tasking LLMs to generate the correct answer directly, CoT prompting incentivizes LLMs to first output intermediate reasoning steps, termed thoughts, that serve as a scratch space for the task, before summarizing

Algorithm 1 Retrieval augmented thoughts (RAT)**Input:** Task Prompt I , Autoregressive Large Language Model p_θ

```

1:  $T = \{T_1, T_2, \dots, T_n\} \leftarrow p_\theta(\cdot|I)$            ▶ Generate zero-shot initial step-by-step thoughts  $T$ 
2:  $T^* \leftarrow T_1, i \leftarrow 1$                    ▶ Draft answer  $T^*$  initialized with the first thought step  $T_1$ 
3: repeat
4:    $Q_i \leftarrow \text{ToQuery}(I, T^*)$                  ▶ Generate query  $Q_i$  based on current draft answer  $T^*$ 
5:    $R_i \leftarrow \text{RetrieveFromCorpus}(Q_i)$            ▶ Retrieve information  $R_i$  from corpus or Internet
6:    $T^* \leftarrow p_\theta(\cdot|I, T^*, R_i)$                ▶ Revise draft answer  $T^*$  based on retrieved text  $R_i$ 
7:    $T^* \leftarrow \text{CONCAT}(T^*, T_{i+1})$              ▶ Append the next thought step  $T_{i+1}$ 
8:    $i \leftarrow i + 1$                                ▶ Begin the next revision round
9: until  $i > n$                                        ▶ Repeat until all the revised thoughts  $T^*_{1:n}$  are obtained
10: return  $T^*$                                        ▶ Output  $T^*$  as the final generation

```

the thoughts into a final answer. Such behavior of LLMs can either be stimulated in *zero-shot* by prompting terms that encourage CoT reasoning (e.g., “let’s think step by step”) (Kojima et al., 2022), or triggered by few-shot examples that perform CoT in similar tasks. However, since no direct supervision is posed to the intermediate thoughts, LLMs could make errors due to the lack of relevant domain knowledge (Touvron et al., 2023) or biased by hallucinations (Rawte et al., 2023).

2.2. Our Approach

Our intuition to mitigate the issues of CoT prompting and RAG mentioned above is to apply RAG to revise every thought step generated by CoT prompting. An overview can be found in Figure 1 and Algorithm 1. Specifically, given a task prompt I , we first prompt LLM to generate step-by-step thoughts in *zero shot* (“let’s think step-by-step”) $T := \{T_i\}_{i=1}^n$, where T_i represents the i th thought step. In long-horizon generation tasks, T can either be the intermediate reasoning steps, e.g. the pseudo code with comments in code generation, article outline in creative writing, etc., or the draft response itself, e.g. a list of sub-goals in embodied task planning as shown in Figure 1.

Since T could be flawed (e.g., contains hallucination), we proceed to use RAG to revise every generated thought step before generating the final response from these thoughts. Specifically, assuming we have fixed the previous thought steps and now are about to revise $T_{1:i}$, we begin by

converting the text $\{I, T_1, \dots, T_i\}$ into a query Q_i :

$$Q_i = \text{ToQuery}(I, T_1, \dots, T_i),$$

where $\text{ToQuery}(\cdot)$ can either be a text encoder or an LLM that translates the task prompt I , the current and the past thought steps T_1, \dots, T_i into a query Q_i that can be processed by the retrieval system. We adopt RAG to retrieve relevant documents R_i using Q_i , which are then prepended to the prompt to generate a revised thought step T^*_i .

$$T^*_{1:i} = p_\theta(\cdot|I, T_1, \dots, T_i, R_i).$$

Finally, depending on the actual task, the revised thought steps $T^*_{1:n}$ can simply be used as the final model response, e.g., embodied task planning. For tasks like code generation, or creative writing, the LLM will be further prompted to produce the complete response (code, passage) from each revised thought step in a step-by-step fashion.

Note that, when revising the i -th thought step T_i , instead of using the current step T_i only, or the complete chain of thoughts T_1, \dots, T_n to produce the query for RAG, we ensure the query Q_i is produced from the current thought step T_i and previous revised thought steps $T^*_{1:i-1}$, i.e., we adopt a *casual reasoning* to revise the thoughts using RAG:

$$Q_i = \text{ToQuery}(I, T^*_{1:i-1}, T_i)$$

$$T^*_{1:i} = p_\theta(\cdot|I, T^*_{1:i-1}, T_i, R_i).$$

This allows for the correction of errors in the original thoughts T by continually consulting different reference texts and ensures that each step

of reasoning is informed by the most accurate and relevant information, significantly improving the quality and reliability of the generated output.

Our hypothesis why our method can address the two problems mentioned at the beginning of this section is as follows. Firstly, the most straightforward way to know what information will be used in complex reasoning is to “see” the reasoning steps. Our approach leverages all the generated thoughts along with the task prompt to provide more clues for more effective retrieval. Secondly, some information cannot be directly retrieved, especially information related to the final answer to a hard complex question. Instead, retrieval of information relevant to intermediate questions, which are assumed to be easier, is more accessible. Thanks to the compositional nature of many reasoning tasks, an iterative retrieval process could also be more effective. Thirdly, correcting potential hallucinations needs to be targeted. Revising a complete CoT with RAG could introduce errors at otherwise already-correct steps. Revising every step one by one could be more reliable. The first two points address question (1) and the last point addresses question (2). Quantitative evidence can be found in our ablation studies in subsection 3.4.

3. Experiments

We test our proposed method RAT on a diverse set of benchmarks that highlight long-horizon generation and reasoning. Existing methods traditionally struggle in those benchmarks; “hallucinated” steps are obvious in LLMs’ outputs. Those steps either fail to stick to the original query or are plainly invalid. We kindly refer readers to subsection 3.3 (case analysis) for a more detailed discussion. Due to space constraints, we do not introduce each benchmark setting, nor do we discuss our results in each benchmark in full length. Rather, this section provides a comprehensive demonstration of our method’s performance and provides a spotlight to provide preliminary empirical analysis about why and when our method works and when it fails.

3.1. Experimental Setups

We adopt four groups of benchmarks.

Code Generation includes HumanEval (Chen et al., 2021), HumanEval+ (Liu et al., 2023b), MBPP (Austin et al., 2021), and MBPP+ (Liu et al., 2023b). These benchmarks encompass a wide range of programming problems, from simple function implementations to more complex algorithmic challenges, providing a robust testbed for assessing generative capabilities.

Mathematical Reasoning evaluation is conducted on GSM8K and GSM-HARD dataset, which comprises thousands of multi-step mathematical problems (Cobbe et al., 2021; Gao et al., 2022).

Creative Writing tasks are conducted to evaluate the versatility of RAT, including survey, summarization *etc.*, highlighting different aspects of open-ended text generation.

Embodied Planning tasks are evaluated on open-ended environments Minecraft. A set of 100 tasks ranging from simple objectives to challenging diamond objectives are evaluated through MC-TextWorld (Lin et al., 2023).

Evaluation Metrics. For code generation, the classical pass rate $\text{pass}@k$ is selected as the evaluation metrics (Chen et al., 2021; Liu et al., 2023b), k denotes the sampling number. We compute accuracy to evaluate every question in mathematical reasoning tasks, aligning with the established metric for the GSM8K (Cobbe et al., 2021). For embodied planning tasks, we compute the plan execution success rate in MC-TextWorld as executability (Lin et al., 2023). We also conduct human elo rating evaluation to compute the trueskill rating score (Herbrich et al., 2006) for embodied planning (as plausibility) and creative writing tasks. These indicators are better the higher they are.

Baselines. To establish a comprehensive and equitable comparison landscape, we incorporate a suite of baseline methods. Our baselines include the original language models, referred to as DIRECT, and the Retrieval-Augmented Generation (RAG) methodology with n retrieved examples, instantiated in both single-shot (1 shot) and multi-shot (5 shots) configurations, as doc-

Table 1 | Code generation results on different benchmarks.*All tests are evaluated under zero-shot (0-demonstration) settings. We also report the **relative improvements** between RAT and DIRECT methods.

Base Models	Method	HumanEval		HumanEval+		MBPP		MBPP+		Average \uparrow	
		pass@1	pass@5	pass@1	pass@5	pass@1	pass@5	pass@1	pass@5	pass@1	pass@5
CodeLlama-7b	DIRECT	33.78%	40.85%	30.85%	36.59%	39.27%	54.27%	41.22%	48.17%	36.28%	44.97%
	CoT	27.86%	29.58%	25.12%	27.83%	31.99%	55.91%	42.19%	47.51%	31.79%	40.21%
	RAG_1 shot	37.50%	47.65%	33.66%	41.83%	35.41%	51.63%	43.66%	50.09%	37.56%	47.80%
	RAG_5 shot	38.90%	47.90%	35.37%	42.75%	34.06%	53.90%	43.35%	51.08%	37.92%	48.91%
	RAT	39.57%	51.34%	36.22%	46.50%	40.86%	60.63%	39.14%	48.04%	38.95%	51.63%
Relative Improvement		17.14%	25.68%	17.41%	27.08%	4.05%	11.72%	-5.05%	-0.27%	7.35%	14.80%
GPT-3.5	DIRECT	50.49%	72.56%	48.09%	70.55%	60.84%	72.95%	54.92%	64.09%	53.59%	70.04%
	CoT	47.31%	75.88%	41.72%	74.85%	55.19%	65.49%	47.69%	62.94%	47.98%	69.79%
	RAG_1 shot	50.61%	76.22%	48.22%	70.55%	55.23%	70.54%	53.62%	68.09%	51.92%	71.35%
	RAG_5 shot	45.49%	74.39%	42.58%	70.55%	54.39%	69.73%	55.98%	70.10%	49.61%	71.19%
	RAT	59.27%	80.49%	56.31%	76.07%	59.31%	74.74%	59.10%	72.61%	58.50%	75.98%
Relative Improvement		17.39%	10.93%	17.09%	7.82%	-2.51%	2.45%	7.61%	13.29%	9.17%	8.48%
GPT-4	DIRECT	57.32%	78.66%	54.36%	76.69%	60.00%	76.07%	66.13%	78.53%	59.45%	77.49%
	CoT	54.87%	72.56%	51.90%	66.25%	61.22%	74.23%	64.42%	79.75%	58.10%	73.20%
	RAG_1 shot	61.10%	79.27%	58.04%	77.30%	58.53%	69.94%	65.77%	77.30%	60.86%	75.95%
	RAG_5 shot	62.80%	82.93%	59.51%	79.75%	60.12%	74.23%	63.56%	78.53%	61.50%	78.86%
	RAT	69.33%	88.40%	64.63%	82.21%	68.90%	79.85%	67.36%	82.14%	67.55%	83.15%
Relative Improvement		20.94%	12.38%	18.89%	7.20%	14.83%	4.97%	1.86%	4.60%	13.63%	7.31%

Table 2 | **Evaluation results on mathematical reasoning, creative writing, and embodied planning tasks.** Among them, mathematical reasoning and creative writing use gpt-3.5 as base models, while embodied planning uses gpt-4 as base models. Δ represents the relative improvements than DIRECT.

Method	Math Reasoning Accuracy * \uparrow			Creative Writing \uparrow			Embodied Planning \uparrow		
	GSM8K	GSMHard	Average (Δ)	Win Rate	TrueSkill Rating (Δ)	Uncertainty	Executability	Plausibility (Δ)	Uncertainty
DIRECT	65.85%	51.26%	58.56%	46.67%	24.39	1.17	19.33 \pm 2.08%	20.57	2.05
CoT	63.82%	44.72%	54.27(-7.32)%	41.67%	24.31(-0.0%)	1.09	49.33 \pm 3.05%	25.75(+25.2%)	2.33
RAG-1 shot	61.81%	51.26%	56.54(+4.17)%	38.71%	23.99(-1.6%)	1.11	31.00 \pm 5.29%	24.97(+21.4%)	2.11
RAG-5 shot	61.81%	56.78%	59.30(+4.88)%	31.67%	23.88(-2.1%)	1.22	33.00 \pm 3.61%	25.02(+21.6%)	2.11
RAT	71.36%	67.34%	69.35(+16.96)%	81.01%	29.07(+19.2%)	1.08	76.67\pm8.02%	29.37(+42.78%)	3.37

umented by Lewis et al. (2020b). Additionally, we examine the zero-shot CoT (CoT) approach, as conceptualized by Kojima et al. (2022), which simulates a step-by-step reasoning process to facilitate complex problem-solving tasks under zero demonstration. For different methods, the same language model is used as base models. To ensure a fair comparison, none of the methods used examples from the benchmark as demonstrations for in-context learning.

RAG Settings. RAT leverages the capabilities of Retrieval-Augmented Generation methods, which enhance the performance of language models by integrating external knowledge sources. Specifically, we employed the `codeparrot/github-jupyter` dataset as our primary search vector library for code generation and mathematical reasoning tasks. For embodied planning tasks in Minecraft, we utilized the Minecraft Wiki¹ and DigMinecraft² websites as the information sources accessible to the LLMs.

¹<https://minecraft.wiki/>

²<https://www.digminecraft.com/>

For open-ended creative writing tasks, we use Google to search the query on the Internet. We utilized OpenAI’s `text-embedding-ada-002` API service for all embedding calculations across different methods and base models.

Acknowledging the risk of benchmark contamination (an issue where the code library may contain solutions to the exact problems being evaluated), we adopted a rigorous pre-processing methodology as described by Guo et al. (2024). The potential implications of benchmark contamination, along with the effectiveness of our pre-processing strategy, are discussed in detail in Appendix D.

3.2. Results

The code generation results presented in Table 1 and results on other benchmarks presented in Table 2 demonstrate the comprehensive evaluation of the RAT across multiple benchmarks. RAT consistently outperforms the other methods across the majority of the benchmarks and met-

rics, showcasing its superior ability to generate long-horizon context. Notably, in the HumanEval and HumanEval+ benchmarks of code generation, RAT achieves remarkable improvements in pass@1 and pass@5 rates, indicating a significant enhancement in first-attempt accuracy and within the top five attempts. For example, on the HumanEval benchmark, RAT improves pass@1 by up to 20.94% and pass@5 by up to 25.68% relative to the base models’ performances. This trend is observed across different underlying base models, highlighting RAT’s effectiveness regardless of the initial model’s capabilities. For mathematical reasoning tasks, RAT demonstrates a significant relative improvement, with an 8.37% increase in accuracy on GSM8K and a remarkable 31.37% increase on GSMHard, culminating in an overall average improvement of 18.44% when deployed on the GPT-3.5 model. RAT significantly outperforms all other methods on open-ended embodied planning tasks in Minecraft, achieving the highest scores with $76.67 \pm 8.02\%$ for executability and 29.37 human rating score for plausibility, demonstrating its superior ability to generate feasible and contextually appropriate plans in the complex open-world environment. RAT’s superior performance also keeps across a broad spectrum of creative writing tasks. Its ability to generate high-quality content in diverse scenarios was demonstrated, highlighting its potential as a powerful tool for enhancing the general creative writing capabilities of LLMs in open-ended scenarios.

The tasks are extremely diverse, while RAT can have consistent improvements over all baselines. These results underline the advantages of RAT’s approach, which leverages iterative refinement of retrieval queries based on evolving reasoning thoughts. This strategy not only enhances the relevance and quality of the information retrieved but also significantly improves the accuracy and efficiency of the generated context.

3.3. Case Analysis

Here we take the embodied planning task and creative writing task to do case analysis.

In a manner analogous to multi-document question-answering tasks (Trivedi et al., 2022a),

the task of long-horizon planning in Minecraft is knowledge-dense, requiring consideration of various items for the completion of each task. However, open-world Minecraft knowledge on the internet is fragmented, making task completion often dependent on information from multiple sources. We observed that while language models like ChatGPT can identify necessary items through zero-shot CoT reasoning, inaccuracies in procedural steps are common. For example, ChatGPT inaccurately identified the materials for a crafting table as 4 wood blocks (the right answer is 4 planks), indicating lower executability reliability in CoT plans. Classical RAG algorithms, retrieving the knowledge with the question as a query and focusing on the final target item, inadequately retrieve intermediary items, offering minimal task improvement. Contrastingly, RAT improves upon CoT’s initial answers by continuously refining thoughts with targeted retrieval, aligning closely with task progression and relevant item knowledge. This methodology significantly enhances planning effectiveness by ensuring a comprehensive understanding and retrieval of all items involved in a plan, highlighting the synergy between structured reasoning and dynamic knowledge retrieval in addressing long-horizon planning challenges in Minecraft.

In addressing open-ended creative writing tasks, assessments of LM’s generations typically focus on completeness and accuracy. When tasked with “summarizing the American Civil War according to a timeline”, LMs under DIRECT and CoT prompts often produce significant hallucinations. For example, the statement “The Civil War officially began on April 12, 1860, when Confederate troops attacked Fort Sumter in South Carolina, a Union-held fort” contains incorrect information, where the year 1860 is erroneously mentioned instead of the correct year, 1861.

Direct queries to the internet for this task tend to retrieve limited events, frequently overlooking the accurate start date of the war, April 12, 1861. Moreover, the RAG approach, which tends to summarize content retrieved from searches, often misses this event in its responses, whether it’s RAG-1 or RAG-5. On the other hand, RAT bases its search on a language model’s draft an-

Table 3 | Comparative Impact of Retrieval Strategies on RAT Performance.

Method	HumanEval		HumanEval+	
	pass@1(Δ) \uparrow	pass@5(Δ) \uparrow	pass@1(Δ) \uparrow	pass@5(Δ) \uparrow
Baseline	50.6%	76.2%	48.2%	70.5%
CoT+RAG	53.9(+3.3)%	76.8(+0.6)%	51.3(+3.1)%	69.3(-1.2)%
RAT	59.2(+8.7)%	80.4(+7.9)%	56.3(+8.2)%	76.0(+5.5)%

swer, finding that hallucinations usually occur in details, such as specific dates, which do not hinder the search engine from identifying relevant information like “American Civil War starting date”. RAT utilizes the content retrieved to identify and correct errors in the draft answer rather than merely summarizing the retrieved content. Therefore, RAT can achieve a complete generation through reasoning and enhance the accuracy and credibility of the answer by leveraging retrieved knowledge. Experimental results validate the effectiveness of RAT.

3.4. Ablation Study

Ablation on retrieval in RAT. In this ablation study, we investigate the influence of various retrieval strategies on the efficacy of RAT, focusing on the optimization of content retrieval for improving generative outputs. The experimental results, detailed in Table 3, highlight the significant advancements achieved through the iterative refinement of retrieval queries in RAT compared to baseline methods. The baseline denoted as RAG-1, employs a direct approach by using the question itself as the retrieval query. In contrast, CoT+RAG enhances this process by utilizing the entirety of the reasoning thoughts output by the language model as the query, aiming for a broader contextual understanding. However, RAT introduces a more dynamic method by employing continuously modified parts of reasoning thoughts as queries, which allows for a more focused and relevant information retrieval process. The comparative analysis shows that RAT surpasses both the baseline and the CoT+RAG method in terms of pass@1 and pass@5 metrics across the HumanEval and HumanEval+ benchmarks. Specifically, RAT demonstrates an 8.7 percentage point increase in pass@1 and a 7.9 percentage point increase in pass@5 over the baseline in the HumanEval benchmark, and similarly

Table 4 | Ablation Study on Causal vs. Non-Causal Reasoning in RAT.

Method	HumanEval		HumanEval+	
	pass@1(Δ) \uparrow	pass@5(Δ) \uparrow	pass@1(Δ) \uparrow	pass@5(Δ) \uparrow
Baseline	47.3%	75.8%	41.7%	74.8%
Non-Causal	57.3(+10.0)%	78.0(+2.1)%	54.9(+13.2)%	74.8(+0.0)%
Causal	59.2(+11.9)%	80.4(+4.6)%	56.3(+14.6)%	76.0(+1.2)%

impressive gains in the HumanEval+ benchmark. These improvements underscore the effectiveness of RAT’s retrieval strategy, which by iteratively refining next queries based on evolving reasoning thoughts and previous queries, ensures the retrieval of highly pertinent information. This process not only enhances the relevance of the information retrieved but also significantly improves the quality and accuracy of the final generated outputs. The results firmly establish the superiority of RAT’s dynamic retrieval method in leveraging contextual nuances to drive more precise and effective generative processes.

Ablation on causal reasoning in RAT. In this ablation study, we systematically examine the impact of causal and non-causal reasoning approaches on the performance of the RAT system, with the Chain of Thought (CoT) serving as our baseline. Our findings, as summarized in Table 4, reveal significant enhancements in generation capabilities when incorporating causal reasoning techniques. Specifically, the causal approach, which iteratively performs reasoning and retrieval, leads to notable improvements in both pass@1 and pass@5 metrics across HumanEval and HumanEval+ benchmarks. For instance, the causal method outperforms the baseline (CoT) by 11.9 percentage points in pass@1 and by 4.6 percentage points in pass@5 on the HumanEval dataset. This approach contrasts with the non-causal method, which, although also surpassing the baseline, leverages the initial reasoning thought to directly retrieve all necessary steps and generate the final answer. The causal method’s superior performance underscores the value of sequential reasoning and information retrieval in enhancing the accuracy and reliability of generated outputs. This iterative process likely aids in refining the search and reasoning steps based on continuously updated context, allowing for more precise and relevant information retrieval, which in turn supports more accurate final answers.

These results firmly establish the efficacy of causal reasoning in long-horizon problem-solving tasks.

3.5. Robustness of RAT

RAT was rigorously validated across a diverse set of tasks, including code generation, mathematical reasoning, creative writing, and embodied planning. This variety of tasks underscores the generalization capability of RAT, demonstrating its robust performance across highly diverse challenges. Furthermore, all our experimental settings were conducted in a zero-shot manner; we did not design task-specific prompts for RAT, but rather used the simplest possible prompts (which can be found in Appendix B) to articulate questions or instructions for all methods. This approach ensures RAT’s generalization ability in open-ended scenarios.

The diversity of our evaluation was further enhanced by testing RAT across various language models of differing capacities. This included CodeLlama-7b (Rozière et al., 2023), ChatGPT (gpt-3.5-turbo) (Ouyang et al., 2022), and the more advanced GPT-4 (gpt-4) model (OpenAI, 2023). Remarkably, RAT maintained its generalization capability across different scales of language models, showing improvements in benchmarks such as the HumanEval for code generation tasks. Notably, the largest improvement was observed with GPT-4, attributed to its superior ability for in-context learning from retrieved text. On MBPP+, CodeLlama-7b based RAT has demonstrated performance degradation. This decline could be due to the limited in-context learning ability of smaller language models.

For mathematical reasoning tasks, RAT demonstrated a significant relative improvement, with an overall average improvement of 18.44% when applied to the GPT-3.5 model. This trend of improvement persisted with GPT-4, which achieved a remarkable 10.26% relative improvement from DIRECT to RAT. These findings highlight RAT’s robustness and its effective enhancement of language models’ performance across a spectrum of computational and creative tasks.

4. Related Works

Retrieval-augmented Generation (RAG). Recently, RAG has gained popularity for boosting the performance of LLMs by guiding their generation process using the retrieved knowledge (Zhao et al., 2023). Without updating model parameters that may be expensive (Lewis et al., 2020a) or unstable (Ke et al., 2022a,b), RAG is a cost-effective way for LLMs to interact with the external world (Gu et al., 2018; Lewis et al., 2020a). RAG is widely applied to downstream tasks, such as code generation (Lu et al., 2022; Nashid et al., 2023; Zhou et al., 2022b), question answering (Baek et al., 2023; Siriwardhana et al., 2023), and creative writing (Asai et al., 2023; Wen et al., 2023).

Reasoning-enhanced RAG. Some recent works also leverage reasoning to enhance the performance of RAG (Li et al., 2023b). For example, IRCoT (Trivedi et al., 2022b) exploits CoT to generate better queries for retrieval, IRGR (Ribeiro et al., 2022) performs iteratively retrieval to search for suitable premises for multi-hop QA, GEEK (Liu et al., 2023a) can choose to query external knowledge or perform a single logical reasoning step in long-horizon generation tasks, and ITRG (Feng et al., 2023a) performs retrieval based on the last-step generation. However, these previous RAG methods simply adopt a single query to retrieve the knowledge for question-answering tasks (Feng et al., 2023b; Gao et al., 2023), while our proposed RAT performs retrieval using reasoning and draft answers in an autoregressive way, which significantly improves the performance of RAG in various tasks as demonstrated in Figure 2.

Language Model for Reasoning. The advancement of reasoning in language models has seen notable methodologies emerge since CoT was proposed by Wei et al. (2022), which showcased LMs’ ability to generate self-derived problem-solving strategies. This foundational work spurred further innovations such as the least-to-most prompting (Zhou et al., 2022a), zero-shot CoT (Kojima et al., 2022), self-consistency (Wang et al., 2022), zero-shot CoT without prompting (Wang and Zhou, 2024). Moving beyond basic prompting, Creswell et al. (2022) intro-

duced the Selection-Inference framework, while [Zelikman et al. \(2022\)](#) developed STaR to refine reasoning through model finetuning. [Creswell and Shanahan \(2022\)](#) proposed a faithful reasoning model, segmenting reasoning into dedicated steps, similar to Scratchpad’s approach by [Nye et al. \(2021\)](#) for enhancing multi-step computation. Tree-of-Thought ([Yao et al., 2023](#)) and Graph-of-Thought ([Besta et al., 2023](#)) also expand the reasoning paths into a complex structure instead of linear CoT. These methods usually aim to improve the reasoning ability of LLM by designing prompts or providing feedback from the environment to assist in better planning and decision-making ([Li et al., 2023a](#); [Shinn et al., 2023](#); [Wang et al., 2023c](#); [Yao et al., 2022](#); [Zhang et al., 2023](#)). However, RAT takes a different approach by using RAG to access external knowledge that can help LLM with its reasoning process.

5. Conclusion

We have presented Retrieval Augmented Thoughts (RAT), a simple yet effective prompting strategy that synergies chain of thought (CoT) prompting and retrieval augmented generation (RAG) to address the challenging long-horizon reasoning and generation tasks. Our key ideas involve revising the zero-shot chain of thoughts produced by LLMs through RAG with the thoughts as queries, and causally revising the thoughts & generating the response progressively. RAT, a **zero-shot** prompting approach, has demonstrated significant advantages over vanilla CoT prompting, RAG, and other baselines on challenging code generation, mathematics reasoning, embodied task planning, and creative writing tasks.

Acknowledgments

We thank a grant from CCF-Tencent Rhino-Bird Open Research Fund. One author is funded in part by NSF grants #IIS-1943641, #IIS-1956441, #CCF-1837129, an SRA from Meta and a research gift from Amazon Alexa AI, and a gift from RelationalAI.

Limitations

In this section, we discuss three limitations of our RAT as follows.

One limitation of this work is that the performance of RAT relies on the chain-of-thought reasoning and in-context learning (or RAG) capability of the base LLM. Since this work does not involve any model training, the capability of base LLM will not change when applying RAT. Despite RAT achieves significant improvement on powerful LLMs such as GPT-3.5 and GPT-4, the effect on smaller and weaker LLMs such as GPT-2 is questionable. On top of that, it is interesting to further explore how to improve RAT via finetuning weaker LLMs ([Ke et al., 2023](#); [Lin et al., 2024](#)).

Another limitation of this work is that the performance of RAT also relies on the quality of the retrieved knowledge. When we have an inferior external knowledge base which is irrelevant to the user query, the retrieved knowledge may be unhelpful for LLMs to generate useful information. Also, even if we select a relatively large knowledge base that entails the relevant information, it will be expensive to maintain and retrieve from such a huge knowledge base and also hurts the retrieval precision. An interesting and crucial direction is to study how to build and evaluate the quality of a knowledge base used for efficient and effective retrieval.

It is noteworthy that the above two limitations also apply to the traditional studies on retrieval-augmented generation (RAG). The last limitation of RAT is that we follow CoT to solve the problems in an explicit step-by-step fashion. Sometimes step-by-step thinking may be redundant for straightforward questions, while some questions require more complex reasoning structures (e.g., tree-of-thoughts ([Yao et al., 2023](#))). It is also interesting to explore the better reasoning methods for LLMs in our future work.

Ethics Statement

All datasets and models are publicly accessible except for OpenAI’s GPT series and the text embedding APIs. We have not identified any signif-

icant ethical considerations associated with this work. We believe our newly proposed RAT can improve the generation of LLMs in various fields and reduce LLMs’ hallucinations.

References

- A. Asai, Z. Wu, Y. Wang, A. Sil, and H. Hajishirzi. Self-rag: Learning to retrieve, generate, and critique through self-reflection. *arXiv preprint arXiv:2310.11511*, 2023.
- J. Austin, A. Odena, M. Nye, M. Bosma, H. Michalewski, D. Dohan, E. Jiang, C. Cai, M. Terry, Q. Le, et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.
- J. Baek, A. F. Aji, and A. Saffari. Knowledge-augmented language model prompting for zero-shot knowledge graph question answering. *arXiv preprint arXiv:2306.04136*, 2023.
- B. Baker, I. Akkaya, P. Zhokhov, J. Huizinga, J. Tang, A. Ecoffet, B. Houghton, R. Sampedro, and J. Clune. Video pretraining (vpt): Learning to act by watching unlabeled online videos. *arXiv preprint arXiv:2206.11795*, 2022.
- M. Besta, N. Blach, A. Kubicek, R. Gerstenberger, L. Gianinazzi, J. Gajda, T. Lehmann, M. Podstawski, H. Niewiadomski, P. Nyczyk, et al. Graph of thoughts: Solving elaborate problems with large language models. *arXiv preprint arXiv:2308.09687*, 2023.
- T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.
- S. Cai, Z. Wang, X. Ma, A. Liu, and Y. Liang. Open-world multi-task control through goal-aware representation learning and adaptive horizon prediction. *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13734–13744, 2023a.
- S. Cai, B. Zhang, Z. Wang, X. Ma, A. Liu, and Y. Liang. Groot: Learning to follow instructions by watching gameplay videos. *arXiv preprint arXiv:2310.08235*, 2023b.
- M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. d. O. Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.
- K. Cobbe, V. Kosaraju, M. Bavarian, M. Chen, H. Jun, L. Kaiser, M. Plappert, J. Tworek, J. Hilton, R. Nakano, C. Hesse, and J. Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.
- A. Creswell and M. Shanahan. Faithful reasoning using large language models. *arXiv preprint arXiv:2208.14271*, 2022.
- A. Creswell, M. Shanahan, and I. Higgins. Selection-inference: Exploiting large language models for interpretable logical reasoning. *arXiv preprint arXiv:2205.09712*, 2022.
- S. Dhuliawala, M. Komeili, J. Xu, R. Raileanu, X. Li, A. Celikyilmaz, and J. Weston. Chain-of-verification reduces hallucination in large language models. *arXiv preprint arXiv:2309.11495*, 2023.
- Z. Feng, X. Feng, D. Zhao, M. Yang, and B. Qin. Retrieval-generation synergy augmented large language models. *ArXiv*, abs/2310.05149, 2023a.
- Z. Feng, X. Feng, D. Zhao, M. Yang, and B. Qin. Retrieval-generation synergy augmented large language models. *arXiv preprint arXiv:2310.05149*, 2023b.
- L. Gao, A. Madaan, S. Zhou, U. Alon, P. Liu, Y. Yang, J. Callan, and G. Neubig. Pal: Program-aided language models. *arXiv preprint arXiv:2211.10435*, 2022.
- Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai, J. Sun, and H. Wang. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2023.
- J. Gu, Y. Wang, K. Cho, and V. O. Li. Search engine guided neural machine translation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018.

- D. Guo, Q. Zhu, D. Yang, Z. Xie, K. Dong, W. Zhang, G. Chen, X. Bi, Y. Wu, Y. K. Li, F. Luo, Y. Xiong, and W. Liang. Deepseek-coder: When the large language model meets programming – the rise of code intelligence. *arXiv preprint arXiv:2401.14196*, 2024.
- R. Herbrich, T. Minka, and T. Graepel. Trueskill™: a bayesian skill rating system. *Advances in neural information processing systems*, 19, 2006.
- K. J. Holyoak and R. G. Morrison. *The Oxford handbook of thinking and reasoning*. Oxford University Press, 2012.
- W. Huang, P. Abbeel, D. Pathak, and I. Mordatch. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. *ICML*, 2022.
- Z. Ke, H. Lin, Y. Shao, H. Xu, L. Shu, and B. Liu. Continual training of language models for few-shot learning. *arXiv preprint arXiv:2210.05549*, 2022a.
- Z. Ke, Y. Shao, H. Lin, H. Xu, L. Shu, and B. Liu. Adapting a language model while preserving its general knowledge. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 10177–10188, 2022b.
- Z. Ke, Y. Shao, H. Lin, T. Konishi, G. Kim, and B. Liu. Continual pre-training of language models. In *The Eleventh International Conference on Learning Representations*, 2023.
- T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, and Y. Iwasawa. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213, 2022.
- P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474, 2020a.
- P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474, 2020b.
- C. Li, J. Liang, A. Zeng, X. Chen, K. Hausman, D. Sadigh, S. Levine, L. Fei-Fei, F. Xia, and B. Ichter. Chain of code: Reasoning with a language model-augmented code emulator, 2023a.
- X. Li, R. Zhao, Y. K. Chia, B. Ding, S. Joty, S. Poria, and L. Bing. Chain-of-knowledge: Grounding large language models via dynamic knowledge adapting over heterogeneous sources. In *The Twelfth International Conference on Learning Representations*, 2023b.
- S. Lifshitz, K. Paster, H. Chan, J. Ba, and S. McIlraith. Steve-1: A generative model for text-to-behavior in minecraft. *arXiv preprint arXiv:2306.00937*, 2023.
- H. Lin, Z. Wang, J. Ma, and Y. Liang. Mcu: A task-centric framework for open-ended agent evaluation in minecraft. *arXiv preprint arXiv:2310.08367*, 2023.
- H. Lin, B. Huang, H. Ye, Q. Chen, Z. Wang, S. Li, J. Ma, X. Wan, J. Zou, and Y. Liang. Selecting large language model to fine-tune via rectified scaling law. *arXiv preprint arXiv:2402.02314*, 2024.
- C. Liu, X. Li, L. Shang, X. Jiang, Q. Liu, E. Y. Lam, and N. Wong. Gradually excavating external knowledge for implicit complex question answering. In *Conference on Empirical Methods in Natural Language Processing*, 2023a.
- J. Liu, C. S. Xia, Y. Wang, and L. Zhang. Is your code generated by chatGPT really correct? rigorous evaluation of large language models for code generation. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023b.
- S. Lu, N. Duan, H. Han, D. Guo, S.-w. Hwang, and A. Svyatkovskiy. Reacc: A retrieval-augmented code completion framework. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6227–6240, 2022.

- N. Nashid, M. Sintaha, and A. Mesbah. Retrieval-based prompt selection for code-related few-shot learning. In *Proceedings of the 45th International Conference on Software Engineering (ICSE'23)*, 2023.
- M. Nye, A. J. Andreassen, G. Gur-Ari, H. Michalewski, J. Austin, D. Bieber, D. Dohan, A. Lewkowycz, M. Bosma, D. Luan, et al. Show your work: Scratchpads for intermediate computation with language models. *arXiv preprint arXiv:2112.00114*, 2021.
- OpenAI. Gpt-4 technical report, 2023.
- L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. L. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, et al. Training language models to follow instructions with human feedback. *arXiv preprint arXiv:2203.02155*, 2022.
- V. Rawte, A. Sheth, and A. Das. A survey of hallucination in large foundation models. *arXiv preprint arXiv:2309.05922*, 2023.
- N. Reimers and I. Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*, 2019.
- D. Ribeiro, S. Wang, X. Ma, R. Dong, X. Wei, H. Zhu, X. Chen, Z. Huang, P. Xu, A. Arnold, et al. Entailment tree explanations via iterative retrieval-generation reasoner. *arXiv preprint arXiv:2205.09224*, 2022.
- B. Rozière, J. Gehring, F. Gloeckle, S. Sootla, I. Gat, X. Tan, Y. Adi, J. Liu, T. Remez, J. Rapin, A. Kozhevnikov, I. Evtimov, J. Bitton, M. P. Bhatt, C. C. Ferrer, A. Grattafiori, W. Xiong, A. D’efosse, J. Copet, F. Azhar, H. Touvron, L. Martin, N. Usunier, T. Scialom, and G. Synnaeve. Code llama: Open foundation models for code. *ArXiv*, abs/2308.12950, 2023.
- N. Shinn, B. Labash, and A. Gopinath. Reflexion: an autonomous agent with dynamic memory and self-reflection. *arXiv preprint arXiv:2303.11366*, 2023.
- S. Siriwardhana, R. Weerasekera, E. Wen, T. Kaluarachchi, R. Rana, and S. Nanayakkara. Improving the domain adaptation of retrieval augmented generation (rag) models for open domain question answering. *Transactions of the Association for Computational Linguistics*, 11: 1–17, 2023.
- G. P. Team. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*, 2022.
- H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.
- H. Trivedi, N. Balasubramanian, T. Khot, and A. Sabharwal. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. *ArXiv*, abs/2212.10509, 2022a.
- H. Trivedi, N. Balasubramanian, T. Khot, and A. Sabharwal. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. *arXiv preprint arXiv:2212.10509*, 2022b.
- X. Wang and D. Zhou. Chain-of-thought reasoning without prompting. *arXiv preprint arXiv:2402.10200*, 2024.
- X. Wang, J. Wei, D. Schuurmans, Q. Le, E. Chi, S. Narang, A. Chowdhery, and D. Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022.
- X. Wang, J. Wei, D. Schuurmans, Q. V. Le, E. H. Chi, S. Narang, A. Chowdhery, and D. Zhou. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023*, 2023a.
- Z. Wang, S. Cai, A. Liu, Y. Jin, J. Hou, B. Zhang, H. Lin, Z. He, Z. Zheng, Y. Yang, X. Ma, and Y. Liang. Jarvis-1: Open-world multi-task agents with memory-augmented multimodal language models. *ArXiv*, abs/2311.05997, 2023b.
- Z. Wang, S. Cai, A. Liu, X. Ma, and Y. Liang. Describe, explain, plan and select: Interactive planning with large language models enables

- open-world multi-task agents. *arXiv preprint arXiv:2302.01560*, 2023c.
- J. Wei, X. Wang, D. Schuurmans, M. Bosma, E. Chi, Q. Le, and D. Zhou. Chain of thought prompting elicits reasoning in large language models. *36th Conference on Neural Information Processing Systems (NeurIPS 2022)*, 2022.
- Z. Wen, Z. Tian, W. Wu, Y. Yang, Y. Shi, Z. Huang, and D. Li. Grove: a retrieval-augmented complex story generation framework with a forest of evidence. *arXiv preprint arXiv:2310.05388*, 2023.
- S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.
- S. Yao, D. Yu, J. Zhao, I. Shafran, T. L. Griffiths, Y. Cao, and K. Narasimhan. Tree of thoughts: Deliberate problem solving with large language models, 2023.
- H. Yuan, C. Zhang, H. Wang, F. Xie, P. Cai, H. Dong, and Z. Lu. Plan4mc: Skill reinforcement learning and planning for open-world minecraft tasks. *arXiv preprint arXiv:2303.16563*, 2023.
- H. Yuan, Z. Mu, F. Xie, and Z. Lu. Pre-training goal-based models for sample-efficient reinforcement learning. In *The Twelfth International Conference on Learning Representations*, 2024.
- E. Zelikman, Y. Wu, J. Mu, and N. Goodman. Star: Bootstrapping reasoning with reasoning. *Advances in Neural Information Processing Systems*, 35:15476–15488, 2022.
- C. Zhang, K. Yang, S. Hu, Z. Wang, G. Li, Y. Sun, C. Zhang, Z. Zhang, A. Liu, S.-C. Zhu, et al. Proagent: Building proactive cooperative ai with large language models. *arXiv preprint arXiv:2308.11339*, 2023.
- R. Zhao, H. Chen, W. Wang, F. Jiao, X. L. Do, C. Qin, B. Ding, X. Guo, M. Li, X. Li, and S. R. Joty. Retrieving multimodal information for augmented generation: A survey. *ArXiv*, abs/2303.10868, 2023.
- D. Zhou, N. Schärli, L. Hou, J. Wei, N. Scales, X. Wang, D. Schuurmans, C. Cui, O. Bousquet, Q. Le, et al. Least-to-most prompting enables complex reasoning in large language models. *arXiv preprint arXiv:2205.10625*, 2022a.
- D. Zhou, N. Schärli, L. Hou, J. Wei, N. Scales, X. Wang, D. Schuurmans, C. Cui, O. Bousquet, Q. V. Le, and E. H. Chi. Least-to-most prompting enables complex reasoning in large language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023*, 2023.
- S. Zhou, U. Alon, F. F. Xu, Z. Jiang, and G. Neubig. Docprompting: Generating code by retrieving the docs. In *The Eleventh International Conference on Learning Representations*, 2022b.

A. Task Details

A.1. Code Generation

Benchmarks. We select HumanEval (Chen et al., 2021), HumanEval+ (Liu et al., 2023b), MBPP (Austin et al., 2021), and MBPP+ (Liu et al., 2023b) as the code generation evaluation benchmark. These benchmarks are commonly used to test the performance of code generation models, which are briefly introduced below:

- **HumanEval** consists of 164 Python programming problems, each with a function signature, docstring, body, and multiple unit tests (Chen et al., 2021).
- **HumanEval+** includes the same programming problems as HumanEval, but with an additional 80 times more unit tests for each of the 164 problems (Liu et al., 2023b).
- **MBPP** is a collection of approximately 1,000 Python programming problems that are intended to be solvable by beginner programmers. Each problem includes an English task description, a code solution, and three automated test cases. We assess the sample test set from index 11 to 175 (Austin et al., 2021).
- **MBPP+** consists of 399 tasks (Liu et al., 2023b), which are a subset of the original MBPP dataset. Additionally, MBPP+ includes extra unit tests for each of the 399 problems (35 times more than the original MBPP). We utilized the first 164 questions as our test set.

These benchmarks encompass a wide range of programming problems, from simple function implementations to more complex algorithmic challenges, providing a robust testbed for assessing the generative capabilities of various models.

Metrics. We adopt the $\text{pass}@k$ metric for evaluating the efficacy of various code generation algorithms, following the methodology proposed by Chen et al. (2021) and extended by Liu et al. (2023b). This metric quantifies the rate at which generated code snippets successfully execute and pass all test cases, where k represents the number of attempts or samples generated by the model for each problem. This approach allows us to rigorously assess the precision and reliability of code generation models in producing functionally correct code across a diverse set of programming challenges.

Baselines. To establish a comprehensive and equitable comparison landscape, we incorporate a suite of baseline methods and diverse code generation models. Our baselines include the original code generation language models, referred to as DIRECT, and the Retrieval-Augmented Generation (RAG) methodology with n retrieved examples, instantiated in both single-shot (1 shot) and multi-shot (5 shots) configurations, as documented by Lewis et al. (2020b). Additionally, we examine the zero-shot CoT (CoT) approach, as conceptualized by Kojima et al. (2022), which simulates a step-by-step reasoning process to facilitate complex problem-solving tasks under zero demonstration. To ensure a fair comparison, none of the methods used examples from the benchmark as demonstrations for in-context learning.

The diversity of our evaluation is further enriched by testing across various language models with differing capacities, including CodeLlama-7b (Rozière et al., 2023), along with ChatGPT(gpt-3.5-turbo) (Ouyang et al., 2022), and the more advanced GPT-4(gpt-4) model (OpenAI, 2023). Recognizing the potential format discrepancies in code outputs, especially considering that models like gpt-3.5-turbo and gpt-4 may produce code in markdown format which is not immediately executable, we implement post-processing steps to convert the original language model outputs into a form that can be executed within a sandbox environment. This normalization ensures that all models are evaluated under uniform execution conditions, thereby facilitating a fair and direct comparison of their code generation capabilities. Through this methodological framework,

we aim to provide a detailed and nuanced understanding of the performance landscape across a spectrum of LLM-driven code generation approaches.

RAG Settings. RAT leverages the capabilities of Retrieval-Augmented Generation methods, which enhance the performance of language models by integrating external knowledge sources. Specifically, we employed the `codeparrot/github-jupyter` dataset as our primary search vector library. This dataset is a comprehensive compilation of 452k markdown and code pairs, meticulously extracted from Jupyter notebooks hosted on GitHub BigQuery, representing a rich repository of programming knowledge and examples. We utilized OpenAI’s `text-embedding-ada-002` API service for all embedding calculations across different methods and base models.

A.2. Mathematical Reasoning

Benchmarks. Our evaluation framework for assessing mathematical reasoning capabilities leverages two primary benchmarks: the GSM8K dataset, which comprises over 8,000 multi-step mathematical problems (Cobbe et al., 2021), and the GSM-HARD dataset, an adaptation of GSM8K where numbers in the questions are replaced with larger values to increase problem complexity (Gao et al., 2022). This study employs the PAL methodology to scrutinize the mathematical reasoning results, involving the utilization of Large Language Models (LLMs) to parse natural language problems, generate intermediary programmatic solutions, and subsequently execute these solutions via a Python interpreter. The test set for each benchmark consists of samples ranging from index 1 to 200. Uniquely, our approach does not use any examples for in-context learning, differing from the original PAL methods.

Metrics and Baselines. Accuracy serves as our principal metric for evaluation, aligning with the established metric for the GSM8K benchmark. Each question undergoes three execution attempts, with the average score recorded as the final result. The baselines, including DIRECT, CoT, RAG (1 shot), and RAG (5 shots), are consistent with those outlined in code generation, facilitating a comprehensive and comparative analysis across different code generation benchmarks. The RAG settings are consistent with the code generation tasks.

A.3. Embodied Planning

We further conduct experiments on embodied planning benchmarks on open-ended environments Minecraft (Lin et al., 2023).

Benchmarks. The complexity and vast item interconnectivity within the open-world Minecraft present an ideal testbed for evaluating the LLM’s capability to generate long-horizon plans (Wang et al., 2023b,c; Yuan et al., 2023). With thousands of items and intricate relationships between them, obtaining a specific item in survival mode from scratch may involve dozens of intermediate items and their quantitative relationships, such as crafting 1 crafting table from 4 planks. This setting rigorously tests the planning abilities of LLMs instead of low-level control policies (Baker et al., 2022; Cai et al., 2023a,b; Lifshitz et al., 2023; Yuan et al., 2024). Moreover, Wang et al. (2023b) have identified instances of hallucinations about Minecraft knowledge in OpenAI’s ChatGPT and a general scarcity of Minecraft-related knowledge in open-source language models, making this task a suitable benchmark for assessing the RAG algorithm’s effectiveness.

The planning prompts are aligned with those used in DEPS (Wang et al., 2023c), structured as Python templates and evaluated using MC-TextWorld as detailed by Lin et al. (2023). A set of 100 tasks were randomly selected for the test set, ranging from simple objectives like obtaining a crafting table to more complex goals such as crafting an iron helmet and even challenging making an enchanting table. The task instruction is formulated as:

- Give you nothing in the inventory, generate a step-by-step plan for the task of obtaining a {placeholder:acacia_boat} in Minecraft survival mode, and describe the object Minecraft item and its number at every step. For every step, start with 'STEP' as start.
- Give you nothing in the inventory, generate a step-by-step plan for the task of obtaining a {placeholder:diamond_pickaxe} boat in Minecraft survival mode, and describe the object Minecraft item and its number at every step. For every step, start with 'STEP' as start.

There are over 100 tasks involving different Minecraft items.

RAG Settings. For the retrieval component of the RAG algorithm, we utilized the Minecraft Wiki³ and DigMinecraft⁴ websites as the information sources accessible to the LLMs. Data from these websites was cleaned and formatted into markdown text, then segmented into trunks not exceeding 2000 tokens each, with embedding calculations performed using OpenAI's `text-embedding-ada-002` API service.

Evaluation Metrics. Based on the methodology of Huang et al. (2022), our evaluation of open-ended, long-horizon planning in Minecraft focuses on both executability and plausibility. Executability primarily examines whether a plan can be carried out, including the accuracy of each step's preconditions and effects. The executability is automatically calculated using MC-TextWorld (Lin et al., 2023). However, executability only evaluates if an objective-level plan can be executed, without considering the specific details involved in executing individual objectives. For instance, crafting a wooden pickaxe requires placing a crafting table and arranging three planks and two sticks in a particular pattern, which are important details for human execution but not assessed by MC-TextWorld. Therefore, we complement our evaluation with human ratings to assess the plausibility of plans.

A.4. Creative Writing

To further understand the potential of Retrieval-Augmented Generation (RAG) models in enhancing the creativity and relevance of generated content, we extend our investigation to open-ended text generation tasks within the realm of creative writing.

Benchmarks. The versatility of RAT was tested through a series of creative writing tasks, each chosen to highlight different aspects of open-ended text generation. These tasks include:

- Write a survey paper to summarize the placeholder:Retrieval-augmented Generation methods for Large Language Models.
- Describe of placeholder:Jin-Yong's life.
- Summarize the placeholder:American Civil War according to the timeline.

For each task, three variants for placeholder were created to ensure a comprehensive evaluation of the model's performance across different contexts and requirements.

RAG Settings. Differing from previous tasks, creative writing is categorized as an open-ended generation task, demanding a broader scope of information retrieval to aid content generation. To accommodate this, Google was utilized as the search engine, with the top-k web pages converted into markdown text to assist the LLM in generating outputs. This approach allowed LLM to leverage a wide array of information sources.

Baselines and Evaluations. To benchmark RAT's performance, we compared it against DIRECT, RAG-1 shot, and RAG-5 shot methods, all based on the `gpt-3.5-turbo` model. The evaluation

³<https://minecraft.wiki/>

⁴<https://www.digminecraft.com/>

was conducted by human experts, employing the TrueSkill rating system (Herbrich et al., 2006) to calculate scores for each method. This evaluation framework enabled a comprehensive assessment of each model’s creative output quality, accuracy, relevance, and innovativeness.

B. Prompt Details

Our prompts consist of three parts: prompt for generating initial answer, prompt for generating search query, and prompt for revising answers according to retrieved context.

Prompt B.1: Prompt for generating initial answers in creative writing tasks

```
{user}
##Question:
{question}
##Instruction:
Try to answer this question/instruction with step-by-step thoughts and make the answer more structural.
Use /n/n to split the answer into several paragraphs.
Just respond to the instruction directly. DO NOT add additional explanations or introduction in the answer unless you
are asked to.
{assistant}
...
```

The process of query generation is omitted in code generation tasks. Instead, we use the generated code draft as a query and compute the embedding of it based on OpenAI Embedding services. For embodied planning and creative writing tasks, we will generate an additional query.

Prompt B.2: Prompt for generating open-search query in creative writing tasks

```
##Question:
{question}
##Content:
{answer}
##Instruction:
I want to verify the content correctness of the given question, especially the last sentences.
Please summarize the content with the corresponding question.
This summarization will be used as a query to search with Bing search engine.
The query should be short but need to be specific to promise Bing can find related knowledge or pages.
You can also use search syntax to make the query short and clear enough for the search engine to find relevant language
data.
Try to make the query as relevant as possible to the last few sentences in the content.
**IMPORTANT**
Just output the query directly. DO NOT add additional explanations or introduction in the answer unless you are
asked to.
{assistant}
...
```

Prompt B.3: Prompt for revising answer according to retrieved materials in creative writing tasks

```

{user}
##Existing Text in Wiki Web:
{content}
##Question:
{question}
##Answer:
{answer}
##Instruction:
I want to revise the answer according to retrieved related text of the question in WIKI pages.
You need to check whether the answer is correct.
If you find some errors in the answer, revise the answer to make it better.
If you find some necessary details are ignored, add it to make the answer more plausible according to the related text.
If you find the answer is right and do not need to add more details, just output the original answer directly.
**IMPORTANT**
Try to keep the structure (multiple paragraphs with its subtitles) in the revised answer and make it more structural
for understanding. Split the paragraphs with /n/n characters. Just output the revised answer directly. DO NOT add
additional explanations or announcement in the revised answer unless you are asked to.
{assistant}
...

```

C. TrueSkill Evaluation Framework

Part of the tasks in “Embodied planning” and “creative writing” involve using humans for labeling. Human labelers have 4 choices: “A is better”, “B is better”, “Tie” or “Both are bad”. In this case, “Tie” and “Both are bad” will be counted as a tie. For each task group, we have selected more than 10 professional annotators to provide labels. We use the Python “trueskill” package to calculate the win rate and score. The default score for every method is set as 25. In order to facilitate user understanding and selection, we also provide prompts when entering the system.

```

# Chatbot Arena : Benchmarking LLMs in the Wild
##Rules
- Refresh to obtain the question and its corresponding answers from two anonymous models.
- Vote for the better answer. And then click "New Round" to get a new question.
- If both answers are bad, vote for "Both are bad".
- If you want to skip, click "Skip".


## Principle
You can evaluate the performance of the model from the following aspects:
1. **Relevance**: Does it answer the question accurately?
2. **Accuracy**: Is it accurate? For example, a crafting table is made by combining 4 wooden planks, not 4 logs; a diamond axe requires 3 diamonds and 2 sticks to craft, not 3 sticks and 2 diamonds.
3. **Completeness**: Is it complete? For example, crafting a wooden pickaxe from logs requires first crafting wooden planks and then crafting sticks before finally being able to craft the pickaxe. The intermediate steps cannot be ignored.
4. **Readability**: Is it coherent?
5. **Executability**: Considering the characteristics of the game, is it executable?

## Vote now!

```

D. Discussions on Benchmark Contamination

To avoid the code library containing solutions to the exact problems being evaluated) in code generation benchmarks, we adopted a rigorous pre-processing methodology as described by Guo et al. (2024). This process was meticulously designed to remove any direct matches or overly similar

 **Vote now!**

Task Group

Minecraft Planning

Instruction

Give you nothing in the inventory, generate a step-by-step plan for the task of obtaining a acacia planks in Minecraft survival mode, and describe the object Minecraft item and its number at every step. For every step, start with 'STEP' as start.

Model A Response:

STEP 1:
Item: Empty Inventory
Plan: Look around your surroundings for Acacia trees. They have a unique grey-brown trunk with a flat canopy. You can usually find these trees in the Savanna biome.

STEP 2:
Item: Empty hands
Plan: Approach the Acacia tree. You need to start breaking the wooden part of the tree, also known as the trunk or log.

STEP 3:
Item: Acacia Log
Plan: After you've collected at least one Acacia log, open your inventory and place the Acacia log in the crafting section. Each log will create 4 wooden planks.

STEP 4:
Item: Acacia Planks
Plan: Once you've placed the Acacia Log in the crafting section of your inventory, you'll have Acacia Planks as the result in the box next to it. Drag them into your inventory.

By following these steps, you've successfully obtained Acacia Planks in survival mode on Minecraft. You started with nothing and using the resources around you were able to create a

Model A From:

[MASK]

Model B Response:

Step 1:
Item: Empty Inventory
Task: Your initial task requires you to locate an Acacia tree which are usually found in the Savanna biome, recognized by their unique gray-brown trunk and flat canopy. After finding the tree, chop it down using your hand or, if available, an axe. You will acquire Acacia logs in return which are essential for the creation of Acacia planks.

Step 2:
Item: Empty Hands
Task: Move closer to the Acacia tree and start breaking the wood, specifically the trunk or log. Once the log has been broken down, quickly collect the stripped Acacia logs which will appear in your hotbar. Make sure to store these logs in your inventory for future use.

Step 3:
Item: Acacia Log
Task: After collecting at least one Acacia log, you need to access your crafting table which contains a 3x3 crafting grid. Position your acquired log in the first box of the first row of the grid, which will create 4 Acacia planks instantly. Move these planks to your inventory. Remember, for each Acacia log you put in the crafting grid, you will yield 4 wooden planks.

Step 4:
Item: Acacia Planks

Model B From:

[MASK]

A is better

B is better

Tie

Both are bad

Skip

New Round

Use via API  · Built with Gradio 

Figure C.1 | The human evaluation UI. We will display responses from two different methods for the same instruction on the page simultaneously. The source of the response will be marked as [MASK], and after human labeling, [MASK] will be replaced with the specific method name.

code snippets from our search vector library, thereby ensuring that our evaluation remains fair and uncontaminated by pre-existing solutions. This examination aims to underscore the importance of maintaining the integrity of the evaluation process while utilizing external knowledge sources to augment the capabilities of language models in code-generation tasks.

Table 5 | Benchmark contamination study based on CodeLlama-7b-Python models. We find using the code corpus finetuning the CodeLlama models directly will damage model performance on the code generation benchmark, while using RAT for in-context generation yields better performance.

Method	HumanEval		HumanEval+	
	pass@1	pass@5	pass@1	pass@5
DIRECT	40.85%	53.65%	37.43%	48.78%
FINETUNE	29.02%	40.24%	26.34%	35.98%
RAT	45.73%	59.75%	43.29%	53.66%

To further explore the potential benchmark contamination, we also conducted additional finetuning on CodeLLaMA-7B-Python using the code corpus in Table 5.

E. More Results

E.1. Embodied Planning

Embodied planning involves multiple steps, each of which relies on specific world knowledge and causal knowledge (i.e., preceding steps are usually preconditions for subsequent steps), including recipes for items in Minecraft, tools for performing related actions, and quantity restrictions. Therefore, although the plan generated by ChatGPT may appear complete and correct, there are often errors within the steps that can affect the proper execution of the plan.

We mark the errors found in the generated plan in **red**.

Response E.1: Response from Zero-Shot CoT(based on ChatGPT-3.5) in embodied planning tasks

{**instruction**}

Give you nothing in the inventory, generate a step-by-step plan for the task of obtaining a golden apple in Minecraft survival mode, and describe the object Minecraft item and its number at every step. For every step, start with 'STEP' as start.

{**response** from Original ChatGPT}

STEP 1: Start by punching trees to gather at least 4 logs. - Minecraft item: 4x Logs

STEP 2: **Craft the logs into a crafting table.** - Minecraft item: 1x Crafting Table

STEP 3: Place the crafting table on the ground and right-click on it to open the crafting interface.

STEP 4: Use the crafting interface to **turn 4 of the oak planks into a wooden pickaxe.**

- Minecraft item: Wooden Pickaxe

STEP 5: Find a cave or dig into the ground to gather some stone. Mine at least 8 blocks of stone using your wooden pickaxe. - Minecraft item: 8x Cobblestone

STEP 6: Craft the cobblestone into a furnace using the crafting table. - Minecraft item: 1x Furnace

STEP 7: Place the furnace on the ground and right-click on it to open the furnace interface.

STEP 8: **Use the furnace interface to smelt some iron ore.** Place at least 3 iron ore in the top slot of the furnace and use wooden planks as fuel in the bottom slot. - Minecraft item: 3x Iron Ingots

STEP 9: Craft the iron ingots into a bucket using the crafting table. - Minecraft item: 1x Bucket

STEP 10: Find a water source (such as a lake or a river) and right-click on it while holding the bucket to fill it with water. - Minecraft item: 1x Water Bucket

STEP 11: Look for an apple tree in a forest biome. Break some leaves until you find an apple. - Minecraft item: 1x Apple

STEP 12: **Combine the water bucket and the apple in the crafting table to craft a golden apple.** - Minecraft item: 1x Golden Apple

Congratulations! You have successfully obtained a golden apple in Minecraft survival mode.

Although the Zero-shot CoT has generated a step-by-step plan overall, there are many factual errors within it. These include recipe errors in STEP 2, where the crafting table requires planks instead of wood; missing raw materials in STEP 4, as the wooden pickaxe needs 2 sticks but lacks the relevant step in the plan; absence of instructions to use a stone pickaxe to mine iron ore in STEP 8; and an incorrect recipe for golden apple in STEP 12, which should include gold ingots and an apple rather than a water bucket.

Response E.2: Response from RAT(based on ChatGPT-3.5) in embodied planning tasks

{instruction}

Give you nothing in the inventory, generate a step-by-step plan for the task of obtaining a golden apple in Minecraft survival mode, and describe the object Minecraft item and its number at every step. For every step, start with 'STEP' as start.

{response} from Original ChatGPT}

STEP 1: Find a tree and chop it down to get wood. You need at least 4 Oak Logs.

- Minecraft items: 4x Oak Logs

STEP 2: Craft the oak log into oak planks.

- Minecraft items: 16x Oak Planks

STEP 3: Convert some oak planks into sticks.

- Minecraft items: 4x Stick

STEP 4: Use the crafting table to create a wooden pickaxe.

- Minecraft items: 1x Wooden Pickaxe

STEP 5: Start digging down to find stone and gather at least three pieces of cobblestone.

- Minecraft items: 3x Cobblestone

STEP 6: Use the crafting table to create a stone pickaxe.

- Minecraft items: 1x Stone Pickaxe

STEP 7: Explore caves and mineshafts to find iron ore. Collect 3 iron ore for crafting an iron pickaxe.

- Minecraft items: 3x Iron Ore

STEP 8: Smelt the iron ore **using a furnace** to obtain iron ingot.

- Minecraft items: 3x Iron Ingot

STEP 9: Use the crafting table to create an iron pickaxe.

- Minecraft items: 1x iron pickaxe

STEP 10: Explore caves and mineshafts to find gold ore. Use an iron pickaxe or higher to mine the gold ore block. Collect as much gold ore as needed for crafting a golden apple.

- Minecraft items: 8x Gold Ore

STEP 11: Smelt the raw gold using a furnace to obtain a gold ingot.

- Minecraft items: 8x Gold Ingot

STEP 12: Find an oak tree and chop it down until an apple falls.

- Minecraft items: 1x Apple

STEP 13: Open the crafting menu and place the items in the correct pattern to craft a Golden Apple. In the 3x3 crafting grid, place 8 Gold Ingots in the first and third rows, leaving the middle square empty. In the second row, place 1 Gold Ingot in the first and third squares, and place an Apple in the middle square.

- Minecraft items: 1x Golden Apple

Congratulations! You have successfully obtained a golden apple in Minecraft survival mode.

There are still errors in the plan generated by RAT, such as the lack of tools before "STEP 8: smelt iron ore into iron ingots", which should be "Mine 8 cobblestone to craft 1 furnace". However, compared to the errors in ChatGPT, the error rate in the plan has been significantly reduced.

We have also listed the links of the retrieved pages involved in different steps. We can see that the text sources retrieved in each step generated by RAT are usually highly related to the synthesized item of that step. Traditional RAG uses instructions for retrieval and can only find the final step. While RAT can retrieve all links related to intermediate items, which improves the accuracy and plausibility greatly.

Table 6 | Page link of retrieved text in embodied Minecraft planning tasks.

Step	Item	Recipe	Link
1	4x Oak Log	-	https://minecraft.fandom.com/wiki/Log
2	16x Oak Planks	4x Oak Log	https://www.digminecraft.com/basic_recipes/make_oak_wood_plank.php
3	4x Stick	2x Oak Planks	https://www.digminecraft.com/basic_recipes/make_stick.php
4	1x Wooden Pickaxe	3x Oak Planks, 2 Stick	https://www.digminecraft.com/tool_recipes/make_wooden_pickaxe.php
5	3x Cobblestone	Wooden Pickaxe	https://minecraft.fandom.com/wiki/Cobblestone
6	1x Stone Pickaxe	3x Cobblestone, 2 Stick	https://www.digminecraft.com/tool_recipes/make_stone_pickaxe.php
7	3x Iron Ore	Stone Pickaxe	https://minecraft.fandom.com/wiki/Iron_Ore
8	3x Iron Ingot	3x Iron Ore	https://www.digminecraft.com/basic_recipes/make_iron_ingot.php
9	1 Iron Pickaxe	3x Iron Ingot, 2x Stick	https://www.digminecraft.com/tool_recipes/make_iron_pickaxe.php
10	8x Gold Ore	Iron Pickaxe	https://minecraft.fandom.com/wiki/Gold_Ore
11	8x Gold Ingot	8x Gold Ore	https://www.digminecraft.com/basic_recipes/make_gold_ingot.php
12	1x Apple	-	https://minecraft.fandom.com/wiki/Apple
13	1x Golden Apple	8x Gold Ingot, 1x Apple	https://www.digminecraft.com/food_recipes/make_golden_apple.php

E.2. Creative Writing

Response E.3: Response from ORIGINAL ChatGPT-3.5 in creative writing tasks

{**instruction**}

Describe the history of the United States. {**response** from Original ChatGPT}

The history of the United States dates back thousands of years, with indigenous peoples such as the Native Americans inhabiting the land long before the arrival of European settlers. However, the official history of the United States begins in 1776 when the Founding Fathers declared independence from Great Britain. This marked the start of the American Revolution, a conflict that ultimately led to the creation of the United States as an independent nation.

After gaining independence, the United States went through a period of rapid expansion in the 19th century. The country acquired vast amounts of territory, including the Louisiana Purchase and the annexation of Texas. This expansion was accompanied by conflicts, such as the Mexican-American War and the Civil War, which tested the unity of the nation.

In the 20th century, the United States emerged as a global superpower. The country played a significant role in both World War I and World War II, eventually becoming the leader of the free world during the Cold War with the Soviet Union. The United States experienced social and cultural shifts during this time, including the Civil Rights Movement and the rise of consumerism.

Today, the United States remains a powerful nation, with a diverse population and a complex history that continues to shape its identity.

Response E.4: Response from Zero-shot CoT in creative writing tasks

{instruction}

Describe the history of the United States.

{response} from Zero-shot CoT****Pre-Columbian Era****

The history of the United States begins long before the arrival of European explorers in the 15th century. The land was initially inhabited by diverse Native American tribes, such as the Apache, Sioux, Iroquois, and Cherokee. These societies had rich cultures, with their own languages, traditions, and social structures. They lived off the land, flourishing through hunting, fishing, farming, and trade. However, the indigenous populations faced significant challenges, including conflicts, displacement, and diseases brought by the European settlers.

****Colonial Period****

In the 16th century, European powers began to claim territories in what is now the United States. The Spanish were the first to establish colonies, initially focusing on Central and South America. Later, English, French, and Dutch explorers arrived, establishing their own colonies along the Atlantic coast. These colonies varied in purpose and organization, with the English colonies primarily seeking economic opportunities and religious freedom. Over time, thirteen British colonies emerged, forging their own identities, developing local governments, and experiencing an increase in population due to immigration.

****American Revolution and Beyond****

Tensions between the British colonies and their colonial masters grew, leading to the American Revolution in 1775. The colonies, unified under the Continental Congress, fought for independence from British rule. The war lasted several years, until the signing of the Treaty of Paris in 1783, which granted the newly formed United States of America independence. After achieving independence, the founding fathers convened to draft the Constitution, establishing the framework for the new nation's government.

This is just a brief overview, and the history of the United States is vast and complex. However, these steps provide a simplified understanding of how the nation evolved from Native American civilizations to a group of British colonies, to an independent country.

Response E.5: Response from RAG(1-shot) in creative writing tasks

{instruction}

Describe the history of the United States.

{response} from RAG-1

The history of the United States is a complex and rich tapestry that spans over four centuries. The story begins with the arrival of European explorers and settlers in the 16th century. In 1492, Christopher Columbus reached the Americas, which led to subsequent expeditions and colonization by the Spanish, French, Dutch, and English. The first successful English settlement was established at Jamestown, Virginia in 1607.

Over the next few centuries, the colonies grew and developed their own unique cultures, economies, and systems of government. Tensions rose between the American colonies and the British Crown, leading to the American Revolutionary War in 1775. The colonies declared their independence in 1776 and formed the United States of America, which became a republic based on democratic ideals and principles.

The young nation faced numerous challenges in its early years, including establishing a central government, expanding westward, and dealing with the issue of slavery. The 19th century saw rapid westward expansion, the Industrial Revolution, and the Civil War. The United States emerged from the Civil War as a more unified and powerful nation, with the abolition of slavery and the preservation of the Union. In the 20th century, the United States became a global superpower, playing a crucial role in both World Wars and the Cold War. It witnessed significant social movements for civil rights and equality, as well as advancements in technology and innovation.

Today, the United States is a diverse and influential country with a strong economy, global military presence, and a complex political landscape. Its history continues to shape its present and future, as the nation grapples with ongoing social, economic, and political challenges.

Response E.6: Response from RAG(5-shot) in creative writing tasks

{instruction}

Describe the history of the United States.

{response} from RAG-5

The history of the United States dates back to the colonization of the Americas by European explorers in the 16th century. The first permanent English settlement was established in Jamestown, Virginia in 1607. This marked the beginning of British colonization in North America.

Over the next century, more British colonies were established along the eastern seaboard, including Massachusetts, New York, Pennsylvania, and Georgia. These colonies grew and developed their own distinct cultures and economies.

Tensions between the colonies and the British government grew in the mid-18th century, leading to the American Revolutionary War in 1775. The colonists, led by figures such as George Washington, Thomas Jefferson, and Benjamin Franklin, fought for their independence from British rule.

In 1776, the Second Continental Congress adopted the Declaration of Independence, which declared the thirteen American colonies to be independent states. This marked the birth of the United States of America.

Following the Revolutionary War, the Founding Fathers drafted the Constitution of the United States, which established a federal government and outlined the rights and freedoms of American citizens. The Constitution was ratified in 1788 and is still the supreme law of the land.

Since its founding, the United States has experienced significant growth and development. It expanded westward through the Louisiana Purchase and the annexation of Texas, and it eventually stretched from coast to coast. The country also faced significant challenges, such as the Civil War, the Great Depression, and the Civil Rights Movement.

Today, the United States is a global superpower with a diverse population and a strong economy. It is known for its democratic system of government, individual freedoms, and cultural influence.

Response E.7: Response from RAT in creative writing tasks

{instruction}

Describe the history of the United States.

{response} from RAT

The history of the United States is characterized by several significant phases. The pre-colonial period witnessed diverse Native American populations with unique cultures inhabiting the land. This era transitioned into colonialism through European exploration and settlement by powers such as the Spanish, French, Dutch, and British, leading to the establishment of colonies along the East Coast with distinct economic and political frameworks.

A pivotal moment in American history was the American Revolution, beginning in 1775, as the colonies sought independence from British rule. The Declaration of Independence in 1776 marked the nation's founding on principles of liberty and democracy, establishing autonomy from Great Britain. This revolutionary period not only birthed a republic based on the will of the people but also forged American national identity, fostering a collective ethos entrenched in history and culture, inspiring revolutions worldwide and aiding in the establishment of republican governments elsewhere.

The United States stands out as the first modern nation to achieve independence through a national war of liberation, articulating its motives and aspirations in a declaration of independence. This event had global reverberations, spreading fundamental values like liberty, equality, natural and civil rights, and civic duty. The formative years were shaped by the drafting of the Constitution in 1787, defining the federal government's framework. The subsequent ratification of the Bill of Rights further cemented individual freedoms and outlined governance principles that have endured across the nation's history, ensuring a system of checks and balances that protect citizens' rights.

The 19th century in the United States witnessed significant developments, including westward expansion, industrialization, and conflicts like the Civil War (1861-1865), which decisively addressed issues of slavery and reinforced federal authority over the states. The late 19th and early 20th centuries were marked by rapid industrial growth, extensive immigration, urbanization, and overseas expansion, including involvement in conflicts such as the Spanish-American War and World War I. The 20th century saw the United States emerge as a global superpower after World War II, engaging in the Cold War with the Soviet Union and participating in conflicts like the Korean and Vietnam Wars, significantly impacting the nation's foreign policy and global standing.

Recent history includes social movements for civil rights, gender equality, and environmental protection, as well as economic shifts, advancements in technology, and challenges in areas such as healthcare and international relations.