Planning with MCTS: Enhancing Problem-Solving in Large Language Models

Anonymous authors
Paper under double-blind review

Abstract

Despite recent advances in Large Language Models (LLMs), their ability to solve complex reasoning problems remains limited by inconsistent planning and logical flaws. We present a novel framework that significantly enhances LLMs' problem-solving capabilities by leveraging Monte Carlo Tree Search (MCTS) for plan generation. Unlike previous approaches that apply MCTS to solution search, our method uniquely integrates MCTS into the planning phase, guided by specialized LLM-powered agents that evaluate plan quality. Experiments across diverse benchmark datasets demonstrate that our approach improves problem-solving accuracy by an average of 40.59% compared to zero-shot Chain-of-Thought prompting. Furthermore, we show that using smaller models for MCTS planning and larger models for execution can maintain high performance while reducing computational costs. This work opens new avenues for developing more robust and efficient AI systems capable of tackling complex real-world problems, with potential applications in fields requiring advanced logical reasoning and long-term planning. Our code examples are publicly available at this Anonymous Github Repository.

1 Introduction

In an era where artificial intelligence increasingly influences our daily lives, the quest for machines capable of human-like reasoning remains a central challenge in computer science. Large Language Models (LLMs) have emerged as powerful tools in natural language processing, demonstrating remarkable capabilities across various tasks (Brown et al., 2020; Chowdhery et al., 2023). However, when faced with complex, multi-step problems requiring logical consistency and long-term planning, even state-of-the-art models like GPT-4 (OpenAI et al., 2024) often falter, revealing a critical gap between machine and human problem-solving abilities.

The limitations of LLMs in complex reasoning tasks stem from their struggle to maintain logical consistency and handle long-term dependencies throughout extended problem-solving processes. Existing approaches, such as Chain-of-Thought (CoT) prompting (Wei et al., 2022), have shown promise in improving step-by-step reasoning. However, these methods often fail to address the fundamental challenge of generating and following a coherent, overarching plan (Wang et al., 2023b; Yao et al., 2023). Similarly, current planning methods for LLMs, including task decomposition (Patel et al., 2022; Zhou et al., 2023) and explicit plan-and-solve approaches (Wang et al., 2023a; Yao et al., 2023), are ultimately constrained by the LLM's inherent reasoning abilities.

This research addresses a critical question: How can we enhance the planning capabilities of LLMs to significantly improve their problem-solving performance in complex, multi-step tasks? We hypothesize that by leveraging advanced search algorithms in the planning phase, we can generate higher-quality plans that guide LLMs towards more effective and logically consistent solutions.

To this end, we propose a novel framework that integrates Monte Carlo Tree Search (MCTS) into the planning process for LLMs. MCTS, renowned for its success in complex game-playing AI (Silver et al., 2016), offers a powerful method for exploring vast search spaces and identifying optimal strategies. Our approach uniquely applies MCTS to the generation of problem-solving plans, rather than to direct solution search. This is achieved through a two-step process: first, MCTS explores the space of possible plans, guided by specialized LLM-powered agents that evaluate plan quality; then, the optimal plan is provided to the LLM for step-by-step execution.

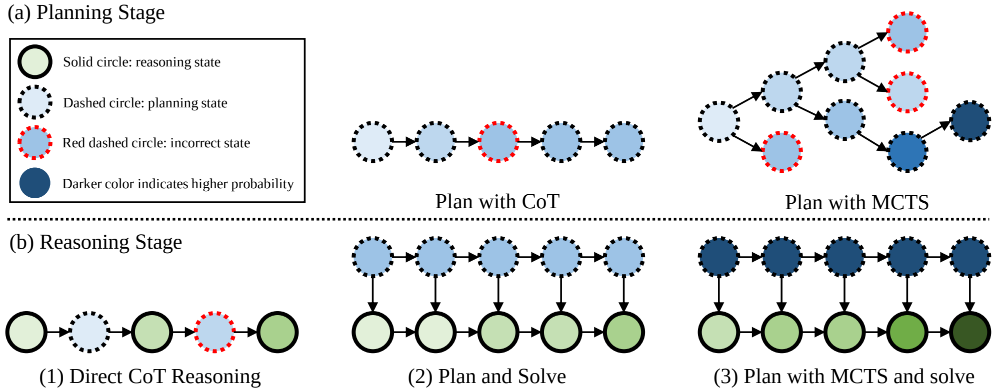

Figure 1: Illustration of the proposed framework for enhancing LLM problem-solving with MCTS planning. (a) CoT prompting generates planning and reasoning states interleavedly, while post states may suffer from cumulative error. (b) Plan and solve framework separates planning and reasoning, while planning is generated manually or based on CoT, which may not be optimal. (c) Our proposed MCTS-enhanced planning approach explicitly generates high-quality plans using MCTS, which are then executed by the LLM for problem-solving. Darker colors of each state indicate higher probability, which leads to better problem-solving performance.

The significance of this research extends beyond mere incremental improvements in LLM performance. By enhancing the planning and problem-solving capabilities of LLMs, we pave the way for more robust and reliable AI systems capable of tackling complex real-world problems. This has far-reaching implications for fields such as automated reasoning, decision support systems, and AI-assisted scientific discovery.

Our work makes several key contributions to the field of AI and natural language processing:

1. We introduce a novel paradigm for enhancing LLMs' problem-solving abilities by applying MCTS to the planning process, addressing a critical gap in current approaches.
2. We provide empirical evidence demonstrating significant improvements in problem-solving accuracy across diverse benchmark datasets, with an average improvement of 40.59% compared to zero-shot Chain-of-Thought prompting.
3. We offer insights into the interplay between planning, reasoning, and execution in LLM-based problem-solving, contributing to a deeper understanding of AI cognition.
4. We propose an efficient approach using smaller models for MCTS planning and larger models for execution, maintaining high performance while reducing computational costs.

2 Searching Plan for Better Problem-Solving

This section presents our novel approach to enhancing the problem-solving capabilities of LLMs through the integration of MCTS in the planning process. We first provide the necessary background and problem formulation, followed by a detailed description of our MCTS-enhanced planning framework.

2.1 Preliminary: From CoT to Planning

LLMs have demonstrated remarkable performance in various natural language processing tasks, yet their ability to solve complex, multi-step problems remains limited. Existing approaches, such as CoT prompting (Wei et al., 2022), often struggle to maintain logical consistency and handle long-term dependencies in reasoning chains, especially as problem complexity increases (Wang et al., 2023b; Yao et al., 2023).

Formally, we can view problem-solving with LLMs as a process of generating a solution Y given a problem X and a context C, which can be represented as a conditional probability distribution

Count Characters: Example 1

Problem: how many R's are there in the word "Strawberry"?
—
CoT Prompt: Let's think step by step.
CoT Result:
Here's how to count the 'R's in "strawberry":
1. **Identify the 'R's:** Look for the letter 'R' in the word.
2. **Count:** There are **two** 'R's in the word "strawberry".
—
Planning Prompt:
To count the R's in "strawberry." we can iterate throuoh each letter of the word, Initially, we'll set a counter to zero, for each leter encountered, we'll check if itis an "R" character. If it is, we'll increment our counter, This proess will continwe until we've examined all letters in the word, allowing us to determine the final count of "R's present.
Planning Result:
Following the plan:
1. **Initialize counter:** 'r_count = 0'.
2. **Iterate through the word:** Check each letter of "strawberry" for 'R'.
3. **Increment counter:** If the letter is 'R', increment the counter ('r_count = r_count + 1').
4. **Final Count:** The word "strawberry" contains **3** 'R's.
—
MCTS Planning:
1. **Define the task:** Our goal is to count the number of "R's present within the word "strawberry."
2. **Iterate through the word:** We will examine each letter of "strawberry" one at a time.
3. **Check for "R":** If the letter is an "R", increment a counter to keep track of our count.
4. **Increment counter:** If the letter is an "R", increment a counter to keep track of our count.
5. **Complete iteration:** We will continue this process until we have examined every letter.
6. **Return the count:** After processing all letters, the counter will hold the final number of "R's.
MCTS Planning Result:
Here's the breakdown:
1. **Define the task:** We need to count how many "R's are in "strawberry".
2. **Iterate through the word:** Let's examine each letter:
* **s** - Not an R, counter remains 0.
* **t** - Not an R, counter remains 0.
* **r** - **It's an R! Increment the counter to 1.**
* **a** - Not an R, counter remains 1.
* **w** - Not an R, counter remains 1.
* **b** - Not an R, counter remains 1.
* **e** - Not an R, counter remains 1.
* **r** - **It's an R! Increment the counter to 2.**
* **r** - **It's an R! Increment the counter to 3.**
* **y** - Not an R, counter remains 3.
3. **Complete iteration:** After examining all letters, the counter holds **3** R's.
4. **Result:** There are **3** R's in the word "strawberry".

Figure 2: Character counting example

$P(Y|X, C)$. In zero-shot CoT prompting, the context C is typically the problem itself, and the LLM attempts to generate the solution directly through a series of reasoning steps $X = \{X_1, X_2, ..., X_n\}$. This direct approach often leads to suboptimal solutions due to the intertwined nature of planning and reasoning.

To address this challenge, we employ a planning-based approach that decouples planning from reasoning. We introduce a plan π as an ordered sequence of actions, where $\pi \in \Pi$, and Π represents the space of all possible plans. In this framework, the context C can be decomposed into two parts: the problem description $C_{problem}$ and a plan π generated based on the problem, which we denote as C_{plan}. Thus, we can rewrite the conditional probability as:

$$P(Y|X, C) = P(Y|X, C_{problem}, C_{plan}) \tag{1}$$

Assuming that the solution Y depends on the problem X and the plan C_{plan}, but not directly on the problem description $C_{problem}$ once the plan is given, we can factorize this probability using the chain rule:

$$P(Y|X, C_{plan}) = P(Y|X, C_{plan})P(X|C) \tag{2}$$

This factorization highlights the two distinct stages of our approach:

1. **Planning** ($P(X|C)$): Generating a sequence of reasoning steps (X) based on the initial context (C), which includes the problem description.
2. **Reasoning** ($P(Y|X, C_{plan})$): Generating the solution (Y) given the problem (X) and the plan (C_{plan}).

By explicitly separating these stages, we aim to improve the logical consistency and coherence of the problem-solving process. The planning stage focuses on finding an effective plan, while the reasoning stage leverages the plan to guide the generation of the solution.

Instead of relying on the LLM to implicitly generate a plan within its reasoning process, we leverage MCTS (Chaslot et al., 2008) to explicitly explore the plan space Π during the planning stage. MCTS is a powerful heuristic search algorithm that has achieved remarkable success in complex game AI (Silver et al., 2016). We adapt MCTS to the problem-solving domain by representing plans as nodes in the search tree and using specialized LLM-powered agents to evaluate the quality of each plan. These agents assess properties such as the logical consistency and feasibility of the plan, providing feedback that guides the MCTS exploration.

This approach differs significantly from previous work that has applied MCTS to LLMs. While some studies have used Breadth First Search (BFS) or Depth First Search (DFS) to guide the selection of reasoning steps in CoT (Yao et al., 2023), our framework focuses on using MCTS to optimize the plan itself, providing a more structured and potentially more effective approach to complex problem-solving.

By decoupling planning and reasoning, and by leveraging the strengths of MCTS and LLMs, our proposed framework aims to enhance the problem-solving capabilities of LLMs, particularly in scenarios requiring complex reasoning and long-term planning.

2.2 Monte Carlo Tree Search for Planning

To address the limitations of existing planning methods and improve the coherence and logical consistency of LLM problem-solving, we propose leveraging MCTS for plan generation. Unlike approaches that use MCTS to directly guide the reasoning process, our approach focuses on finding a high-quality plan **before** the LLM begins to reason towards a solution. This corresponds to explicitly searching for a good sequence of reasoning steps (X) in the probabilistic framework $P(Y|X, C_{plan})$ discussed in Sec. 2.1.

Selection: Starting from the root node (which represents an initial plan, generated by prompting the LLM with the problem description), we traverse the tree by selecting the child node with the highest Upper Confidence Bound 1 (UCB1) value. UCB1 balances exploration and exploitation by considering both the average reward of simulations passing through a node and the number of times it has been visited:

$$UCB1(node) = Q(node) + C\sqrt{\frac{\ln(N(parent))}{N(node)}} \tag{3}$$

where $Q(node)$ is the average reward of simulations passing through the node, $N(node)$ is the number of times the node has been visited, $N(parent)$ is the number of times the parent node has been visited, and C is an exploration constant that controls the balance between exploration and exploitation.

Expansion: When a leaf node is reached, a new node is added to the tree. This new node represents a modified version of the parent node's plan.

Plans are represented as sequences of natural language instructions that guide the LLM's reasoning. For example, a plan to solve a math word problem might be: "1. Identify the given quantities. 2. Determine the relationship between the quantities. 3. Formulate an equation. 4. Solve the equation."

Simulation and Reward: The newly generated plan is then subjected to a simulation to estimate its effectiveness. Instead of real execution in traditional MCTS (Chaslot et al., 2008; Silver et al., 2016), we employ multi-agents to assess the quality of the plan inspired by (Zhang et al., 2024b), including:

- **Logical Consistency Agent:** Checks for contradictions or inconsistencies in the plan.
- **Feasibility Agent:** Determines whether the plan is executable.

Each evaluation agent assigns a score (e.g., between 0 and 1) to the plan, and may also provide textual feedback explaining its assessment. For example, the Logical Consistency Agent might give a low score and feedback like: "Step 2 contradicts the information given in Step 1." This feedback can be used to guide plan modification in subsequent expansion steps.

The individual scores from the evaluation agents are combined by a reward function to produce an overall reward signal for the plan. The reward function can be a simple weighted average or a more complex function that takes into account the relative importance of different evaluation criteria.

Backpropagation: The reward signal obtained from the simulation and evaluation is backpropagated up the MCTS tree, updating the value estimates of all nodes along the path from the root to the newly expanded node.

By iteratively applying these steps, MCTS builds a search tree of potential plans, gradually focusing on areas of the plan space that are likely to lead to high-quality solutions. Once a predefined number of rollouts have been completed, or a time limit is reached, the MCTS algorithm selects the node with the highest average reward as the optimal plan. We present an illustrative example of the MCTS planning process in Fig. 2 with counting “R”s in the word “Strawberry” and a more complex example in Sec. A.1.

3 EXPERIMENTS

3.1 RESEARCH QUESTIONS

In this research, we would like to investigatee the potential of MCTS to enhance the planning and problem-solving capabilities of LLMs, addressing limitations of existing methods like zero-shot CoT (Kojima et al., 2022) and Plan-and-Solve (Wang et al., 2023a) prompting in handling complex multi-step problems. Specifically, we explore two core research questions:

RQ1: *Does MCTS-based planning improve LLM problem-solving performance?* We will evaluate this by comparing the accuracy of our proposed approach against baseline prompting and analyzing the correlation between plan quality and LLM performance.

RQ2: *How to optimize the MCTS parameters for planning in LLMs?* We will investigate the impact of different MCTS strategy, such as different LLM for planning, evaluating, and executing, on the problem-solving performance of our approach.

By answering these questions, we aim to provide insights into the interplay between planning and reasoning in LLMs, contributing to the development of more robust and reliable AI systems capable of tackling complex real-world problems.

3.2 DATASET AND EVALUATION METRICS

Benchmark Datasets: To evaluate the effectiveness of our MCTS-enhanced planning approach, we conduct experiments on a diverse set of benchmark datasets specifically chosen to assess its performance across various problem-solving tasks and challenge the limitations of existing methods, including (1) arithmatic: GSM8K (Cobbe et al., 2021), AddSub (Hosseini et al., 2014), MultiArith (Roy & Roth, 2015), SVAMP (Patel et al., 2021), and SingleEq (Koncel-Kedziorski et al., 2015) (2) commonsense reasonin: CommonsensQA (Talmor et al., 2019) (3) symbolic reasoning: Last Letters (Wei et al., 2022) (4) Gaming reasoning: Object Tracking (Srivastava et al., 2023).

With these datasets, we aim to cover a wide range of problem-solving tasks, including arithmetic, commonsense reasoning, symbolic reasoning, and gaming reasoning, to evaluate the generalizability of our approach.

Evaluation Metrics: We employ accuracy as our primary evaluation metric across all datasets, measuring the percentage of correctly solved problems. Accuracy provides a direct measure of the LLM's problem-solving capabilities and allows for straightforward comparison between different approaches.

Table 1: Comparison of MCTS Planning and Zero-shot CoT on different datasets.

Type	Dataset / Model	AddSub	CommonsensQA	GSM8K	Last Letters	MultiArith	Object Tracking	SingleEq	SVAMP
Zero-shot CoT	Qwen2.5-7B-Instruct	85.06	63.72	80.89	21.00	95.33	74.80	77.17	83.40
((Kojima et al., 2022))	Meta-Llama-3.1-8B-Instruct	28.61	63.80	57.32	26.40	38.17	49.33	39.76	27.00
CoT Plan	Qwen2.5-0.5B-Instruct	36.96	31.20	17.82	0.00	37.00	31.87	44.09	29.90
((Wang et al., 2023a))	Qwen2.5-1.5B-Instruct	28.61	44.23	35.03	7.20	44.67	29.87	34.84	33.90
	Qwen2.5-7B-Instruct	87.59	78.62	88.84	55.20	98.33	79.33	93.70	91.90
	Meta-Llama-3.1-8B-Instruct	78.23	57.14	74.77	15.40	91.58	57.94	84.65	79.20
MCTS Plan (Ours)	Qwen2.5-0.5B-Instruct	58.23	32.43	29.80	0.20	68.33	27.20	69.69	43.80
	Qwen2.5-1.5B-Instruct	75.70	58.72	64.29	6.40	86.50	26.40	83.66	72.10
	Qwen2.5-7B-Instruct	88.10	79.20	90.14	56.60	98.67	79.33	92.91	92.90
	Meta-Llama-3.1-8B-Instruct	80.51	68.57	77.28	12.80	92.76	55.43	87.99	81.20
CoT Plan Avg.		57.85	52.80	54.12	19.45	67.89	49.75	64.32	58.73
MCTS Plan Avg.		75.63	59.73	65.38	19.00	86.56	47.09	83.56	72.50
Changes		+17.79	+6.93	+11.26	-0.45	+18.67	-2.66	+19.24	+13.77

3.3 RQ1: Enhanced Problem-Solving through MCTS-Guided Planning

Existing methods for enhancing LLM problem-solving often face challenges in maintaining logical consistency and handling long-term dependencies in complex multi-step problems. This is primarily because these methods rely heavily on the LLM's inherent reasoning capabilities, which can be limited in such scenarios. We hypothesize that applying MCTS to the planning process can address these limitations by generating higher-quality, more logically sound plans that guide the LLM towards more effective solutions. MCTS excels at exploring large search spaces and identifying optimal strategies through its balance of exploration and exploitation. By leveraging MCTS to generate plans, we aim to overcome the inherent limitations of relying solely on the LLM's reasoning for planning.

To investigate the impact of MCTS-generated plans on LLM problem-solving, we conduct experiments comparing the performance of our MCTS-enhanced planning approach against two baselines: (1) standard CoT prompting (Kojima et al., 2022) and (2) a plan-and-solve approach (Wang et al., 2023a) where the LLM first generates a plan and then executes it. We evaluate these methods on a diverse set of datasets introduced in Sec. 3.2.

We employ two state-of-the-art open-sourced LLMs: LLama 3.1 (8B parameters) (Dubey et al., 2024) and Qwen 2.5 (0.5B, 1.5B, and 7B parameters) (Yang et al., 2024). These models represent a strong baseline for current LLM capabilities and allow for a fair comparison between different planning approaches. Notably, we include smaller variants of Qwen 2.5 (0.5B and 1.5B) for MCTS to investigate the impact of model size on planning performance and computational efficiency, especially considering the potentially high computational cost of MCTS.

We adopt a zero-shot setting for both CoT and MCTS to assess the methods' ability to generalize to new problems without task-specific fine-tuning. For the CoT baseline, we use code and data from (Kong et al., 2024) and (Kojima et al., 2022). We leverage the SGLang platform (Zheng et al., 2024) for hosting and interacting with the LLMs.

Tab. 1 presents the comparison of MCTS Planning and Zero-shot CoT on different datasets. Overall, the MCTS-enhanced planning approach outperforms the CoT baseline across most datasets, with an average improvement of 40.59%. This substantial improvement highlights the effectiveness of MCTS in generating high-quality plans for LLMs. As expected, larger models generally perform better across all methods. However, the performance gap between small and large models is narrower for MCTS Planning compared to CoT, suggesting that our approach can partially compensate for the limitations of smaller models.

The improvements are particularly pronounced in tasks that require complex reasoning and long-term planning. For example, on arithmetic tasks (MultiArith, SingleEq, and AddSub), MCTS Planning shows average improvements of 18.67%, 19.24%, and 17.79%, respectively. Similarly, on problem-solving tasks (GSM8K and SVAMP), we observe improvements of 11.26% and 13.77%, respectively.

These results strongly support our hypothesis that MCTS can enhance LLM problem-solving by providing more structured and coherent plans that guide the reasoning process effectively. The

Table 2: Performance Comparison of Language Models

Model	Max Depth								Number of Rollouts					
	1	3	5	7	10	20	50	100	1	3	5	7	10	20
Meta-Llama-3.1-8B-Instruct	74.22	75.97	76.12	76.42	76.80	76.42	77.18	77.41	79.08	76.88	76.04	76.57	76.19	77.48
Qwen2.5-7B-Instruct	87.64	88.17	88.02	88.40	88.32	88.48	88.55	88.78	89.01	89.76	89.08	89.23	90.14	89.92

Table 3: Performance Cmparison of Different Evaluation Agents for MCTS.

Model	Evaluator	AddSub	CommonsensQA	GSM8K	Last Letters	MultiArith	Object Tracking	SingleEq	SVAMP
Qwen2.5-7B-Instruct	Feasibility	88.1	71.3	89.5	58.4	97.7	65.5	91.5	92.2
Qwen2.5-7B-Instruct	Logical Consistency	86.6	70.9	89.2	58.2	97.2	65.0	91.5	91.4
Qwen2.5-7B-Instruct	Combined (Ours)	88.1	79.2	90.1	56.6	98.7	79.3	92.9	92.9

superior performance in complex tasks suggests that MCTS is particularly adept at decomposing multi-step problems and maintaining logical consistency throughout the solution process.

To investigate the key factors influencing the effectiveness of MCTS in planning for LLMs, we conduct an ablation study to analyze the impact of different components of our approach. Specifically, we evaluate the following factors on the GSM8K dataset: (1) the maximum depth of the search tree, (2) the number of rollouts in MCTS, and (3) different evaluation agents for reward computation.

Tab. 2 illustrates the results of our ablation study. We observe that:

Depth: The performance of MCTS Planning generally improves with increasing depth, indicating that deeper search trees allow for more thorough exploration of the plan space and lead to higher-quality plans. The rate of improvement slows down as depth increases, suggesting that the plan can be easily optimized in the first few steps, while later steps are more difficult to optimize. This behavior implies that the MCTS may early-stop the search when evaluator agents give high scores to the plan.

Rollouts: The number of rollouts also impacts the performance of MCTS Planning. Initially, increasing the number of rollouts from a small number yields significant improvements. However, the uncertainty of performance improvement increases as the number of rollouts grows, indicating that with an expanding search space and higher computational cost, the performance gains may diminish. We implemented backpropagate in a zero-sum game manner, which may contribute to the increased uncertainty in performance improvement. Future work could explore alternative improvement strategies to address this issue.

Evaluation Agents: We tested different evaluation agents for computing rewards in the MCTS process. Tab. 3 shows the results of using feasibility and logical consistency evaluators compared to our final approach. While both evaluators improved performance over the baseline, our combined approach yielded the best results across most datasets.

These results demonstrate the effectiveness of our MCTS-guided planning approach in enhancing the problem-solving capabilities of LLMs. By generating high-quality plans through MCTS, we enable LLMs to tackle complex reasoning tasks more effectively, maintaining logical consistency and coherence throughout the problem-solving process. However, it is important to note that the computational cost of MCTS may be a limiting factor in some applications, and future work should explore ways to optimize the algorithm for efficiency without sacrificing performance.

3.4 RQ2: Enhancing the Efficiency of MCTS in Planning for LLMs

As demonstrated in Sec. 3.3, MCTS-based planning can significantly improve the problem-solving performance of LLMs. However, the computational cost of MCTS poses challenges for real-world applications, particularly when using large models for complex problem-solving tasks. Interestingly, our observations from Tab. 1 suggest that smaller models can benefit substantially from MCTS planning. This finding presents a promising direction for optimizing the efficiency of MCTS in planning for LLMs. We therefore pose the following research question: **Can we leverage small LLMs for optimal plan search and large LLMs for plan execution to enhance the efficiency and effectiveness of MCTS-based planning for LLMs?**

Table 4: Performance Comparison of Different LLMs for Planning and Execution.

Planner Model	Evaluator Model	Dataset Executor Model	AddSub	CommonsensQA	GSM8K	Last Letters	MultiArith	Object Tracking	SingleEq	SVAMP
Qwen2.5-1.5B-Instruct	Qwen2.5-1.5B-Instruct	Qwen2.5-1.5B-Instruct	75.70	58.72	64.29	6.40	86.50	26.40	83.66	72.10
gemma-2-2b-it	gemma-2-2b-it	gemma-2-2b-it	81.52	17.16	52.67	4.47	86.83	20.00	86.02	69.40
Qwen2.5-1.5B-Instruct	Qwen2.5-1.5B-Instruct	Qwen2.5-72B-Instruct	88.86	78.49	86.96	47.20	96.83	80.40	95.28	90.70
	Qwen2.5-72B-Instruct	Qwen2.5-1.5B-Instruct	86.58	68.36	81.35	28.20	92.50	47.12	88.78	81.40
	Qwen2.5-72B-Instruct	Qwen2.5-72B-Instruct	90.63	80.71	92.80	76.80	98.67	89.47	94.88	92.00
gemma-2-2b-it	gemma-2-2b-it	Qwen2.5-72B-Instruct	91.39	77.72	88.48	69.40	97.67	83.87	95.67	92.30
	Qwen2.5-72B-Instruct	gemma-2-2b-it	90.89	34.64	79.83	54.60	95.00	35.87	92.13	84.00
	Qwen2.5-72B-Instruct	Qwen2.5-72B-Instruct	92.41	78.54	92.42	78.20	98.33	80.53	95.87	93.00
Qwen2.5-72B-Instruct	Qwen2.5-72B-Instruct	Qwen2.5-72B-Instruct	91.14	83.95	94.62	85.60	98.67	97.86	95.08	93.40

To investigate this question, we designed an experimental setup using two small LLMs (Qwen2.5-1.5B-Instruct and Gemma-2-2b-it (Team et al., 2024)) for plan search, and a large LLM (Qwen2.5-72B-Instruct) for plan evaluation or execution. We evaluated this approach across the benchmark datasets introduced in Sec. 3.2, comparing various combinations of small and large models for planning, evaluation, and execution tasks.

Tab. 4 presents a comprehensive performance comparison of different LLM combinations for planning and execution tasks. Our key findings include:

Small-Large Model Synergy: Using a small model for planning and a large model for execution significantly improved problem-solving performance across all datasets. For instance, the Qwen2.5-1.5B-Instruct (planner) + Qwen2.5-72B-Instruct (evaluator+executor) combination achieved an average improvement of 23.87% compared to using Qwen2.5-1.5B-Instruct alone.

Efficiency Gains: The small-large model combination approached the performance of the large model (Qwen2.5-72B-Instruct) used alone, while potentially offering significant computational savings during the planning phase. For example, on the GSM8K dataset, the small-large combination achieved 92.80% accuracy, compared to 94.62% for the large model alone.

Model-Specific Performance: Interestingly, the Gemma-2-2b-it model, despite its smaller size, showed competitive performance when used for planning. This suggests that model architecture and training, not just size, play crucial roles in planning effectiveness.

Our findings demonstrate that leveraging small LLMs for planning and large LLMs for execution can significantly enhance the efficiency of MCTS-based planning while maintaining high problem-solving performance. This approach offers several advantages:

1. **Computational Efficiency:** By using smaller models for the computationally intensive planning phase, we can reduce the overall computational requirements without significantly sacrificing performance.

2. **Scalability:** This method allows for more efficient scaling of MCTS-based planning to larger and more complex problem spaces.

3. **Resource Optimization:** Organizations can optimize their use of computational resources, potentially reducing costs and environmental impact.

However, it's important to note potential limitations, such as the need for careful model selection and the possibility of suboptimal plans due to the use of smaller models in the planning phase. Future work should investigate these trade-offs more thoroughly and explore techniques to mitigate potential drawbacks.

In conclusion, our research demonstrates a promising approach to enhancing the efficiency and effectiveness of MCTS-based planning for LLMs, opening new avenues for improving AI problem-solving capabilities in resource-constrained environments.

4 Related Work

4.1 Chain-of-Thought Prompting

Chain-of-Thought (CoT) prompting has emerged as a prominent technique to improve the reasoning abilities of LLMs by encouraging step-by-step problem-solving (Wei et al., 2022). By decomposing complex tasks into smaller, manageable steps, CoT aims to enhance logical consistency and reduce errors in multi-step reasoning. Subsequent studies have explored various enhancements to CoT, such as self-consistency approaches (Wang et al., 2023b) and tree-of-thought methods (Yao et al., 2023), which attempt to maintain logical coherence over extended reasoning paths. Despite these advancements, CoT methods often struggle with maintaining overall logical flow in highly complex scenarios, leading to inconsistencies and suboptimal solutions. How to maintain logical consistency and coherence in multi-step reasoning tasks is a research frontier in the field.

4.2 Planning Techniques for LLMs

Beyond CoT, several planning-based approaches have been proposed to bolster the problem-solving capabilities of LLMs. Task decomposition techniques (Patel et al., 2022; Zhou et al., 2023) involve breaking down complex problems into simpler sub-tasks, which the LLM can solve sequentially. Explicit plan-and-solve frameworks (Wang et al., 2023a; Besta et al., 2024) require the LLM to generate a plan before executing it, aiming to structure the reasoning process more effectively. Advanced methods like Skeleton-of-Thoughts (Ning et al., 2024) and Graph-of-Thought (Besta et al., 2024) introduce more sophisticated representations of plans to capture dependencies and improve coherence. However, these methods remain constrained by the LLM's inherent reasoning limitations, often failing to produce optimal plans in the face of complex, multi-step problems. How to generate high-quality plans for LLMs remains a challenging research question in the field.

4.3 Monte Carlo Tree Search

MCTS is a heuristic search algorithm renowned for its success in game-playing AI, particularly in games with vast search spaces like Go and Chess (Silver et al., 2016). MCTS operates through four main phases: selection, expansion, simulation, and backpropagation (Chaslot et al., 2008). Its ability to balance exploration and exploitation makes it highly effective in navigating large decision trees to identify optimal strategies. Recent research has begun to explore the application of MCTS in the context of LLMs, particularly for solution search in CoT processes (OpenAI, 2024) or self-training process (Zhang et al., 2024a). These studies have demonstrated that MCTS can enhance the search for high-quality solutions by efficiently exploring the space of possible reasoning paths. However, the direct application of MCTS to LLMs presents challenges, such as the computational expense of evaluating generated content and the difficulty in effectively integrating heuristic evaluations within the planning process. We investigate these challenges in our research and propose to use smaller LLM for planning and larger LLM for evaluation and execution to address these challenges.

4.4 Research Gap and Our Contribution

While MCTS has shown promise in enhancing solution search for LLMs, its application to the planning process remains largely unexplored. Existing planning methods for LLMs, including Chain-of-Thought prompting and plan-and-solve frameworks, predominantly rely on the LLM's inherent reasoning abilities to generate plans. However, these reasoning abilities can be inconsistent and prone to errors, particularly when dealing with complex, multi-step problems requiring long-term planning and logical consistency. For instance, in tasks involving multi-step mathematical reasoning or intricate commonsense scenarios, LLMs often struggle to devise and maintain a coherent plan, leading to suboptimal or incorrect solutions.

Our research addresses this gap by integrating MCTS into the planning phase. Unlike previous work focusing on solution search, we leverage MCTS to explicitly generate high-quality plans before the LLM starts reasoning. By systematically exploring the plan space and evaluating quality with specialized LLM agents, we aim to generate more effective and logically consistent plans, significantly enhancing LLM problem-solving accuracy and reliability in complex tasks.

5 Conclusion and Discussion

This paper introduced a novel framework for enhancing the problem-solving capabilities of LLMs by leveraging MCTS for plan generation. Our approach explicitly separates the planning and reasoning stages of problem-solving, using MCTS to explore the space of possible plans before the LLM begins to reason towards a solution. Experiments on a diverse set of benchmark datasets demonstrated that MCTS-enhanced planning significantly improves LLM problem-solving accuracy compared to standard Chain-of-Thought prompting and existing plan-and-solve methods.

Our findings highlight the effectiveness of MCTS in generating high-quality, logically consistent plans that guide the LLM towards more effective solutions, particularly in complex reasoning tasks. We also observed that MCTS planning offers a stronger performance boost for smaller LLMs, suggesting it can partially compensate for limited reasoning capabilities in resource-constrained settings. Further analysis revealed that the performance of MCTS planning is influenced by factors such as search depth and the number of rollouts, with deeper search and more rollouts generally leading to better plans.

To address the computational cost of MCTS, we explored the use of smaller LLMs for planning and larger LLMs for evaluation and execution. Our results demonstrated that this approach can significantly enhance efficiency while maintaining high problem-solving performance, offering a promising direction for optimizing MCTS planning in real-world applications.

We believe that our research contributes to a deeper understanding of the interplay between planning and reasoning in LLMs and opens new avenues for improving AI problem-solving capabilities. The integration of MCTS and LLMs holds significant promise for developing more robust and reliable AI systems capable of tackling complex real-world problems.

Ethical Considerations

This research enhances LLM problem-solving via MCTS planning. While our benchmark datasets pose minimal direct ethical concerns, the potential impact of improved AI problem-solving necessitates broader ethical consideration.

Integrating MCTS introduces unique challenges. Unlike CoT methods, MCTS explores a wider range of plans, some potentially ethical yet logically sound. For instance, directly plan and executing a toxic action could be rejected by LLM with alignment to ethical principles. However, if the search objective is to find such a plan, MCTS may escape the LLM's ethical constraints. In this context, Mitigation strategies warrant investigation. Incorporating fairness constraints into MCTS, adversarial training for agents, and human-in-the-loop plan review could enhance ethical outcomes.

Limitations

While our MCTS-enhanced planning approach demonstrates significant improvements in LLM problem-solving capabilities, it is important to acknowledge several limitations of our current work.

Firstly, the computational cost of MCTS, particularly for larger language models, remains a significant challenge. Although we have shown that using smaller models for planning can mitigate this issue to some extent, further research is needed to optimize the efficiency of MCTS in the context of LLMs. Future work could explore pruning techniques or more sophisticated heuristics to reduce the search space without compromising plan quality.

Secondly, our study primarily focused on a specific set of benchmark datasets. While these datasets cover a range of problem types, they may not fully represent the diversity of real-world problems that LLMs might encounter. Expanding our evaluation to a broader set of tasks and domains would provide a more comprehensive assessment of our method's generalizability and robustness.

Despite these limitations, we believe that our work represents a significant step forward in enhancing the problem-solving capabilities of LLMs. By explicitly addressing these challenges, we hope to inspire further research that will lead to even more powerful and reliable AI systems capable of tackling complex real-world problems.

REFERENCES

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, et al. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 17682–17690, 2024.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020. URL `https://arxiv.org/abs/2005.14165`.

Guillaume Chaslot, Sander Bakkes, Istvan Szita, and Pieter Spronck. Monte-carlo tree search: A new framework for game ai. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 4, pp. 216–217, 2008.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240): 1–113, 2023.

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

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaoqing Ellen Tan, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aaron Grattafiori, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alex Vaughan, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Franco, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, Danny Wyatt, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Firat Ozgenel, Francesco Caggioni, Francisco Guzmán, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Govind Thattai, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Karthik Prasad, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kun Huang, Kunal Chawla, Kushal Lakhotia, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Maria Tsimpoukelli, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikolay Pavlovich Laptev, Ning Dong, Ning Zhang, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Rohan Maheswari, Russ Howes, Ruty Rinott, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Kohler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vítor Albiero, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaofang Wang, Xiaojian Wu, Xiaolan Wang, Xide Xia, Xilun Wu, Xinbo Gao, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yuchen Hao, Yundi Qian, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, and Zhiwei Zhao. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. Learning to solve arithmetic word problems with verb categorization. In Alessandro Moschitti, Bo Pang, and Walter Daelemans (eds.), *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 523–533, Doha, Qatar, October 2014. Association for Computational Linguistics. doi: 10.3115/v1/D14-1058. URL https://aclanthology.org/D14-1058.

Takeshi Kojima, Shixiang (Shane) Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 22199–22213. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/8bb0d291acd4acf06ef112099c16f326-Paper-Conference.pdf.

Rik Koncel-Kedziorski, Hannaneh Hajishirzi, Ashish Sabharwal, Oren Etzioni, and Siena Dumas Ang. Parsing algebraic word problems into equations. *Transactions of the Association for Computational Linguistics*, 3:585–597, 2015.

Aobo Kong, Shiwan Zhao, Hao Chen, Qicheng Li, Yong Qin, Ruiqi Sun, Xin Zhou, Enzhi Wang, and Xiaohang Dong. Better zero-shot reasoning with role-play prompting. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 4099–4113, 2024.

Xuefei Ning, Zinan Lin, Zixuan Zhou, Zifu Wang, Huazhong Yang, and Yu Wang. Skeleton-of-thought: Prompting LLMs for efficient parallel generation. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=mqVgBbNCm9.

OpenAI. Openai o1 system card, 2024. URL https://cdn.openai.com/o1-system-card.pdf.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, ukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Jan Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, ukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine B. Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. Gpt-4 technical report, 2024. URL `https://arxiv.org/abs/2303.08774`.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. Are nlp models really able to solve simple math word problems? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 2080–2094, 2021.

Pruthvi Patel, Swaroop Mishra, Mihir Parmar, and Chitta Baral. Is a question decomposition unit all we need? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 4553–4569, 2022.

Subhro Roy and Dan Roth. Solving general arithmetic word problems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 1743–1752, 2015.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R. Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, Agnieszka Kluska, Aitor Lewkowycz, Akshat Agarwal, Alethea Power, Alex Ray, Alex Warstadt, Alexander W. Kocurek, Ali Safaya, Ali Tazarv, Alice Xiang, Alicia Parrish, Allen Nie, Aman Hussain, Amanda Askell, Amanda Dsouza, Ambrose Slone, Ameet Rahane, Anantharaman S. Iyer, Anders Johan Andreassen, Andrea Madotto, Andrea Santilli, Andreas Stuhlmüller, Andrew M. Dai, Andrew La, Andrew Lampinen, Andy Zou, Angela Jiang, Angelica Chen, Anh Vuong, Animesh Gupta, Anna Gottardi, Antonio Norelli, Anu Venkatesh, Arash Gholamidavoodi, Arfa Tabassum, Arul Menezes, Arun Kirubarajan, Asher Mullokandov, Ashish Sabharwal, Austin Herrick, Avia Efrat, Aykut Erdem, Ayla Karakaş, B. Ryan Roberts, Bao Sheng Loe, Barret Zoph, Bartłomiej Bojanowski, Batuhan Özyurt, Behnam Hedayatnia, Behnam Neyshabur, Benjamin Inden, Benno Stein, Berk Ekmekci, Bill Yuchen Lin, Blake Howald, Bryan Orinion, Cameron Diao, Cameron Dour, Catherine Stinson, Cedrick Argueta, Cesar Ferri, Chandan Singh, Charles Rathkopf, Chenlin Meng, Chitta Baral, Chiyu Wu, Chris Callison-Burch, Christopher Waites, Christian Voigt, Christopher D Manning, Christopher Potts, Cindy Ramirez, Clara E. Rivera, Clemencia Siro, Colin Raffel, Courtney Ashcraft, Cristina Garbacea, Damien Sileo, Dan Garrette, Dan Hendrycks, Dan Kilman, Dan Roth, C. Daniel Freeman, Daniel Khashabi, Daniel Levy, Daniel Moseguí González, Danielle Perszyk, Danny Hernandez, Danqi Chen, Daphne Ippolito, Dar Gilboa, David Dohan, David Drakard, David Jurgens, Debajyoti Datta, Deep Ganguli, Denis Emelin, Denis Kleyko, Deniz Yuret, Derek Chen, Derek Tam, Dieuwke Hupkes, Diganta Misra, Dilyar Buzan, Dimitri Coelho Mollo, Diyi Yang, Dong-Ho Lee, Dylan Schrader, Ekaterina Shutova, Ekin Dogus Cubuk, Elad Segal, Eleanor Hagerman, Elizabeth Barnes, Elizabeth Donoway, Ellie Pavlick, Emanuele Rodolà, Emma Lam, Eric Chu, Eric Tang, Erkut Erdem, Ernie Chang, Ethan A Chi, Ethan Dyer, Ethan Jerzak, Ethan Kim, Eunice Engefu Manyasi, Evgenii Zheltonozhskii, Fanyue Xia, Fatemeh Siar, Fernando Martínez-Plumed, Francesca Happé, Francois Chollet, Frieda Rong, Gaurav Mishra, Genta Indra Winata, Gerard de Melo, Germán Kruszewski, Giambattista Parascandolo, Giorgio Mariani, Gloria Xinyue Wang, Gonzalo Jaimovitch-Lopez, Gregor Betz, Guy Gur-Ari, Hana Galijasevic, Hannah Kim, Hannah Rashkin, Hannaneh Hajishirzi, Harsh Mehta, Hayden Bogar, Henry Francis Anthony Shevlin, Hinrich Schuetze, Hiromu Yakura, Hongming Zhang, Hugh Mee Wong, Ian Ng, Isaac Noble, Jaap Jumelet, Jack Geissinger, Jackson Kernion, Jacob Hilton, Jaehoon Lee, Jaime Fernández Fisac, James B Simon, James Koppel, James Zheng, James Zou, Jan Kocon, Jana Thompson, Janelle Wingfield, Jared Kaplan, Jarema Radom, Jascha Sohl-Dickstein, Jason Phang, Jason Wei, Jason Yosinski, Jekaterina Novikova, Jelle Bosscher, Jennifer Marsh, Jeremy Kim, Jeroen Taal, Jesse Engel, Jesujoba Alabi, Jiacheng Xu, Jiaming Song, Jillian Tang, Joan Waweru, John Burden, John Miller, John U. Balis, Jonathan Batchelder, Jonathan Berant, Jörg Frohberg, Jos Rozen, Jose Hernandez-Orallo, Joseph Boudeman, Joseph Guerr, Joseph Jones, Joshua B. Tenenbaum, Joshua S. Rule, Joyce Chua, Kamil Kanclerz, Karen Livescu, Karl Krauth, Karthik Gopalakrishnan, Katerina Ignatyeva, Katja Markert, Kaustubh Dhole, Kevin Gimpel, Kevin Omondi, Kory Wallace Mathewson, Kristen Chiafullo, Ksenia Shkaruta, Kumar Shridhar, Kyle McDonell, Kyle Richardson, Laria Reynolds, Leo Gao, Li Zhang, Liam Dugan, Lianhui Qin, Lidia Contreras-Ochando, Louis-Philippe Morency, Luca Moschella, Lucas Lam, Lucy Noble, Ludwig Schmidt, Luheng He, Luis Oliveros-Colón, Luke Metz, Lütfi Kerem Senel, Maarten Bosma, Maarten Sap, Maartje Ter Hoeve, Maheen Farooqi, Manaal Faruqui, Mantas Mazeika, Marco Baturan, Marco Marelli, Marco Maru, Maria Jose Ramirez-Quintana, Marie Tolkiehn, Mario Giulianelli, Martha Lewis, Martin Potthast, Matthew L Leavitt, Matthias Hagen, Mátyás Schubert, Medina Orduna Baitemirova, Melody Arnaud, Melvin McElrath, Michael Andrew Yee, Michael Cohen, Michael Gu, Michael Ivanitskiy, Michael Starritt, Michael Strube, Michał Swędrowski, Michele Bevilacqua, Michihiro Yasunaga, Mihir Kale, Mike Cain, Mimee Xu, Mirac Suzgun, Mitch Walker, Mo Tiwari, Mohit Bansal, Moin Aminnaseri, Mor Geva, Mozhdeh Gheini, Mukund Varma T, Nanyun Peng, Nathan Andrew Chi, Nayeon Lee, Neta Gur-Ari Krakover, Nicholas Cameron, Nicholas Roberts, Nick Doiron, Nicole Martinez, Nikita Nangia, Niklas Deckers, Niklas Muennighoff, Nitish Shirish Keskar, Niveditha S. Iyer, Noah Constant, Noah Fiedel, Nuan Wen, Oliver Zhang, Omar Agha, Omar Elbaghdadi, Omer Levy, Owain Evans, Pablo Antonio Moreno Casares, Parth Doshi, Pascale Fung, Paul Pu Liang, Paul Vicol, Pegah Alipoormolabashi, Peiyuan Liao, Percy Liang, Peter W Chang, Peter Eckersley, Phu Mon Htut, Pinyu Hwang, Piotr Miłkowski, Piyush Patil, Pouya Pezeshkpour, Priti Oli, Qiaozhu Mei, Qing Lyu, Qinlang Chen, Rabin Banjade, Rachel Etta Rudolph, Raefer Gabriel, Rahel Habacker, Ramon Risco, Raphaël Millière, Rhythm Garg, Richard Barnes, Rif A. Saurous, Riku Arakawa, Robbe Raymaekers, Robert Frank, Rohan Sikand, Roman Novak, Roman Sitelew, Ronan Le Bras, Rosanne Liu, Rowan Jacobs, Rui Zhang, Russ Salakhutdinov, Ryan Andrew Chi, Seungjae Ryan Lee, Ryan Stovall, Ryan Teehan, Rylan Yang, Sahib Singh, Saif M. Mohammad, Sajant Anand, Sam Dillavou, Sam Shleifer, Sam Wiseman, Samuel Gruetter, Samuel R. Bowman, Samuel Stern Schoenholz, Sanghyun Han, Sanjeev Kwatra, Sarah A. Rous, Sarik Ghazarian, Sayan Ghosh, Sean Casey, Sebastian Bischoff, Sebastian Gehrmann, Sebastian Schuster, Sepideh Sadeghi, Shadi Hamdan, Sharon Zhou, Shashank Srivastava, Sherry Shi, Shikhar Singh, Shima Asaadi, Shixiang Shane Gu, Shubh Pachchigar, Shubham Toshniwal, Shyam Upadhyay, Shyamolima Shammie Debnath, Siamak Shakeri, Simon Thormeyer, Simone Melzi, Siva Reddy, Sneha Priscilla Makini, Soo-Hwan Lee, Spencer Torene, Sriharsha Hatwar, Stanislas Dehaene, Stefan Divic, Stefano Ermon, Stella Biderman, Stephanie Lin, Stephen Prasad, Steven Piantadosi, Stuart Shieber, Summer Misherghi, Svetlana Kiritchenko, Swaroop Mishra, Tal Linzen, Tal Schuster, Tao Li, Tao Yu, Tariq Ali, Tatsunori Hashimoto, Te-Lin Wu, Théo Desbordes, Theodore Rothschild, Thomas Phan, Tianle Wang, Tiberius Nkinyili, Timo Schick, Timofei Kornev, Titus Tunduny, Tobias Gerstenberg, Trenton Chang, Trishala Neeraj, Tushar Khot, Tyler Shultz, Uri Shaham, Vedant Misra, Vera Demberg, Victoria Nyamai, Vikas Raunak, Vinay Venkatesh Ramasesh, vinay uday prabhu, Vishakh Padmakumar, Vivek Srikumar, William Fedus, William Saunders, William Zhang, Wout Vossen, Xiang Ren, Xiaoyu Tong, Xinran Zhao, Xinyi Wu, Xudong Shen, Yadollah Yaghoobzadeh, Yair Lakretz, Yangqiu Song, Yasaman Bahri, Yejin Choi, Yichi Yang, Yiding Hao, Yifu Chen, Yonatan Belinkov, Yu Hou, Yufang Hou, Yuntao Bai, Zachary Seid, Zhuoye Zhao, Zijian Wang, Zijie J. Wang, Zirui Wang, and Ziyi Wu. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL `https://openreview.net/forum?id=uyTL5Bvosj`.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In Jill Burstein, Christy Doran, and

Thamar Solorio (eds.), *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4149–4158, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1421. URL https://aclanthology.org/N19-1421.

Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, Johan Ferret, Peter Liu, Pouya Tafti, Abe Friesen, Michelle Casbon, Sabela Ramos, Ravin Kumar, Charline Le Lan, Sammy Jerome, Anton Tsitsulin, Nino Vieillard, Piotr Stanczyk, Sertan Girgin, Nikola Momchev, Matt Hoffman, Shantanu Thakoor, Jean-Bastien Grill, Behnam Neyshabur, Olivier Bachem, Alanna Walton, Aliaksei Severyn, Alicia Parrish, Aliya Ahmad, Allen Hutchison, Alvin Abdagic, Amanda Carl, Amy Shen, Andy Brock, Andy Coenen, Anthony Laforge, Antonia Paterson, Ben Bastian, Bilal Piot, Bo Wu, Brandon Royal, Charlie Chen, Chintu Kumar, Chris Perry, Chris Welty, Christopher A. Choquette-Choo, Danila Sinopalnikov, David Weinberger, Dimple Vijaykumar, Dominika Rogoziska, Dustin Herbison, Elisa Bandy, Emma Wang, Eric Noland, Erica Moreira, Evan Senter, Evgenii Eltyshev, Francesco Visin, Gabriel Rasskin, Gary Wei, Glenn Cameron, Gus Martins, Hadi Hashemi, Hanna Klimczak-Pluciska, Harleen Batra, Harsh Dhand, Ivan Nardini, Jacinda Mein, Jack Zhou, James Svensson, Jeff Stanway, Jetha Chan, Jin Peng Zhou, Joana Carrasqueira, Joana Iljazi, Jocelyn Becker, Joe Fernandez, Joost van Amersfoort, Josh Gordon, Josh Lipschultz, Josh Newlan, Ju yeong Ji, Kareem Mohamed, Kartikeya Badola, Kat Black, Katie Millican, Keelin McDonell, Kelvin Nguyen, Kiranbir Sodhia, Kish Greene, Lars Lowe Sjoesund, Lauren Usui, Laurent Sifre, Lena Heuermann, Leticia Lago, Lilly McNealus, Livio Baldini Soares, Logan Kilpatrick, Lucas Dixon, Luciano Martins, Machel Reid, Manvinder Singh, Mark Iverson, Martin Görner, Mat Velloso, Mateo Wirth, Matt Davidow, Matt Miller, Matthew Rahtz, Matthew Watson, Meg Risdal, Mehran Kazemi, Michael Moynihan, Ming Zhang, Minsuk Kahng, Minwoo Park, Mofi Rahman, Mohit Khatwani, Natalie Dao, Nenshad Bardoliwalla, Nesh Devanathan, Neta Dumai, Nilay Chauhan, Oscar Wahltinez, Pankil Botarda, Parker Barnes, Paul Barham, Paul Michel, Pengchong Jin, Petko Georgiev, Phil Culliton, Pradeep Kuppala, Ramona Comanescu, Ramona Merhej, Reena Jana, Reza Ardeshir Rokni, Rishabh Agarwal, Ryan Mullins, Samaneh Saadat, Sara Mc Carthy, Sarah Perrin, Sébastien M. R. Arnold, Sebastian Krause, Shengyang Dai, Shruti Garg, Shruti Sheth, Sue Ronstrom, Susan Chan, Timothy Jordan, Ting Yu, Tom Eccles, Tom Hennigan, Tomas Kocisky, Tulsee Doshi, Vihan Jain, Vikas Yadav, Vilobh Meshram, Vishal Dharmadhikari, Warren Barkley, Wei Wei, Wenming Ye, Woohyun Han, Woosuk Kwon, Xiang Xu, Zhe Shen, Zhitao Gong, Zichuan Wei, Victor Cotruta, Phoebe Kirk, Anand Rao, Minh Giang, Ludovic Peran, Tris Warkentin, Eli Collins, Joelle Barral, Zoubin Ghahramani, Raia Hadsell, D. Sculley, Jeanine Banks, Anca Dragan, Slav Petrov, Oriol Vinyals, Jeff Dean, Demis Hassabis, Koray Kavukcuoglu, Clement Farabet, Elena Buchatskaya, Sebastian Borgeaud, Noah Fiedel, Armand Joulin, Kathleen Kenealy, Robert Dadashi, and Alek Andreev. Gemma 2: Improving open language models at a practical size, 2024. URL https://arxiv.org/abs/2408.00118.

Lei Wang, Wanyu Xu, Yihuai Lan, Zhiqiang Hu, Yunshi Lan, Roy Ka-Wei Lee, and Ee-Peng Lim. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2609–2634, 2023a.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*, 2023b. URL https://openreview.net/forum?id=1PL1NIMMrw.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 24824–24837. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/9d5609613524ecf4f15af0f7b31abca4-Paper-Conference.pdf.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jianxin Yang, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Keqin Chen, Kexin Yang, Mei Li, Mingfeng Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Xuejing Liu, Yang Fan, Yang Yao, Yichang Zhang, Yu Wan, Yunfei Chu, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, Zhifang Guo, and Zhihao Fan. Qwen2 technical report, 2024. URL https://arxiv.org/abs/2407.10671.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 11809–11822. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/271db9922b8d1f4dd7aaef84ed5ac703-Paper-Conference.pdf.

Dan Zhang, Sining Zhoubian, Ziniu Hu, Yisong Yue, Yuxiao Dong, and Jie Tang. Rest-mcts*: Llm self-training via process reward guided tree search, 2024a. URL https://arxiv.org/abs/2406.03816.

Lunjun Zhang, Arian Hosseini, Hritik Bansal, Mehran Kazemi, Aviral Kumar, and Rishabh Agarwal. Generative verifiers: Reward modeling as next-token prediction, 2024b. URL https://arxiv.org/abs/2408.15240.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E. Gonzalez, Clark Barrett, and Ying Sheng. Sglang: Efficient execution of structured language model programs, 2024. URL https://arxiv.org/abs/2312.07104.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc V Le, and Ed H. Chi. 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.

A Appendix

A.1 Count Characters: Example

Count Characters: Example 2.1

Problem: how many R's are there in the word "Pneumonoultramicroscopicsilicovolcanoconiosis"?
—
CoT Prompt: Let's think step by step.
CoT Result:
1. **Identify the letter:** We're looking for the letter 'R'.
2. **Scan the word:** Carefully read through the word "Pneumonoultramicroscopicsilicovolcanoconiosis".
3. **Count the occurrences:** Count each instance of the letter 'R'. There is only one 'R'.
Therefore, there is only **one** 'R' in the word "Pneumonoultramicroscopicsilicovolcanoconiosis".
— **Planning Prompt:**
To count the R's in "Pneumonoultramicroscopicsilicovolcanoconiosis." we can iterate throuoh each letter of the word, Initially, we'll set a counter to zero, for each leter encountered, we'll check if itis an "R" character. If it is, we'll increment our counter, This proess will continwe until we've examined all letters in the word, allowing us to determine the final count of "R's present.
Planning Result:
Following the plan:
1. **Initialization:** Set a counter to zero (e.g., ‘r_count = 0‘).
2. **Iteration:** Iterate through each letter of the word "Pneumonoultramicroscopicsilicovolcanoconiosis".
3. **Comparison:** For each letter, check if it is equal to "R".
4. **Increment:** If the letter is "R", increment the counter (‘r_count = r_count + 1‘).
5. **Final Count:** After iterating through all letters, the value of ‘r_count‘ will be the total number of "R"s.
Let's apply this to the word:
The word "Pneumonoultramicroscopicsilicovolcanoconiosis" contains **one** "R".

Figure 3: Count Characters Example of "R" in the word "Pneumonoultramicroscopicsilicovolcanoconiosis" with CoT and Planning.

Count Characters: Example 2.2

MCTS Planning Prompt:
1. **Define the task:** Our goal is to count the number of "R's present within the word "strawberry."
2. **Iterate through the word:** We will examine each letter of "strawberry" one at a time.
3. **Check for "R":** If the letter is an "R", increment a counter to keep track of our count.
4. **Increment counter:** If the letter is an "R", increment a counter to keep track of our count.
5. **Complete iteration:** We will continue this process until we have examined every letter in the word.
6. **Return the count:** After processing all letters, the counter will hold the final number of "R's in the word.
MCTS Planning Result:
Okay, let's adapt your plan to count the R's in "Pneumonoultramicroscopicsilicovolcanoconiosis":
Define the task: Our goal is to count the number of "R's" present within the word "Pneumonoultramicroscopicsilicovolcanoconiosis".
Iterate through the word: We will examine each letter of "Pneumonoultramicroscopicsilicovolcanoconiosis" one at a time.
Let's start with a counter set to 0.
* **P** - Not an R, counter remains 0.
* **n** - Not an R, counter remains 0.
* **e** - Not an R, counter remains 0.
* **u** - Not an R, counter remains 0.
* **m** - Not an R, counter remains 0.
* **o** - Not an R, counter remains 0.
* **n** - Not an R, counter remains 0.
* **o** - Not an R, counter remains 0.
* **u** - Not an R, counter remains 0.
* **l** - Not an R, counter remains 0.
* **t** - Not an R, counter remains 0.
* **r** - **It's an R! Increment the counter to 1.**
* **a** - Not an R, counter remains 1.
* **m** - Not an R, counter remains 1.
* **i** - Not an R, counter remains 1.
* **c** - Not an R, counter remains 1.
* **r** - **It's an R! Increment the counter to 2.**
* **o** - Not an R, counter remains 2.
* **s** - Not an R, counter remains 2.
* **i** - Not an R, counter remains 2.
* **l** - Not an R, counter remains 2.
* **i** - Not an R, counter remains 2.
* **c** - Not an R, counter remains 2.
* **o** - Not an R, counter remains 2.
* **v** - Not an R, counter remains 2.
* **o** - Not an R, counter remains 2.
SKIP FOR SPACE

Complete iteration: We have now processed all letters, and the counter holds the final number of "R's" in the word.
Result: There are **2** R's in the word "Pneumonoultramicroscopicsilicovolcanoconiosis".

Figure 4: Count Characters Example of "R" in the word "Pneumonoultramicroscopicsilicovolcanoconiosis" with MCTS Planning.