Optimizing Instruction Synthesis: Effective Exploration of Evolutionary Space with Tree Search

Anonymous ACL submission

Abstract

Instruction tuning is a crucial technique for aligning language models with humans' actual goals in the real world. Extensive research has highlighted the quality of instruction data is essential for the success of this alignment. However, creating high-quality data manually is labor-intensive and time-consuming, which leads researchers to explore using LLMs to synthesize data. Recent studies have focused on using a stronger LLM to iteratively enhance existing instruction data, showing promising results. Nevertheless, previous work often lacks control over the evolution direction, resulting in high uncertainty in the data synthesis process and low-quality instructions. In this paper, we introduce a general and scalable framework, IDEA-MCTS (Instruction Data Enhancement using Monte Carlo Tree Search), a scalable framework for efficiently synthesizing instructions. With tree search and evaluation models, it can efficiently guide each instruction to evolve into a high-quality form, aiding in instruction fine-tuning. Experimental results show that IDEA-MCTS significantly enhances the seed instruction data, raising the average evaluation scores of quality, diversity, and complexity from 2.19 to 3.81. Furthermore, in open-domain benchmarks, experimental results show that IDEA-MCTS improves the accuracy of real-world instruction-following skills in LLMs by an average of 5% in low-resource settings.

1 Introduction

Large language models (LLMs) have exhibited remarkable capabilities across a wide range of tasks in the field of natural language processing (NLP) (Brown et al., 2020; Kojima et al., 2022; Wei et al., 2022; Ouyang et al., 2022; Touvron et al., 2023; Jiang et al., 2023; OpenAI, 2023). Notably, LLMs can be trained to enhance their instruction-following skills through various methods, including fine-tuning on human-annotated data (Ouyang et al., 2022; Zhou et al., 2023b; Touvron et al., 2023) or extracted knowledge from stronger LLMs (Wang et al., 2022; Xu et al., 2023a; Zhao et al., 2023; Xu et al., 2023a,b; Wang et al., 2024). Zhou et al. (2023b) have demonstrated that this alignment can be achieved with low-resource 1k data. However, acquiring such data through human annotation remains high-cost, thus limiting further progress.

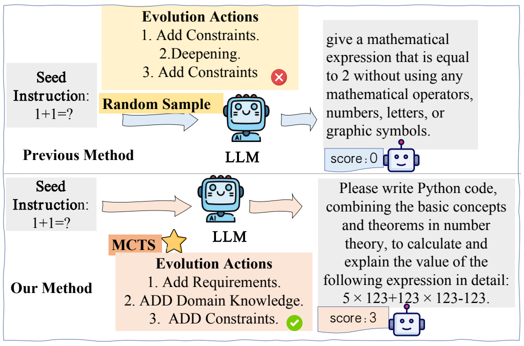

Figure 1: Iteratively enhance seed instructions using LLMs: The prior method's random sampling instruction evolution led to a perplexing instruction by selecting "Add constraints" multiple times. Our method uses MCTS to find suitable prompts, resulting in high-value instructions that align the language model to effectively learn multiple skills.

Recent work explores synthesizing instruction data with LLMs by prompting them with example data or prompts and iteratively enhancing the instruction data, offering an efficient and cost-effective alternative to human annotation (Xu et al., 2023a; Luo et al., 2023b,a; Liu et al., 2023). They introduced evolution prompts for LLMs, such as "Add constraints", "Increase reasoning" and "Complete input.", enabling LLMs to iteratively improve seed instructions. However, the process suffers high uncertainty due to the limited evolution prompts, random selection methods, and lack of control over the evolution direction. Specifically,

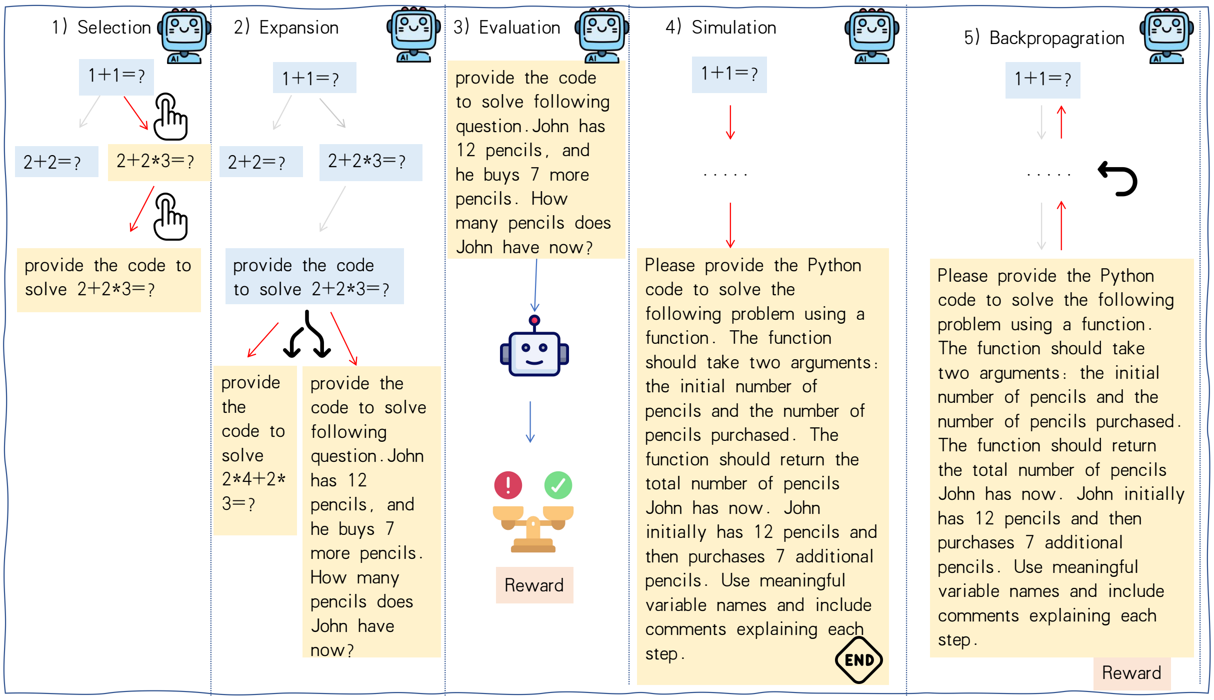

Figure 2: Framework of MCTS for instruction synthesis: 1. **Selection**: Choose high-value leaf nodes. 2. **Expansion**: Evolve the selected leaf nodes to generate new nodes. 3. **Evaluation**: Assess the current node to determine a reward. 4. **Simulation**: Randomly evolve the current instruction to a terminal state. 5. **Backpropagation**: Propagate the terminal state's reward back through the path's nodes.

failures occur when LLMs select inappropriate evolution prompts or fail to halt the instruction synthesis process appropriately. As shown in Figure 1, randomly selecting the prompts can turn a seed instruction like "1+1=" into a perplexing instruction. Language models will struggle to learn new skills from these low-value instructions, as humans also find them difficult to understand. Conversely, a few high-value instructions can significantly enhance the model's skills, enabling it to solve real-world problems.

Intuitively, simple seed instructions can evolve into a wide variety of forms during the evolutionary process. To efficiently optimize and control this evolution, we introduce a novel framework, IDEA-MCTS, which expands the evolution prompts as the action space and incorporates a tree search algorithm to iteratively enhance seed instruction data. In MCTS, each seed instruction acts as the root node. High-value nodes are identified through selection and use evolution prompts for further expansion, followed by simulation and backtracking, to find an optimal evolution action space to enhance the instructions. In this process, we employ customizable evaluation models to assess the quality, diversity, and complexity of the nodes, effectively controlling the direction of instruction evolution. This framework enhances instruction data and provides a clearer understanding of the evolution process, as shown in the case analysis in Appendix D. Our experimental results show that IDEA-MCTS significantly enhances the seed instruction data and models fine-tuned on this enhanced data exhibit substantial improvements compared to previous methods. We believe this work provides clear guidance for instruction synthesis, aiding models in achieving data-efficient alignment and enhancing overall performance. The contributions of our work are as follows:

- To synthesize high-value instructions for enhancing language model skills, we propose IDEA-MCTS, a scalable framework that controls the direction of instruction evolution by expanding the evolution space and integrating evaluation models in tree search.

- To enhance the efficiency and accuracy of instruction evolution, we expand the existing limited evolutionary space in two ways: evolving general effective instructions from themselves, and evolving task-specific instructions by designing meta-prompts.

- We demonstrate the effectiveness of our framework by analyzing the generated instructions and fine-tuning open-source models, including LLaMA2, LLaMA3, Phi-3, and Mistral, across different seed datasets and tasks, achieving a 5% improvement over the previous random evolution method on the open-domain instruction-following benchmark.

2 Related Work

Data Synthesis for Instruction Tuning Instruction tuning (IT) is a crucial technique for enhancing the performance and alignment of LLMs (Taori et al., 2023; Chiang et al., 2023; Wang et al., 2023). Recent efforts have extended into open-domain IT, characterized by a wide range of formats and task types, driven by crowdsourced human-generated instruction-response pairs (Köpf et al.; Conover et al., 2023; Zhang et al., 2023a; Peng et al., 2023; Zhou et al., 2023b). However, the high cost of human annotation poses significant challenges (Zhang et al., 2023a). One promising solution for this limitation is the synthesis of instruction data with the help of stronger LLMs (Bai et al., 2022; OpenAI, 2023; Anil et al., 2023; Team, 2023). Yet, using LLM-generated data increases the risk of low-quality examples, highlighting the need for more focus on dataset refinement and enhancement. Some works (Chen et al., 2023; Lu et al., 2023; Liu et al., 2023) address this by prompting stronger LLMs to filter instruction data based on its quality, diversity, and complexity, serving as a form of refinement. However, this approach lacks the synthesis of new instruction, limiting the model's instruction-following capabilities, especially in low-resource scenarios where only a small amount of data is available. Other works (Zhao et al., 2024; Xu et al., 2023a) enhance existing seed instructions by using LLMs with carefully designed prompt templates. Zhao et al. (2024) enhanced the original instructions using tree-structured prompts but focused only on the complexity and heavily relies on LLMs' intrinsic knowledge. Additionally, some work (Xu et al., 2023a; Luo et al., 2023b,a; Liu et al., 2023) design a series of evolution prompts to iteratively guide LLMs in enhancing the seed instructions. However, random selection during instruction evolution introduces high uncertainty and affects the quality of generated instructions. To effectively enhance the seed instruction data, we propose IDEA-MCTS, which expands the evolution action space, introduces evaluation models and iteratively improves instruction data with MCTS.

Tree Search for LLM Enhancement Tree search methods such as BFS, A* search (Hart et al., 1968), and MCTS (Kocsis and Szepesvári, 2006; Coulom, 2006; Ye et al., 2021; Silver et al., 2016), are widely used to find an optimal state in a tree structure. Integrating tree-search methods with LLMs presents a novel approach to find an effective sequence of actions that leads to a favorable outcome. Effective search strategy is crucial for reasoning and planning (Hao et al., 2023; Zhou et al., 2023a; Hu et al., 2023). Depth/breadth-first search in (Yao et al., 2023), A* search in (Zhuang et al., 2023) and MCTS in (Zhang et al., 2023b; Yu et al., 2023; Hao et al., 2023; Zhou et al., 2023a; Chen et al., 2024b). Feng et al. (2023); Tian et al. (2024); Chen et al. (2024a) have utilized tree search for LLM self-improvement. Unlike previous approaches, we leverage the powerful generative capabilities of LLMs and MCTS for instruction synthesis.

3 Approach

In this section, we introduce the novel framework IDEA-MCTS, which enhances the quality, diversity, and complexity of seed instructions with a stronger LLM, using MCTS. We first define the problem, including the state, action space, and reward function. Then, we discuss the expansion of evolution prompts from two key aspects and the use of MCTS with LLM to efficiently explore the action spaces. Finally, we fine-tune models based on the instruction data generated by the LLM, proving the effectiveness of the framework in low-resource settings.

3.1 Problem Setting

We begin with a seed instruction sample x as the root node and employ a stronger language model p_θ. Our goal is to improve the quality, diversity, and complexity of x. To achieve this, we use evolution prompts, such as 'add constraints', as our action space. During the tree search, intermediate instructions generated by the LLM, denoted as z_t, serve as new nodes.

$$z_{t+1} = p_\theta(z_t, a) \tag{1}$$

By applying an action a, which is an evolution prompt to wrap the state z_t, we obtain the next instruction z_{t+1} via p_θ. We assess each intermediate instruction z_t based on its quality, diversity, and

Please rate according to the accuracy and quality . Score 1-5. You can give a score of 6 if the question is high quality. You should respond with the format: [1] Score: [2] Score:
[1] <Instruction 1> [2] <Instruction 2> [3] <Instruction 3> [4] <Instruction 4> [5] <Instruction 5>
(Quality)

Please rate according to the difficulty and complexity. Score 1-5. You can give a score of 6 if the question is too complex for you to answer it. You should respond with the format:[1] Score: [2] Score:
[1] <Instruction 1> [2] <Instruction 2>
(Complexity)

Please identify tags of user intentions in the following instruction and provide an explanation for each tag. Please response in the JSON format {"tag": str, "explanation": str}.
Instruction: <Instruction>
(Diversity)

Figure 3: Evaluation prompt used to assess the quality, complexity, and diversity of instructions. Instruction diversity is measured by the number of distinct intents.

complexity. The value $v(z_t)$ of an instruction is determined using the following equation:

$$v(z_t) = p_{\theta_q}(z_t) + p_{\theta_d}(z_t) + p_{\theta_c}(z_t) \quad (2)$$

In this equation, $p_{\theta_q}(z_t)$, $p_{\theta_d}(z_t)$, and $p_{\theta_c}(z_t)$ represent the quality, diversity, and complexity scores of the instruction z_i, respectively. Notably, instruction diversity is measured by the number of distinct intents. Further details about these value scores will be discussed in the following sections. By integrating these elements, we aim to create a framework that robustly enhances seed instructions.

Quality & Complexity & Diversity Following the (Liu et al., 2023; Lu et al., 2023), we continue training based on models, EVOL_QUALITY, EVOL_COMPLEXITY, and InsTagger from these works with 1k data points. We apply a random evolution method(Xu et al., 2023a) to create new data points from a base sample, gradually adjusting their complexity, quality, and diversity of instruction. We evaluate these data points using ChatGPT and train an automatic scoring model with LLaMA2-7B to predict ChatGPT's scores. The evaluation prompt we use is shown in Figure 3. These scoring models are used to assess the quality, complexity, and diversity of instructions as rewards in MCTS.

3.2 Instruction Evolution with MCTS

In our framework, we leverage a stronger language model p_θ and value function v to evolve the seed instruction x using MCTS, as shown in Figure 2.

Intuitively, more precise and diverse evolution prompts contribute to enhancing the quality of seed instructions. To achieve this, we first expand the evolution prompts from two ways, general effective and task-specific instructions. We explore the open-space evolution prompts, that contribute a general effective instructions such as goals, key constraints, and requirements (Xu et al., 2023a; Tianle Li*, 2024). On the other hand, we aim to ensure that the seed instructions can effectively transfer to task-specific contexts. With LLMs, we design the meta prompts, as shown in Figure 4, to extract task-related evolution prompts that contain the words "such as." As shown in Table 1, the designed evolution prompts can enhance both the depth and breadth of the seed instruction. We show more details about the evolution prompts in Appendix B.

Then we construct a decision tree. MCTS proceeds for k episodes, starting from the root (initial state) and progressively expanding this tree through two primary steps: Selection and Expansion. During **Selection**, the child with the highest Upper Confidence bounds applied to Trees (UCT) value (Kocsis and Szepesvári, 2006; Coulom, 2006) is chosen for the next iteration. The UCT of a child state z is computed as follows:

$$UCT(z) = V(z) + C \cdot \sqrt{\frac{\ln(N(p))}{N(z)}} \quad (3)$$

where $N(z)$ represents the number of visits to node z, and $V(z)$ is the value function (expected return). During **Expansion**, multiple child states z are explored from the current state p by sampling n actions. The child node with the highest UCT value is selected for expansion in the subsequent iteration.

Evolution Space	Description
Add Global and Local Goals	Add one or more global and local goals into the instruction to enhance its direction and purpose.
Add Key Constraints	Add one or more constraints where necessary to define its limitations and boundaries.
Add Task Requirements	Specify one or more detailed requirements to clarify the tasks within the instruction.
Add Problem-Solving Skills	Add one or more problem-solving task skills.
Add Reasoning Complexity	Increase complexity by adding one or more reasoning elements.
Add Domain Knowledge	Add one or more areas of domain-specific knowledge, such as medicine, law, finance, IT technology.
Add Life Topics	Add one or more life topics. Topics can range from health and nutrition, cooking, photography, music, and travel, to parenting.
Add Real-World Applications	Add one or more real-world applications to provide practical context and applicability, such as education, customer service, and Business.
Add Emotional Expression	Add one or more emotional content elements such as excitement or concern.
Format the Input Style	Define one or more input formatting styles, such as a doctor, teacher, or customer.
Format the Output Style	Specify one or more output formats, such as report format or summarized in paragraphs.
Refine the Factuality	Refine the instruction to make it more factual and clear, to ensure it is more factual, clear, and able to be responded to.
Create a New One	Create one instruction within the same domain to introduce fresh perspectives.

Table 1: Expanded Space for Instruction Evolution.

In **Evaluation**, we assess the quality, complexity, and diversity of the instruction data using the value function v, which serves as the node's reward. In **Simulation**, selection and expansion are performed repeatedly until a termination state is reached, constructing the rollout policy. The termination state occurs when the tree's depth or node value meets a specified threshold. **Backpropagation** is performed at the end of an episode: the return v is used to update every $V(z)$ along the path using the formula:

$$V(z) = V_{\text{old}}(z) \cdot \left(\frac{N(z)-1}{N(z)}\right) + \frac{v}{N(z)} \quad (4)$$

where $V_{\text{old}}(z)$ denotes the old value function.

MCTS relies on an environment model to reverse steps and build a search tree, imposing strict assumptions. This constraint does not apply to LLMs. Our method allows resetting to any step by copying historical text input, overcoming the limitation. By integrating MCTS with LLMs, we demonstrate how heuristic search algorithms can efficiently evolve instructions by leveraging the powerful generative capabilities of LLMs.

Finally, after evolving the seed instructions, we obtain responses from the stronger LLM and fine-tune the open-source model. To ensure clarity and logic, we avoided complex templates from previous works (Wei et al., 2021; Longpre et al., 2023). Instead, our method follows a straightforward instruction template (Taori et al., 2023).

4 Experiments

4.1 Experiments Setting

Baselines We compare our method with manually annotated data and other techniques for enhancing instructional data using a stronger LLM. We also present the baselines utilized in our experiments. **Seed** serves as the baseline without any enhancement methods. **LIMA** (Jha et al., 2023) contains 1k human-annotated high-quality instruc-

Method	Metric			
	Quality	Instag	Complexity	Average
Seed	3.58	1.60	1.40	2.19
Lima	3.58	1.99	2.09	2.55
Tree-instruct	4.37	2.40	2.44	3.07
WizardLM	3.82	2.30	2.52	2.87
WizardLM+	4.01	2.80	2.70	3.17
MCTS	3.96	3.12	3.51	3.53
MCTS+	**4.56**	**3.24**	**3.62**	**3.81**

Table 2: Statistics of instruction dataset. The "+" symbol indicates methods that expand the evolution prompts space.

tions data, showing notable improvement for LLMs. **Tree-instruct** (Zhao et al., 2024) enhances the complexity of instruction data by adding nodes to the semantic tree. **WizardLM** (Xu et al., 2023a; Luo et al., 2023a,b) stands out by prompting LLM to evolve instruction data randomly step by step.

Test Datasets Many studies have focused on assessing the capabilities of LLMs (Liang et al., 2022). However, the challenge remains unresolved. A prevalent method involves using the powerful language model as the evaluator (Li et al., 2023a; Zheng et al., 2024; Chiang et al., 2023; Chen et al., 2023).

In our framework, we employ two distinct methods to assess the model's capabilities: LLM evaluation and human evaluation. Specifically, we use Alpaca-Eval (Li et al., 2023b) and MT-Bench (Zheng et al., 2024) to assess real-world instruction-following capabilities. We show more details in Appendix A. In the Alpaca-Eval, we compare our model's output with Text-Davinci-003 and use GPT-3.5-turbo to evaluate and score the output in MT-Bench. Additionally, we evaluate the model's capabilities in the NLP benchmark with the Open-LLM Leaderboard, which comprises four tasks: ARC (Clark et al., 2018), HellaSwag (Zellers et al., 2019), GSM8K (Cobbe et al., 2021), and TruthfulQA (Lin et al., 2022).

Experiment Setting We randomly select 1,000 seed instructions each from Alpaca-52K (Taori et al., 2023) and Dolly (Conover et al., 2023) as a low-resource setting. We initialize our MCTS evolution process with the stronger LLM, GPT-3.5-turbo model. When calling this API, we define the temperature parameter to 0.7, set the maximum token limit to 2048, and apply no penalty. In the MCTS setup, we generate evolution prompts for the

> Your goal is to analyze the task-specific instructions and extract features that are essential for high-quality instructions in this task-specific context. Then Use these features to create evolution prompt as actions that can improve the seed instructions. Follow the steps below:
> 1. Identify Key Features: Analyze the task-specific instructions and list the key features of them.
> 2. Define Evolution Actions: Based on the identified features, define evolution prompt as action that can enhance the ##seed instructions.
> ##seed instructions:{seed instructions}
> ##task-specific instructions:{task-specific instructions}

Figure 4: Meta prompt used in LLM: Extracting evolutionary prompt actions from task-related contexts.

seed instructions based on the task-specific Alpaca-eval benchmark. The terminal state is defined as either reaching a depth of more than 4 or achieving a reward of more than 10. For each iteration, we expand 5 nodes per step, and the MCTS process is iterated 3 times. We randomly select 1,000 data points from the generated data, collected from paths between the root node and terminal state nodes, as training data for the low-resource setting. During tuning, the foundational models for our experiments are the LLaMA2-7B, Mistral-7B, LLaMA3-8B, and Phi-3. To efficiently fine-tune these models, we adopted the QLORA approach (Dettmers et al., 2023). Throughout the tuning process, we maintained a batch size of 32 and ended the process after a maximum of 800 training steps. It's important to note that these preliminary experiments were conducted on a single GPU with 48GB of memory. For technical execution, we harnessed the capabilities of HuggingFace Transformers, PyTorch, and Accelerate, ensuring strict adherence to academic integrity and standards throughout the entire process.

4.2 Statistical Analysis of the Data Evolved from MCTS

We conduct a comprehensive analysis of the evolved instruction data from three critical dimensions: quality, complexity, and diversity with the EVOL_QUALITY, EVOL_COMPLEXITY, and InsTagger. As shown in Table 2, the Seed contains 1,000 instructions selected from the Alpaca-52K. The WizardLM contains 1,000 instructions obtained through random evolution, while the MCTS

Model	Method	Metrics					
		help_base	koala	self_instruct	oasst	vicuna	overall
LLaMA2	Seed	39.53	49.36	38.10	55.85	45.00	45.59
	Lima	44.96	42.95	30.95	51.60	33.75	40.68
	Tree-instruct	47.29	50.64	46.43	62.23	40.00	50.50
	WizardLM	44.96	45.51	43.65	51.60	42.50	46.46
	MCTS+ (ours)	51.94	50.64	45.24	63.83	42.50	**51.61**
LLaMA3	Seed	58.14	54.49	51.98	64.89	47.50	56.27
	Lima	44.19	41.67	38.10	52.13	36.25	42.98
	Tree-instruct	53.49	57.05	55.95	67.55	52.50	58.21
	WizardLM	56.59	52.56	51.98	68.09	52.50	57.08
	MCTS+ (ours)	53.49	61.54	62.70	70.21	46.25	**60.37**
Phi-3	Seed	46.51	55.13	55.16	64.89	43.75	55.22
	Lima	41.86	50.64	41.27	60.11	43.75	48.36
	Tree-instruct	51.94	57.05	51.59	69.08	51.25	57.52
	WizardLM	51.94	55.77	51.19	68.09	46.25	56.09
	MCTS+ (ours)	57.36	57.05	59.52	72.34	65.00	**62.36**
Mistral	Seed	56.59	55.77	52.78	70.21	45.00	57.52
	Lima	41.86	48.08	42.86	53.72	36.25	45.59
	Tree-instruct	55.04	59.62	53.57	69.62	46.25	58.39
	WizardLM	57.36	55.77	52.78	70.21	41.25	57.52
	MCTS+ (ours)	61.24	60.90	58.33	72.87	58.75	**62.80**

Table 3: Results of different instruction-tuned models on Alpaca-Eval (%).

	MT-Bench		
	Turn-1	Turn-2	Average Score
Seed	6.25	5.54	5.90
Lima	6.71	6.61	6.66
Tree-instruct	6.55	6.03	6.29
WizardLM	6.63	6.45	6.54
MCTS	6.71	6.61	6.66
WizardLM+	6.56	7.14	6.69
MCTS+	6.74	7.14	**6.94**

Table 4: Results of different instruction-tuned models on MT-Bench.

contains 1,000 instructions obtained through the MCTS evolution. MCTS+ method can achieve the highest scores across all evaluation metrics (Liu et al., 2023; Lu et al., 2023), demonstrating significant improvement in quality, diversity, and complexity. It outperforms the Seed, with average scores increasing from 2.19 to 3.81. The expansion of the instruction evolution space proves to be a highly effective strategy for enhancing the quality of instruction data.

4.3 Main results

The main results presented below are based on LLM evaluations and further human evaluations are provided in Appendix C.

Table 3 demonstrates that models fine-tuned with data evolved from MCTS+ exhibit better performance compared to other fine-tuning methods. In particular, LLaMA2 and LLaMA3 can show significant gains with MCTS+, with improvements of 6.02% and 4.1%, respectively, over the Seed method. Furthermore, Phi-3 and Mistral fine-tuned with MCTS+ method outperform previous methods across various skills, including help_base, koala, self_instruct, oasst, and vicuna. Notably, the Mistral model achieves a win rate of 61.24% in help_base, surpassing the previous highest win rate by 3.88 obtained using the WizardLM method. Overall, Mistral exhibits a 5.28% enhancement in performance compared to the WizardLM method. These results show that MCTS effectively enhances models' instruction-following capabilities better than traditional methods. Additionally, fine-tuning with the LIMA method does not significantly improve the model's performance on Alpaca-eval, suggesting potential generalization limitations of manually annotated models.

4.4 Generalization

During the expanded evolution process, with a focus on task-specific instruction data features on Alpaca-Eval, we also evaluate the model's performance on the open-domain benchmark MT-Bench and assess its capabilities on the NLP benchmark, OpenLLM. Additionally, we consider the effectiveness of using Dolly as a seed dataset.

As shown in Table 4, the MCTS+ method enhances both the model's single-turn and multi-turn dialogue capabilities. The single-turn score is improved from 6.25 (Seed) to 6.74 (MCTS+), while

Model	Method	Metrics					
		help_base	koala	self_instruct	oasst	vicuna	overall
LLaMA2	Tree-instruct	51.16	51.28	60.11	42.46	43.75	49.81
	WizardLM	44.19	49.36	60.11	42.46	41.25	48.07
	MCTS	44.96	51.92	62.23	44.84	41.25	50.00
	WizardLM+	50.39	53.21	59.57	42.86	37.50	49.50
	MCTS+	49.61	51.92	63.30	44.84	42.50	**51.18**

Table 5: Results of different instruction-tuned models on Alpaca-Eval using Dolly as the Seed Dataset (%).

Model	Method	ARC-Easy	ARC-Challenge	HellaSwag	TruthfulQA	GSM8k	Average
LLaMA2	Seed	80.30	52.82	77.96	29.80	13.04	50.78
	Lima	80.43	53.07	78.54	31.18	14.10	51.59
	Tree-instruct	80.85	53.92	78.80	33.09	13.80	52.01
	WizardLM	80.81	54.10	78.81	32.51	13.50	51.95
	MCTS+	81.02	54.10	78.94	33.73	13.72	**52.30**

Table 6: Results of different instruction-tuned models on the NLP benchmark, OpenLLM (%).

the multi-turn score is increased from 4.54 (Seed) to 7.15 (MCTS+). This results in an overall average score improvement from 5.90 (Seed) to 6.94 (MCTS+), highlighting the method's effectiveness in handling more complex, multi-turn dialogues.

Using Dolly as the seed instruction dataset, Table 5 shows that the MCTS+ method can achieve the best performance, with a 3% improvement compared to the WizardLM method. Specifically, the overall score is improved from 48.07% (WizardLM) to 51.18% (MCTS+). In individual metrics, MCTS+ can improve the help_base from 44.19 to 49.61, koala from 49.36% to 51.92%, self_instruct from 60.11% to 63.30%, oasst from 42.46% to 44.84%, and vicuna from 41.25% to 42.50%.

As shown in Table 6, despite being fine-tuned on very different instruction-following prompts, the model's capabilities in NLP tasks show a slight improvement, with a 1.5% increase compared to the seed method.

4.5 Ablation experiment

Our method can be proved effective in two key areas: expanding the action space and using MCTS evolution. As shown in Table 7, models with expanded action space (denoted as + methods) consistently outperform those without it, regardless of using random or MCTS evolution. For example, the Mistral using the MCTS+ method shows a 3.72% improvement over the WizardLM method. Additionally, data evolved through MCTS maintains high quality, further improving the instruction-following abilities of the model. The Phi-3 model, using MCTS evolution, improves performance by 1.5% before action space expansion and by 2.92% after expansion.

Model	Method	Overall (%)
LLaMA2	WizardLM	46.46
	MCTS	47.89
	WizardLM+	49.32
	MCTS+	**51.61**
LLaMA3	WizardLM	57.08
	MCTS	57.52
	WizardLM+	58.12
	MCTS+	**60.37**
Phi-3	WizardLM	56.09
	MCTS	57.64
	WizardLM+	59.44
	MCTS+	**62.36**
Mistral	WizardLM	57.52
	MCTS	57.57
	WizardLM+	60.11
	MCTS+	**62.80**

Table 7: Ablation study results on Alpaca-Eval (%).

5 Conclusion

In this paper, we introduce a novel framework that leverages the power of MCTS combined with heuristic evaluation to synthesis high-value instruction data. Our statistical analysis validates the framework's effectiveness in synthesizing high-value data. By fine-tuning open-source models with these evolved instructions, models achieve competitive competitive performance compared to previous methods.

Limitations

We need to acknowledge that the process of using LLMs for evolving instructions with MCTS is opaque and incurs API costs. Knowledge distillation might balance the trade-off between expenses and synthesizing high-quality instructions. On the other hand, we have demonstrated the effectiveness of MCTS-evolved instructions under low-resource conditions. Further exploration of scaling laws could enhance our understanding of the framework.

References

Rohan Anil, Andrew M. Dai, Orhan Firat, Melvin Johnson, and Dmitry Lepikhin. 2023. Palm 2 technical report. *ArXiv*, abs/2305.10403.

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al. 2022. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Guoxin Chen, Minpeng Liao, Chengxi Li, and Kai Fan. 2024a. Alphamath almost zero: process supervision without process. *arXiv preprint arXiv:2405.03553*.

Lichang Chen, Shiyang Li, Jun Yan, Hai Wang, Kalpa Gunaratna, Vikas Yadav, Zheng Tang, Vijay Srinivasan, Tianyi Zhou, Heng Huang, et al. 2023. Alpagasus: Training a better alpaca with fewer data. *arXiv preprint arXiv:2307.08701*.

Ziru Chen, Michael White, Raymond Mooney, Ali Payani, Yu Su, and Huan Sun. 2024b. When is tree search useful for llm planning? it depends on the discriminator. *arXiv preprint arXiv:2402.10890*.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E Gonzalez, et al. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality. *See https://vicuna. lmsys. org (accessed 14 April 2023)*.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Mike Conover, Matt Hayes, Ankit Mathur, Jianwei Xie, Jun Wan, Sam Shah, Ali Ghodsi, Patrick Wendell, Matei Zaharia, and Reynold Xin. 2023. Free dolly: Introducing the world's first truly open instruction-tuned llm. *Company Blog of Databricks*.

Rémi Coulom. 2006. Efficient selectivity and backup operators in monte-carlo tree search. In *International conference on computers and games*, pages 72–83. Springer.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. Qlora: Efficient finetuning of quantized llms. *arXiv preprint arXiv:2305.14314*.

Xidong Feng, Ziyu Wan, Muning Wen, Ying Wen, Weinan Zhang, and Jun Wang. 2023. Alphazero-like tree-search can guide large language model decoding and training. *arXiv preprint arXiv:2309.17179*.

Xinyang Geng, Arnav Gudibande, Hao Liu, Eric Wallace, Pieter Abbeel, Sergey Levine, and Dawn Song. 2023. Koala: A dialogue model for academic research. Blog post.

Shibo Hao, Yi Gu, Haodi Ma, Joshua Jiahua Hong, Zhen Wang, Daisy Zhe Wang, and Zhiting Hu. 2023. Reasoning with language model is planning with world model. *arXiv preprint arXiv:2305.14992*.

Peter E Hart, Nils J Nilsson, and Bertram Raphael. 1968. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*, 4(2):100–107.

Mengkang Hu, Yao Mu, Xinmiao Yu, Mingyu Ding, Shiguang Wu, Wenqi Shao, Qiguang Chen, Bin Wang, Yu Qiao, and Ping Luo. 2023. Tree-planner: Efficient close-loop task planning with large language models. *arXiv preprint arXiv:2310.08582*.

Aditi Jha, Sam Havens, Jeremey Dohmann, Alex Trott, and Jacob Portes. 2023. Limit: Less is more for instruction tuning across evaluation paradigms. *arXiv preprint arXiv:2311.13133*.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b.

Levente Kocsis and Csaba Szepesvári. 2006. Bandit based monte-carlo planning. In *European conference on machine learning*, pages 282–293. Springer.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213.

Andreas Köpf, Yannic Kilcher, Dimitri von Rütte, Sotiris Anagnostidis, Zhi-Rui Tam, Keith Stevens, Abdullah Barhoum, Nguyen Minh Duc, Oliver Stanley, Richárd Nagyfi, et al. Openassistant conversations-democratizing large language model alignment. corr, abs/2304.07327, 2023. doi: 10.48550. *arXiv preprint arXiv.2304.07327*.

Xuechen Li, Tianyi Zhang, Yann Dubois, Rohan Taori, Ishaan Gulrajani, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. 2023a. Alpacaeval: An automatic evaluator of instruction-following models. *GitHub repository*.

Xuechen Li, Tianyi Zhang, Yann Dubois, Rohan Taori, Ishaan Gulrajani, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023b. Alpacaeval: An automatic evaluator of instruction-following models. https://github.com/tatsu-lab/alpaca_eval.

Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, et al. 2022. Holistic evaluation of language models. *arXiv preprint arXiv:2211.09110*.

Stephanie Lin, Jacob Hilton, and Owain Evans. 2022. Truthfulqa: Measuring how models mimic human falsehoods. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3214–3252.

Wei Liu, Weihao Zeng, Keqing He, Yong Jiang, and Junxian He. 2023. What makes good data for alignment? a comprehensive study of automatic data selection in instruction tuning. *arXiv preprint arXiv:2312.15685*.

Shayne Longpre, Le Hou, Tu Vu, Albert Webson, Hyung Won Chung, Yi Tay, Denny Zhou, Quoc V Le, Barret Zoph, Jason Wei, et al. 2023. The flan collection: Designing data and methods for effective instruction tuning. *arXiv preprint arXiv:2301.13688*.

Keming Lu, Hongyi Yuan, Zheng Yuan, Runji Lin, Junyang Lin, Chuanqi Tan, and Chang Zhou. 2023. # instag: Instruction tagging for diversity and complexity analysis. *arXiv preprint arXiv:2308.07074*.

Haipeng Luo, Qingfeng Sun, Can Xu, Pu Zhao, Jianguang Lou, Chongyang Tao, Xiubo Geng, Qingwei Lin, Shifeng Chen, and Dongmei Zhang. 2023a. Wizardmath: Empowering mathematical reasoning for large language models via reinforced evol-instruct. *arXiv preprint arXiv:2308.09583*.

Ziyang Luo, Can Xu, Pu Zhao, Qingfeng Sun, Xiubo Geng, Wenxiang Hu, Chongyang Tao, Jing Ma, Qingwei Lin, and Daxin Jiang. 2023b. Wizardcoder: Empowering code large language models with evol-instruct. *arXiv preprint arXiv:2306.08568*.

OpenAI. 2023. Gpt-4 technical report. *ArXiv*, abs/2303.08774.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744.

Baolin Peng, Chunyuan Li, Pengcheng He, Michel Galley, and Jianfeng Gao. 2023. Instruction tuning with gpt-4. *arXiv preprint arXiv:2304.03277*.

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. 2016. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489.

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

Gemini Team. 2023. Gemini: A family of highly capable multimodal models.

Ye Tian, Baolin Peng, Linfeng Song, Lifeng Jin, Dian Yu, Haitao Mi, and Dong Yu. 2024. Toward self-improvement of llms via imagination, searching, and criticizing. *arXiv preprint arXiv:2404.12253*.

Evan Frick Lisa Dunlap Banghua Zhu Joseph E. Gonzalez Ion Stoica Tianle Li*, Wei-Lin Chiang*. 2024. From live data to high-quality benchmarks: The arena-hard pipeline.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Jiahao Wang, Bolin Zhang, Qianlong Du, Jiajun Zhang, and Dianhui Chu. 2024. A survey on data selection for llm instruction tuning. *arXiv preprint arXiv:2402.05123*.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2022. Self-instruct: Aligning language model with self generated instructions. *arXiv preprint arXiv:2212.10560*.

Yufei Wang, Wanjun Zhong, Liangyou Li, Fei Mi, Xingshan Zeng, Wenyong Huang, Lifeng Shang, Xin Jiang, and Qun Liu. 2023. Aligning large language models with human: A survey. *arXiv preprint arXiv:2307.12966*.

Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. 2021. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837.

Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, and Daxin Jiang. 2023a. Wizardlm: Empowering large language models to follow complex instructions. *arXiv preprint arXiv:2304.12244*.

Yang Xu, Yongqiang Yao, Yufan Huang, Mengnan Qi, Maoquan Wang, Bin Gu, and Neel Sundaresan. 2023b. Rethinking the instruction quality: Lift is what you need.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. *arXiv preprint arXiv:2305.10601*.

Weirui Ye, Shaohuai Liu, Thanard Kurutach, Pieter Abbeel, and Yang Gao. 2021. Mastering atari games with limited data. *Advances in Neural Information Processing Systems*, 34:25476–25488.

Xiao Yu, Maximillian Chen, and Zhou Yu. 2023. Prompt-based monte-carlo tree search for goal-oriented dialogue policy planning. *arXiv preprint arXiv:2305.13660*.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*.

Shengyu Zhang, Linfeng Dong, Xiaoya Li, Sen Zhang, Xiaofei Sun, Shuhe Wang, Jiwei Li, Runyi Hu, Tianwei Zhang, Fei Wu, et al. 2023a. Instruction tuning for large language models: A survey. *arXiv preprint arXiv:2308.10792*.

Shun Zhang, Zhenfang Chen, Yikang Shen, Mingyu Ding, Joshua B Tenenbaum, and Chuang Gan. 2023b. Planning with large language models for code generation. *arXiv preprint arXiv:2303.05510*.

Yingxiu Zhao, Bowen Yu, Binyuan Hui, Haiyang Yu, Fei Huang, Yongbin Li, and Nevin L Zhang. 2023. A preliminary study of the intrinsic relationship between complexity and alignment. *arXiv preprint arXiv:2308.05696*.

Yingxiu Zhao, Bowen Yu, Binyuan Hui, Haiyang Yu, Minghao Li, Fei Huang, Nevin L Zhang, and Yongbin Li. 2024. Tree-instruct: A preliminary study of the intrinsic relationship between complexity and alignment. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 16776–16789.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. 2024. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36.

Andy Zhou, Kai Yan, Michal Shlapentokh-Rothman, Haohan Wang, and Yu-Xiong Wang. 2023a. Language agent tree search unifies reasoning acting and planning in language models. *arXiv preprint arXiv:2310.04406*.

Chunting Zhou, Pengfei Liu, Puxin Xu, Srini Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, et al. 2023b. Lima: Less is more for alignment. *arXiv preprint arXiv:2305.11206*.

Yuchen Zhuang, Xiang Chen, Tong Yu, Saayan Mitra, Victor Bursztyn, Ryan A Rossi, Somdeb Sarkhel, and Chao Zhang. 2023. Toolchain*: Efficient action space navigation in large language models with a* search. *arXiv preprint arXiv:2310.13227*.

A Benchmark Details

Alpaca-Eval (Li et al., 2023b) is a comprehensive evaluation framework incorporating examples from diverse datasets, including self-instruct (Wang et al., 2022), open-assistant (Köpf et al.), Vicuna (Chiang et al., 2023) and Koala (Geng et al., 2023). This framework uses English instructions across multiple categories and tasks to evaluate model performance in real-world scenarios.

MT-Bench (Zheng et al., 2024) is a benchmark designed to assess models' multi-turn conversational and instruction-following abilities. It contains 80 high-quality, multi-turn questions that represent common use cases. The development of MT-Bench is informed by eight categories of user prompts: writing, roleplay, extraction, reasoning, math, coding, stem knowledge, and humanities/social sciences knowledge.

B Evolution Prompts

We designed the evolution prompts to serve as the action space. As shown in Figure 6, it demonstrates a complete evolution prompt. By adding 10-20 words at each step, we ensure the iterative enhancement of the instruction data. Additionally, we presented the case of evolution action, as shown in 8.

C Human Eval

We conducted a blind pairwise comparison between two models: one trained on data generated by MCTS and the other on data generated through random evolution (WizardLM). For this evaluation, we recruited 3 well-educated annotators. Each annotator was presented with two responses: one from the MCTS-based model and one from the random evolution-based model, with their sources randomly shuffled to ensure anonymity. The annotators evaluated each response based on the following criteria (Xu et al., 2023a): (1) Relevance, (2) Knowledgeability, (3) Reasoning, (4) Calculation, and (5) Accuracy. They judged which response was superior for each comparable instance. To estimate the win rate, we compared the frequency of model wins with MCTS. As shown in Figure 5, the model trained on MCTS-generated data achieved significantly better results than the model trained on randomly evolved data. This demonstrates the effectiveness of the MCTS method. Detailed results based on the LLaMA2 are provided in Tables 9, 10, 11, 12, 13, and 14.

D Case Study

We present a case study in Figure 7 to show the iterative evolution of a seed instruction. Starting with the seed instruction, "Name the three Baltic states," we progressively refine it to, "Can you please tell me the names of the three Baltic states and express excitement while sharing them? You can also describe their location on a map.". This process, guided by evaluation models, enhances the efficiency of evolving instructions. High-value instructions are identified and used as the basis for further evolution. Examples of instructions before and after the evolution are provided in Table 15.

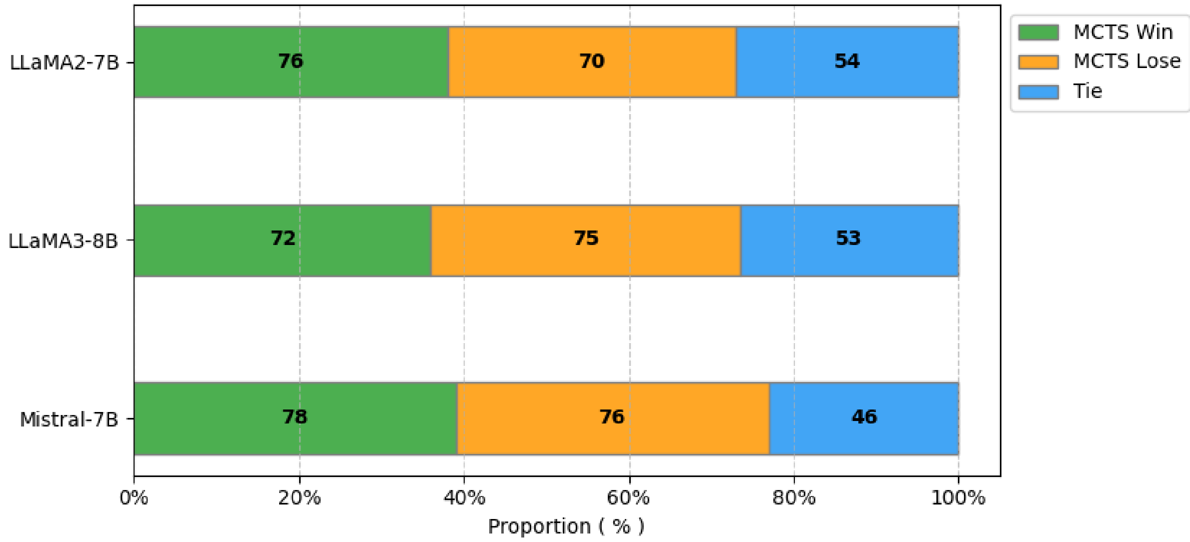

Figure 5: Manual evaluation of the results on Alpaca-eval.

I want you act as a Prompt Rewriter.
Your objective is to rewrite a given prompt into a more complex version which those famous AI systems (e.g., ChatGPT and GPT4) find a bit harder to handle.
But the rewritten prompt must be reasonable and must be understood and responded by humans.
Your rewriting cannot omit the non-text parts such as the table and code in #Given Prompt#:. Also, please do not omit the input in #Given Prompt#.
You should complicate the given prompt using the following method:
Add one or more constraints where necessary into #Given Prompt# to define its limitations and boundaries.
You should try your best not to make the #Rewritten Prompt# become verbose, #Rewritten Prompt# can only add 10 to 20 words into #Given Prompt#.
'#Given Prompt#', '#Rewritten Prompt#', 'given prompt' and 'rewritten prompt' are not allowed to appear in #Rewritten Prompt#
#Given Prompt#: <Here is instruction> #Rewritten Prompt#:

Figure 6: The Evolution Prompt: Add Key Constraints.

Evolution Action	Instruction Evolution Case
Add Global and Local Goals	Original: "Please help me write an article about climate change." Evolved: "Please help me write an article about climate change, aiming to educate readers on the causes and effects of climate change and suggest individual actions to combat it."
Add Key Constraints	Original: "Please help me design a website." Evolved: "Please help me design a website that supports mobile access, and meets accessibility standards."
Add Task Requirements	Original: "Please help me prepare a meeting report." Evolved: "Please help me prepare a meeting report including next quarter's sales strategy recommendations."
Add Problem-Solving Skills	Original: "Please help me solve this math problem. " Evolved: "Please help me solve this math problem and explain each step and the mathematical principles used. "
Add Reasoning Complexity	Original: "Please provide some productivity tips." Evolved: "Please provide some productivity tips, considering different work environments. Explain why these tips are effective."
Add Domain Knowledge	Original: "Please explain blockchain technology." Evolved: "Please provide a detailed explanation of blockchain technology from an IT perspective, including its principles, applications, and future trends."
Add Life Topics	Original: "Please give me some healthy eating advice." Evolved: "Please give me some healthy eating advice, especially for people who exercise regularly."
Add Real-World Applications	Original: "Please explain artificial intelligence." Evolved: "Please explain artificial intelligence and provide examples of its applications in education, customer service, and business."
Add Emotional Expression	Original: "Please help me plan a trip." Evolved: "Please help me plan an exciting trip, including some unique attractions and experiences to ensure the journey is fun."
Format the Input Style	Original: "Please give me some investment advice." Evolved: "As a financial advisor, please give me some stock investment advice, especially beginner strategies."
Format the Output Style	Original: "Please summarize this article." Evolved: "Please summarize this article in a report format, including main points, supporting data, and conclusions."
Refine the Factuality	Original: "Please describe the process of recycling." Evolved: "Please accurately describe the process of recycling plastic bottles, including the collection, sorting, cleaning, shredding, and reprocessing steps."
Create a New One	Original: "Please give me some time management advice." Evolved: "Please give me some advice on how to manage time efficiently throughout the day."

Table 8: Examples of Evolution Action

Instruction: When were smartphones first made?	
MCTS-Evolved Response	**Randomly-Evolved Response**
Smartphones have been around since the early 2000s, but the first model that can be considered a true smartphone was the Apple iPhone, which was released in 2007. The iPhone was a revolutionary device, introducing features such as a touchscreen, web browsing, and the App Store, which allowed users to download and install third-party applications. It was also the first phone to popularize the concept of a "smart" device.	2007 is considered to be the first year that smartphones were commercially available. The first smartphone to be released was the Apple iPhone, which was launched in June 2007.

Table 9: Comparison of LLama2's responses to the instruction about the history of smartphones using different methods of data-driven fine-tuning.

Instruction: Why does it feel like less effort to watch 3-hour long TV episodes back to back than a film?	
MCTS-Evolved Response	**Randomly-Evolved Response**
It is commonly believed that watching TV episodes back-to-back has a psychological effect that makes it feel like less effort than watching a film. Here are some reasons why: 1. Repetitive narrative: TV shows typically have a longer narrative arc, with multiple episodes building up to a larger storyline. Watching multiple episodes in a row allows for the repetition of the same characters, setting, and plot points, which can create a sense of familiarity and ease. 2. Slower pace: TV shows often have a slower pace than films, with more time devoted to character development and dialogue. Watching multiple episodes in a row can make the narrative feel less rushed, as the characters and their interactions unfold over a longer period. 3. Emotional investment: When watching a TV show, viewers often develop emotional investment in the characters and storylines. When viewers watch multiple episodes back-to-back, they are able to maintain this investment, as they are immersed in the world of the show for a longer period. 4. Comfort and familiarity: Watching TV shows back-to-back can create a sense of comfort and familiarity	It is likely due to the way our brains process and perceive time. When watching a 3-hour long TV episode, our brains may perceive it as a single continuous unit, allowing us to experience a sense of flow and continuity. On the other hand, a film typically has distinct acts or segments, each with its own beginning, middle, and end. This makes it easier for our brains to break up the time spent watching it into smaller segments, which may feel more effortful. Additionally, the structure of TV episodes often lends itself to binge-watching, as they are designed to be consumed in large doses. The consistency of the narrative and character arcs, combined with the familiarity of the setting and characters, can also contribute to the perception of less effort when watching multiple episodes back to back.

Table 10: Comparison of LLama2's responses to the instruction about the effort involved in watching TV episodes versus films using different methods of data-driven fine-tuning.

Instruction: In what genres does the given movie or series fall? Manifest	
MCTS-Evolved Response	**Randomly-Evolved Response**
The movie or series you mentioned, "Manifest," falls under the genres of drama, science fiction, and mystery.	The movie or series you mentioned, "Manifest," falls under the genres of drama and science fiction.

Table 11: Comparison of LLama2's responses to the instruction about the genres of "Manifest" using different methods of data-driven fine-tuning.

Instruction: Answer the following literature question as accurately as possible based on your knowledge of literature for high school students. What does the green light symbolize at the end of the book? The Great Gatsby.	
MCTS-Evolved Response	**Randomly-Evolved Response**
In the novel "The Great Gatsby" by F. Scott Fitzgerald, the green light symbolizes Gatsby's unattainable dreams and hopes. It is a beacon of hope and desire, representing Gatsby's longing for Daisy Buchanan and his unrealistic expectations of achieving her love and happiness.	In the novel "The Great Gatsby" by F. Scott Fitzgerald, the green light symbolizes Gatsby's unattainable dream of winning Daisy back.

Table 12: Comparison of LLama2's responses to the instruction about "The Great Gatsby" using different methods of data-driven fine-tuning.

Instruction: The movie tagline is the film's advertising slogan, usually printed on posters prior to the official release. It is a short description or comment on a movie to capture the essence of the movie, and ultimately make you watch the movie. Here is an example: Movie: Joker (2019)	
MCTS-Evolved Response	**Randomly-Evolved Response**
"Put on a Happy Face."	"A gritty origin story that will leave you laughing in the dark."

Table 13: Comparison of LLama2's responses to the instruction about creating a tagline for the movie "Joker" using different methods of data-driven fine-tuning.

Instruction: Can you give any tips on how to cook a juicy, medium-rare steak?	
MCTS-Evolved Response	**Randomly-Evolved Response**
Certainly! Here are some tips for cooking a juicy, medium-rare steak: 1. Use high-quality, grass-fed, or organic beef. The quality of the meat will make a significant difference in the flavor and tenderness. 2. Season your steak with salt and pepper before cooking. This helps to enhance the flavor of the meat and helps to draw out moisture. 3. Heat a cast-iron skillet or grill pan over medium-high heat. Preheat the skillet or grill pan for a few minutes before adding the steak. 4. Cook the steak for about 3 minutes per side for a 1-inch thick steak. Flip the steak only once to ensure even cooking. 5. Use a meat thermometer to check the internal temperature. A medium-rare steak should be cooked to an internal temperature of 135-140°F. 6. Let the steak rest for 5-10 minutes before slicing or serving. This allows the juices to redistribute and the flavors to develop	Certainly! Here are some tips for cooking a juicy, medium-rare steak: 1. Use high-quality, grass-fed beef: Using high-quality beef is essential for achieving a juicy and flavorful steak. Choose beef that is grass-fed and free from added hormones or antibiotics. 2. Trim the fat: Remove as much fat as possible from the steak before cooking. This will help prevent the steak from becoming greasy and ensure a more even cooking. 3. Use a meat thermometer: It is essential to monitor the internal temperature of the steak while cooking. A meat thermometer is the best tool for ensuring that the steak is cooked to your desired level of doneness. Aim for a temperature of 130-135°F (54-57°C) for a medium-rare steak. 4. Season the steak: Before cooking, season the steak with salt and pepper, or your favorite spice mix. This will enhance the flavor of the steak and help to retain moisture

Table 14: Comparison of LLama2's responses to the instruction about cooking a medium-rare steak using different methods of data-driven fine-tuning.

Seed Instruction	Evolved Instruction
Create a list of ingredients to make a traditional lasagna.	Can you compile a tantalizing list of ingredients for both a traditional and a vegetarian lasagna recipe, each with no more than 10 ingredients, ensuring they are both delicious and visually appealing?
Generate a conversation between two friends talking about a dream they shared	Generate a detailed conversation between two friends talking about a vivid and exciting dream they shared last night.
Create a list of ten shared characteristics between birds and reptiles.	Can you generate a list of ten common traits shared by avian and reptilian species, utilizing biological taxonomy and comparative anatomy?
Name five famous French writers.	Imagine you are preparing a presentation for a literary seminar. Can you name five famous French writers from the 19th and 20th centuries, provide brief biographical information for each, and cite at least one notable work? Emphasize their contributions to French culture and literature, and highlight the impact of their works on the global literary community.
What are the best methods for controlling finances?	Can you identify and implement the most effective and sustainable methods for managing personal finances in today's technology-driven society? This should include detailed budgeting techniques and saving strategies that leverage modern financial tools and apps.

Table 15: Seed and Evolved Instructions with MCTS.

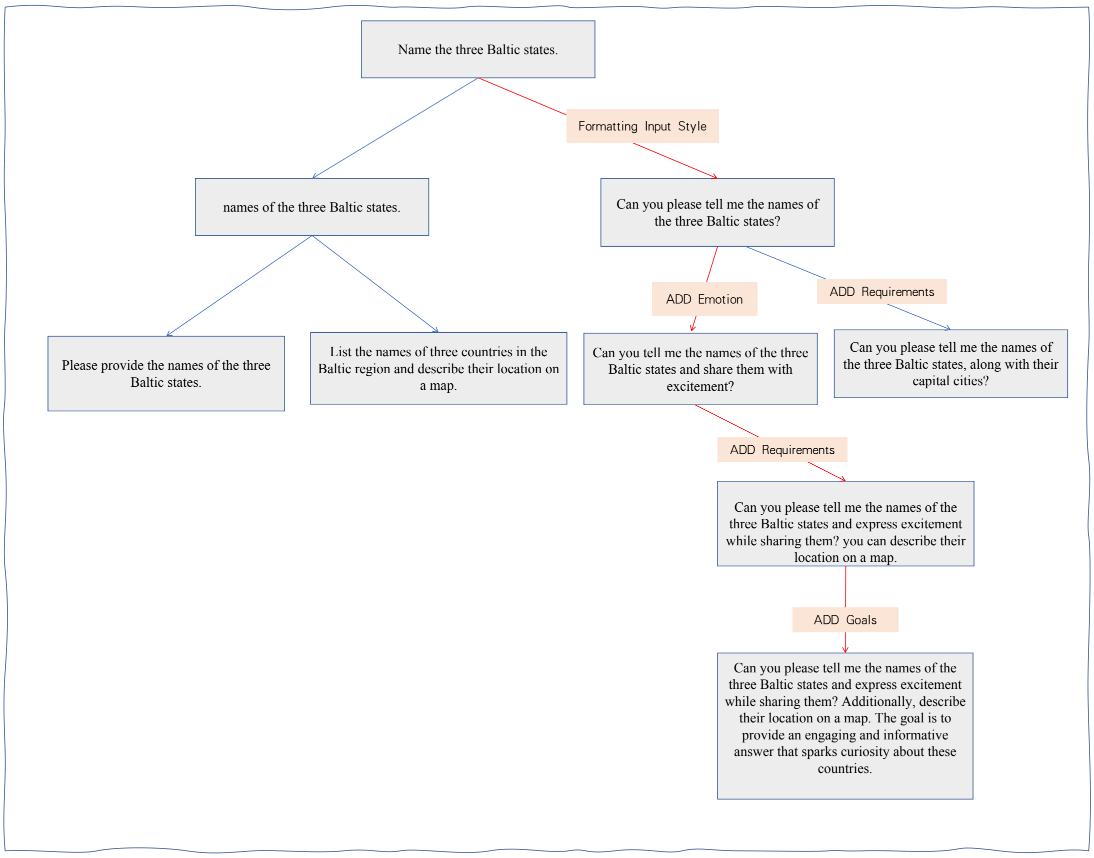

Figure 7: A Case of Instruction Evolution with MCTS.