RIOT: Efficient Prompt Refinement with Residual Optimization Tree

Chenyi Zhou[1], Zhengyan Shi[2], Yuan Yao[1], Lei Liang[3], Huajun Chen[1], Qiang Zhang[1†]

[1]Zhejiang University, [2]University College London, [3]Ant Group

Abstract

Recent advancements in large language models (LLMs) have highlighted their potential across a variety of tasks, but their performance still heavily relies on the design of effective prompts. Existing methods for automatic prompt optimization face two challenges: lack of diversity, limiting the exploration of valuable and innovative directions and semantic drift, where optimizations for one task can degrade performance in others. To address these issues, we propose Residual Optimization Tree (RIOT), a novel framework for automatic prompt optimization. RIOT iteratively refines prompts through text gradients, generating multiple semantically diverse candidates at each step, and selects the best prompt using perplexity. Additionally, RIOT incorporates the text residual connection to mitigate semantic drift by selectively retaining beneficial content across optimization iterations. A tree structure efficiently manages the optimization process, ensuring scalability and flexibility. Extensive experiments across five benchmarks — covering commonsense, mathematical, logical, temporal, and semantic reasoning — demonstrate that RIOT outperforms both previous prompt optimization methods and manual prompting. Our code is released at https://github.com/Qing1Zhong/RiOT.

1 Introduction

Recent advances in large language models (LLMs) have showcased their exceptional performance across a wide range of tasks (Bommasani et al., 2021; Li et al., 2022; Achiam et al., 2023; Team et al., 2024; Chen et al., 2024; Zhang et al., 2024; Jiang et al., 2024b; Xu et al., 2025; Bai et al., 2025). However, the effectiveness of these models often relies heavily on the careful design of prompts (Brown et al., 2020; Reynolds and McDonell, 2021; Kojima et al., 2022; Wei et al., 2022; Amatriain, 2024; Chen, 2024; Jiang et al., 2024a), typically involving labor-intensive trial-and-error processes that require substantial domain expertise and computational resources. Consequently, there is an increasing demand for methods that can automatically optimize prompts, simplifying the task of designing high-performance prompts.

Recently, researchers have begun to explore automatic prompt optimization by directly leveraging LLM outputs and token logits. Zhou et al. (2023) first introduce Automatic Prompt Engineer (APE) concept, showing that LLMs can self-generate effective prompts given a small set of input-output pair demonstrations. Subsequent studies (Pryzant et al., 2023; Yang et al., 2024; Yuksekgonul et al., 2024) expanded on this by integrating concepts such as optimizers and text gradients, using meta-prompts to guide LLMs in refining prompts iteratively based on feedback from training samples.

Despite the promising performance of existing methods (Zhou et al., 2023; Pryzant et al., 2023; Yang et al., 2024; Yuksekgonul et al., 2024), they face two challenges. First, these approaches often struggle with limited diversity during the optimization process. For example, Yang et al. (2024); Yuksekgonul et al. (2024); Pryzant et al. (2023) generate only one candidate prompt per iteration, restricting the breadth of the search space. While Zhou et al. (2023) generates multiple candidates in parallel, it lacks mechanisms for iterative refinement, limiting its ability to optimize the generated prompts effectively. Second, iterative prompt optimization can lead to semantic drift. This occurs when optimizing a prompt for one task inadvertently degrades its performance in other, previously successful scenarios. This issue is similar to the stability-plasticity dilemma observed in continual learning (McCloskey and Cohen, 1989; Kirkpatrick et al., 2017a; Ren et al., 2024), but has yet to be

† Corresponding author: qiang.zhang.cs@zju.edu.cn

Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 22307–22323

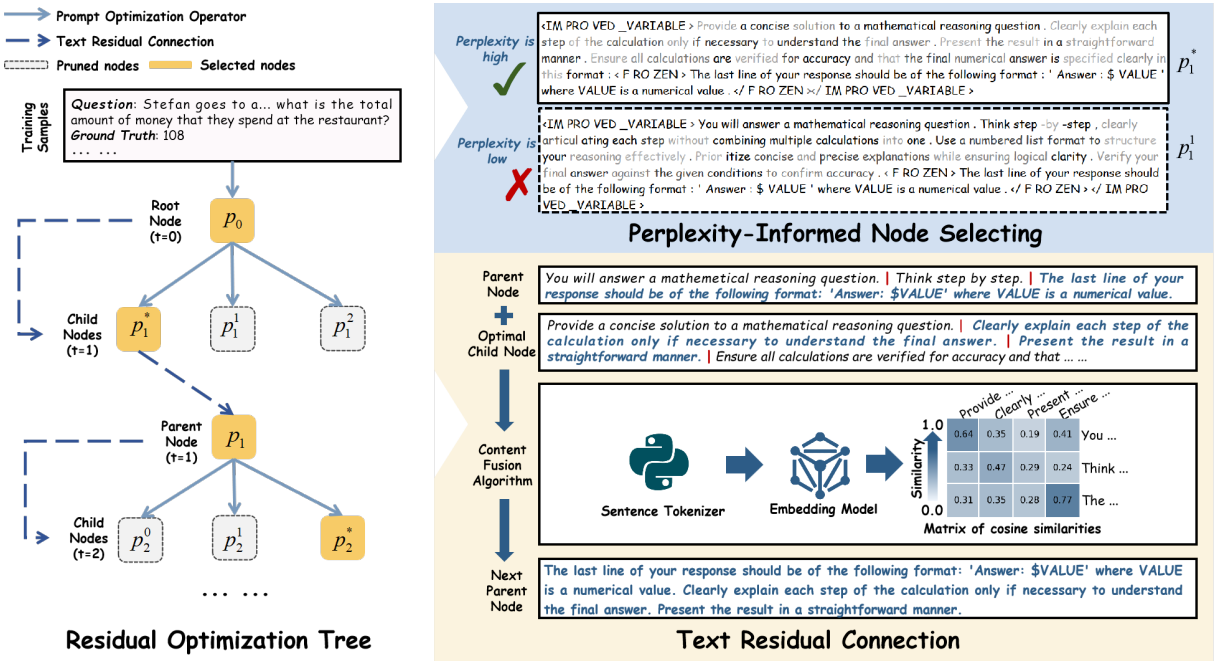

Figure 1: Overview of RIOT. **(i)** At each step, RIOT generates multiple candidate prompts (child nodes). **(ii)** Then, the optimal child node is selected based on perplexity. The perplexity-informed selection process is shown in the region within the blue block. **(iii)** Finally, the parent node is connected to the optimal child node based on semantic similarity. This residual connection process is highlighted within the yellow block.

thoroughly explored in the context of prompt optimization in discrete combinatorial spaces.

To address these challenges, we propose Residual Optimization Tree (RIOT), a novel framework for automatic prompt optimization (see §4). Building on previous works (Zhou et al., 2023; Yang et al., 2024; Yuksekgonul et al., 2024), RIOT iteratively refines prompts using text gradients. However, unlike prior methods, RIOT generates multiple candidate prompts with distinct semantic meanings at each optimization step. Next, RIOT selects the best candidate by calculating the perplexity of each prompt. This approach substantially enhances the diversity during the optimization process while reducing additional computational overhead. To mitigate the risk of semantic drift, RIOT introduces a text residual connection algorithm, which selectively preserves the content of candidates from different optimization iterations based on semantic similarity (see §4.2). Additionally, RIOT employs a tree structure to efficiently manage the hierarchical nature of the optimization process. Each level in the tree corresponds to an optimization iteration, with each node representing a candidate prompt. The selection process acts as a pruning mechanism, while text residual connections link nodes across different levels, ensuring that beneficial information is retained. This tree structure not only enables effective tracking of the optimization process but also guarantees scalability and flexibility.

RIOT demonstrates robust performance in prompt optimization, substantially enhancing the capabilities of LLMs. Across five diverse benchmarks—spanning commonsense reasoning, mathematical reasoning, logical reasoning, temporal understanding, and semantic understanding (see §5)—RIOT outperforms both previous prompt optimization methods and manual prompting (see §6). For example, on the GSM8K benchmark (Cobbe et al., 2021), RIOT improves accuracy by 4.6% (in absolute) over the leading baseline. Our contributions can be summarized as follows:

- We introduce the first tree-based framework with a residual connection for automatic prompt optimization.
- We address the challenge of limited diversity in prompt optimization by effectively exploring prompt space.
- We mitigate the phenomenon of semantic drift by introducing residual learning that retains crucial elements in optimization.

2 Related Work

Prompt Optimization. The majority of work on prompt optimization can be categorized into two main types. The first category focuses on soft prompt tuning, where prompts are represented as task-specific continuous vectors and trained through gradient-based methods (Lester et al., 2021; Li and Liang, 2021; Qin and Eisner, 2021; Liu et al., 2024). The second category employs discrete token search, using gradient-guided approaches (Shin et al., 2020; Gao et al., 2021; Shi et al., 2023; Wen et al., 2024) or reinforcement learning (Deng et al., 2022; Zhang et al., 2023). These works require full access to the language model's parameters or auxiliary reward models, which face substantial limitations when applied to modern LLMs accessible only via API. This underscores the need for black-box strategies that rely solely on textual feedback to optimize the prompt systematically. Some recent works have developed gradient-free methods for prompt optimization. This approach was proposed by Zhou et al. (2023), which involved instructing LLMs to infer the prompt. Pryzant et al. (2023) later formalized the concept of text gradient to refer to textual feedback-based optimization. Subsequent works have further improved upon this by incorporating optimization logic in the prompt space (Yang et al., 2024; Yuksekgonul et al., 2024). However, these methods are limited by a constrained optimization space, which hampers diversity, and fail to address the issue of semantic drift, resulting in suboptimal performance in dynamic contexts.

Semantic Drift. Catastrophic forgetting typically occurs in the context of continual learning. In such scenarios, neural networks may lose previously acquired knowledge when adapting to new tasks, a challenge that has inspired prior work to stabilize parameter updates in continuous spaces through methods such as parameter isolation and regularization (Li and Hoiem, 2017; Kirkpatrick et al., 2017b; Maltoni and Lomonaco, 2019; Dou et al., 2024). The core of these approaches lies in preserving the original weights as much as possible during updates. A similar issue arises in discrete prompt optimization, where iterative modification of prompts can overwrite crucial semantic components, this phenomenon we refer to as semantic drift. In this paper, we address this issue by dynamically preserving high-value semantic components in the prompts through text residual connection.

3 Preliminaries

3.1 Problem Definition

We define the task dataset as $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$, where x_i represents the natural language query, y_i denotes the corresponding ground truth, and N is the total number of test samples. The objective of the prompt optimization task is to search for the optimal prompt p^* that maximizes the LLM performance $\mathcal{F}(\cdot)$ on the given task. This task can be formally defined as:

$$p^* = \arg\max_{p \in \mathcal{P}_{\text{space}}} \sum_{i=1}^{N} \mathcal{S}\left(\mathcal{F}(x_i; p), y_i\right), \tag{1}$$

where $\mathcal{P}_{\text{space}}$ represents the set of all possible prompts (*e.g.,* manually designed prompts or automatically generated prompts) and $\mathcal{S}(\cdot)$ denotes the corresponding evaluation metrics.

3.2 Prompt Optimization Operator

Several frameworks for automatic prompt optimization typically employ similar optimization logic, achieving improved performance. We select TextGrad (Yuksekgonul et al., 2024) as the backbone for executing optimization at each step, as it both follows this successful path and offers great extensibility. For clarity, we define the model responsible for evaluating prompt performance as the target model $\mathcal{F}_{\text{target}}(\cdot)$, and the model used for feedback generation or prompt optimization as the optimization model $\mathcal{F}_{\text{opt}}(\cdot)$. Let $\mathcal{D}_{\text{train}}$ denote the training set. At step t, the target model $\mathcal{F}_{\text{target}}(\cdot)$ generates responses $\{\hat{y}_i^{(t)}\}$ using the current prompt p_t and computes the loss $\mathcal{L}(p_t)$ over $\mathcal{D}_{\text{train}}$:

$$\{\hat{y}_i^{(t)}\} = \mathcal{F}_{\text{target}}(\mathcal{D}_{\text{train}}, p_t) \tag{2}$$

$$\mathcal{L}(p_t) = \mathcal{S}(\mathcal{D}_{\text{train}}, \{\hat{y}_i^{(t)}\}). \tag{3}$$

Then, the optimizer model proposes the candidate prompt p_{t+1}:

$$p_{t+1} = \mathcal{F}_{\text{opt}}(\nabla_{p_t}\mathcal{L}(p_t), p_t). \tag{4}$$

where $\nabla_{p_t}\mathcal{L}(p_t)$ denotes the text gradient encoding improvement directions in natural language. We encapsulate this iterative process into a Prompt Optimization Operator $\mathcal{M}(\cdot)$, defined as:

$$p_{t+1} = \mathcal{M}(\mathcal{D}_{\text{train}}, p_t) \tag{5}$$

This operator abstraction enables modular integration with downstream components.

Algorithm 1 Content Fusion Algorithm

Require: Step t, parent node $\mathcal{P}_{t-1}$, the optimal child node $\mathcal{P}_t^*$, a pretrained embedding model $\mathcal{E}(\cdot)$, a sentence tokenizer $\mathcal{T}(\cdot)$, and two hyperparameters b_1 and b_2.

Divide into sentences:

$\{s_0, s_1, \ldots, s_n\} \leftarrow \mathcal{T}(\mathcal{P}_{t-1})$, $\{r_0, r_1, \ldots, r_m\} \leftarrow \mathcal{T}(\mathcal{P}_t^*)$ ▷ Tokenize $\mathcal{P}_{t-1}$ and $\mathcal{P}_t^*$ using $\mathcal{T}$

Obtain the semantic representation:

$\mathbf{E}_{t-1} \leftarrow \{\mathcal{E}(s_i) \mid i \in [1, n]\}$, $\mathbf{E}_t^* \leftarrow \{\mathcal{E}(r_j) \mid j \in [1, m]\}$ ▷ Embed the sentences using $\mathcal{E}$. $\mathbf{E}_{t-1}$ is an $n \times d$ matrix of embeddings for $\mathcal{P}_{t-1}$. $\mathbf{E}_t^*$ is an $m \times d$ matrix of embeddings for $\mathcal{P}_t^*$. d is the embedding dimension.

compute cosine similarities:

$\mathbf{sim}_{ab} \leftarrow \frac{\mathbf{E}_{t-1} \cdot \mathbf{E}_t^{*-T}}{\|\mathbf{E}_{t-1}\|_2 \|\mathbf{E}_t^*\|_2}$ ▷ Matrix of cosine similarities of size $n \times m$

Select sentences based on similarity thresholds:

$selected\_t \leftarrow \{s_{idx} \mid \max(\text{sim}_{ab}[idx, :]) \geq 1 - b_1\}$ ▷ Select sentences from $\mathcal{P}_{t-1}$

$selected\_r \leftarrow \{r_{idx} \mid \max(\text{sim}_{ab}[:, idx]) < 1 - b_2\}$ ▷ Select sentences from $\mathcal{P}_t^*$

$\mathcal{P}_t \leftarrow selected\_t \cup selected\_r$

Return $\mathcal{P}_t$

4 Residual Optimization Tree

In this section, we introduce our proposed method Residual Optimization Tree (RIOT), as shown in Figure 1. Specifically, RIOT starts with an initial prompt p_0 as the root node. At step t, each parent node p_t produces K candidate child nodes $\{p_{t+1}^{(i)}\}_{i=1}^K$ through prompt optimization operator $\mathcal{M}(\cdot)$ (*i.e.,* the width of tree is K). Since LLMs exhibit inherent variability in their outputs, the same input may yield different outputs. To account for this, RIOT computes the loss $\mathcal{L}(p_t)$ K times, generating K different candidate prompts. Then we propose a dynamic pruning strategy to select the optimal child node p_{t+1}^* from candidates $\{p_{t+1}^{(i)}\}_{i=1}^K$ (explained in §4.1). This approach fosters greater diversity within the prompt space without sacrificing quality.

To address the challenge of semantic drift, we introduce the Text Residual Connection (detailed in §4.2). This mechanism is inspired by the concept of residual learning in deep networks and ensures that semantic components from the parent node can effectively be transferred to child nodes during iterative optimization, reducing excessive divergence between parent and child nodes.

4.1 Perplexity-Informed Node Selection

A common strategy of child node selection often relies solely on the quality of the candidate nodes generated by the LLM, overlooking the importance of diversity in the optimization process. This can result in less innovative outcomes, limiting the exploration of valuable prompt combinations and missing innovative directions. Instead, we introduce a perplexity-informed approach to select the most promising child node. The idea is to prioritize nodes with higher semantic diversity.

For each child node candidate $p_{t+1}^{(i)}$, we compute its perplexity $PPL(p_{t+1}^{(i)})$, which measures the uncertainty of the model's prediction for the candidate content. Given the candidate prompt $p_{t+1}^{(i)} = (x_0, x_1, \cdots, x_J)$, the perplexity $PPL(p_{t+1}^{(i)})$ is defined as:

$$PPL(p_{t+1}^{(i)}) = \exp\{-\frac{1}{J}\sum_{j=1}^{J} p_\theta(x_j \mid x_{<j})\}, \quad (6)$$

where $p_\theta(x_j \mid x_{<j})$ is the likelihood of the j-th token conditioned on the preceding tokens $x_{<j}$ according to the LLM. The perplexity value is then incorporated into the selection process. First, from an information theoretic perspective (Shannon, 1948), higher perplexity indicates lower token co-occurrence probability, suggesting more information. Second, established practices in Bayesian optimization (Snoek et al., 2012) preferentially explore regions of high uncertainty to maximize information gain. Therefore, we select the optimal child node p_{t+1}^* by maximizing the perplexity value:

$$p_{t+1}^* = \arg\max_i PPL(p_{t+1}^{(i)}). \quad (7)$$

This perplexity-informed node selection approach enhances RIOT's ability to traverse a diverse and effective prompt space.

4.2 Text Residual Connection

To mitigate semantic drift during the iterative prompt optimization process, we propose the Text Residual Connection, which is inspired by residual learning techniques in deep neural networks. The core idea is to ensure that the important information from the parent prompt is preserved and effectively transferred to the successor prompt, preventing the forgetting phenomenon that can occur when iteratively modifying prompts. In traditional neural networks (He et al., 2016), residual connections facilitate the learning of differences between the input and target representations, enabling deeper networks without suffering from vanishing gradients. We adopt a similar concept to prompt optimization. Initializing with prompt p_0 as the tree root, at step t RIOT generate an optimized child node p_t^* from the parent node p_{t-1}. Contrary to previous strategies that directly propagate p_t^* as the immediate successor p_t, RIOT computes the semantic residual components which represent the difference between the child node p_t^* and the parent node p_{t-1}, then fuses them through controlled composition, as follows:

$$p_t = \mathcal{G}(p_{t-1}, p_t^*). \tag{8}$$

Here, $\mathcal{G}(\cdot)$ represents the **Content Fusion Algorithm** outlined in Algorithm 1, which is achieved by adjusting two hyperparameters, b_1 and b_2, that control the degree of fusion between the parent and child prompts. This ensures that the successor prompt p_t retains the meaningful elements from the parent that contribute to the optimization trajectory, while introducing new elements from the optimized child node.

In essence, the Text Residual Connection can be viewed as a way of progressively refining the prompt, ensuring that each iteration of optimization builds upon the information already captured by the previous node, rather than starting from scratch, thereby semantic consistency across all prompts in the optimization tree.

5 Experimental Step

Datasets. We evaluate RIOT across five benchmarks, including **LogiQA 2.0** (Liu et al., 2023), **StrategyQA** (Geva et al., 2021), **Object Counting** (Srivastava et al., 2023; Suzgun et al., 2023), **GSM8K** (Cobbe et al., 2021), and **Date Understanding** (Srivastava et al., 2023; Suzgun et al., 2023), which are designed to cover diverse task types such as temporal understanding, semantic reasoning, mathematical computation, and commonsense inference. For detailed dataset distribution, please refer to the Appendix E.

Baselines. We compare RIOT to prior works in two categories:

- **Manual Prompting Methods.** We experiment with (1) zero-shot Chain-of-Thought (CoT) prompting and (2) few-shot CoT prompting (Kojima et al., 2022; Wei et al., 2022). See Appendix F for the prompt template construction details.

- **Automatic Prompt Optimization Methods.** We compare with four established approaches. (1) APE (Zhou et al., 2023) leverages LLMs to generate candidate prompts based on task-specific input-output pairs, followed by a selection process to identify the optimal prompts. (2) OPRO (Yang et al., 2024) positions language models as optimizers, utilizing the meta-prompt to systematically refine the initial prompt. (3) TextGrad (Yuksekgonul et al., 2024) adopts a parameterization perspective by treating prompts as optimizable parameters while introducing the concept of textual gradients for optimization. (4) DSPy (Khattab et al., 2024) formulates prompt optimization as a declarative program synthesis task.

Implementation Details. Following prior work (Yuksekgonul et al., 2024), we configure both GPT-3.5-turbo (OpenAI, 2022) (as the target model) and GPT-4o (OpenAI, 2024) (as the optimization model) with temperature parameters fixed at 0. Moreover, we tokenize the prompt using the NLTK sentence tokenizer (Bird, 2006). The `text-embedding-3-large` model (OpenAI, 2023) is used for embedding, with b_1 set to 0.25 and b_2 set to 0.5. K is set to 3. A batch size of 4 is utilized with 3 training epochs, yielding 15 iterations through the optimization process. We use the zero-shot CoT prompts as initial prompts for prompt optimization methods.

We report accuracy with corresponding standard deviation, based on five independent runs using fixed prompts and test samples. This approach accounts primarily for the variance introduced by the decoding strategy. The optimal prompt, identified on the validation set, is used for testing.

Method	True / False		Generative		Multiple-choice
	LogiQA 2.0 (N=160)	StrategyQA (N=100)	Object Counting (N=210)	GSM8K (N=100)	Date Understanding (N=329)
Manual Prompting Methods					
Zero-Shot CoT	59.0 ±1.5	65.8 ±1.9	71.0 ±1.9	60.2 ±1.5	76.1 ±1.0
Four-Shot CoT	59.7 ±3.5 ↑0.7	71.2 ±1.0 ↑5.4	71.2 ±2.2 ↑0.2	73.4 ±1.5 ↑13.2	73.2 ±1.7 ↓2.9
Twenty-Shot CoT	59.4 ±1.0 ↑0.4	71.6 ±0.8 ↑5.8	70.8 ±1.4 ↓0.2	72.8 ±1.2 ↑12.6	76.3 ±1.9 ↑0.2
Automatic Prompt Optimization Methods					
APE (Zhou et al., 2023)	57.4 ±3.5 ↓1.6	70.4 ±3.3 ↑4.6	79.2 ±1.9 ↑8.2	76.6 ±1.9 ↑16.4	72.8 ±2.8 ↓3.3
OPRO (Yang et al., 2024)	58.0 ±4.8 ↓1.0	67.0 ±3.0 ↑1.2	79.9 ±1.3 ↑8.9	72.8 ±2.0 ↑12.6	76.3 ±0.5 ↑0.2
TextGrad (Yuksekgonul et al., 2024)	60.0 ±1.2 ↑1.0	68.8 ±1.7 ↑3.0	**88.3** ±1.2 ↑17.3	74.8 ±1.0 ↑14.6	71.9 ±0.7 ↓4.2
DSPy (Khattab et al., 2024)	59.8 ±1.6 ↑0.8	73.4 ±1.5 ↑7.6	84.5 ±2.2 ↑13.5	79.0 ±1.7 ↑18.8	74.3 ±1.1 ↓1.8
RIOT (**Ours**)	**61.4** ±1.5 ↑2.4	**74.6** ±1.5 ↑8.8	86.9 ±0.6 ↑15.9	**81.2** ±1.2 ↑21.0	**78.2** ±0.6 ↑2.1

Table 1: Comparison of Automatic Prompt Optimization Methods. We report mean accuracy (%) and standard deviation. **Bold** and underlined values indicate best and second-best performance, respectively. Red arrows indicate the absolute performance decrease, while green arrows indicate the absolute performance increase, both compared to the Zero-Shot CoT prompting. N represents the number of samples used for testing.

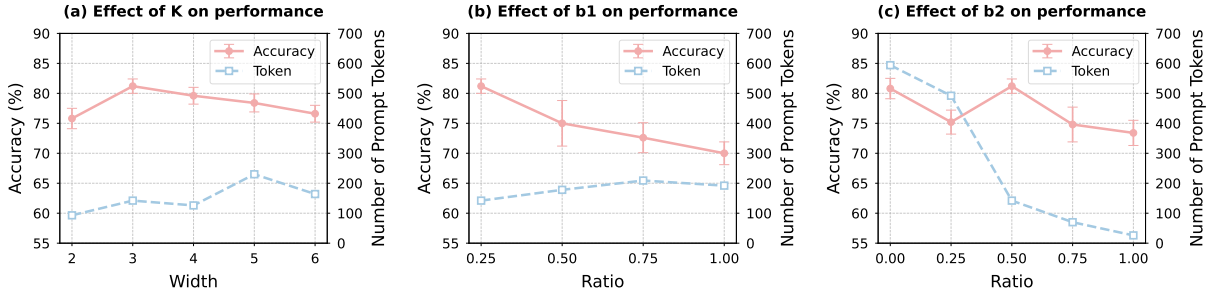

Figure 2: Hyperparametric Senstivity Analysis of RIOT. Our results show that **(a)**: Increasing the tree width K leads to a unimodal curve, with peak accuracy at $K = 3$; **(b)**: The parameter b_1 is negatively correlated with performance; and **(c)**: The parameter b_2 plays a critical role in balancing performance and computational overhead.

6 Results

6.1 Main Results

Finding #1: Scaling the number of demonstration examples in few-shot CoT prompting leads to diminishing returns. As shown in Table 1, while few-shot CoT outperforms zero-shot across tasks, scaling the number of demonstrations yields minimal gains. For instance, on StrategyQA, expanding the prompt from 4 to 20 examples improves accuracy by merely 0.4%, while on LogiQA 2.0, Twenty-shot CoT prompting unexpectedly underperforms Four-shot CoT prompting by 0.3%. We posit that this phenomenon stems from two critical factors: (1) the escalating prompt length challenges models' contextual comprehension due to attention dilution, and (2) redundant examples often fail to provide task-specific inductive signals, forcing models to implicitly extrapolate patterns from noisy demonstrations.

Finding #2: RIOT outperforms manual prompting and automatic prompt optimization baselines. In Table 1, RIOT consistently improves accuracy across all five reasoning tasks compared to manual prompting methods. For instance, RIOT achieves accuracy gains of 3.0% (71.6% → 74.6%) on StrategyQA and 7.8% (73.4% → 81.2%) on GSM8K compared to the best few-shot CoT baseline. When compared to zero-shot CoT prompting, RIOT achieves even larger improvements, with accuracy rising by 8.8% (65.8% → 74.6%) and 21.0% (60.2% → 81.2%) on these two tasks. Additionally, RIOT surpasses other automatic prompt optimization methods across four benchmarks. Specifically, it achieves a 1.4% increase (60.0% → 61.4%) on LogiQA 2.0, 1.2% (73.4% → 74.6%) on StrategyQA, 2.2% (79.0% → 81.2%) on GSM8K, and 1.9% (76.3% → 78.2%) on Date Understanding, while attaining suboptimal performance on Object Counting with an accuracy of 86.9%. Beyond per-task gains, RIOT also achieves the highest weighted average accuracy across all datasets, as reported in Appendix A, exceeding the best-performing baseline by 2.7%. These results highlight RIOT's capability to generate higher-quality

Experiment	TRUE / FALSE		GENERATIVE		MULTIPLE-CHOICE
	LogiQA 2.0 (N=160)	StrategyQA (N=100)	Object Counting (N=210)	GSM8K (N=100)	Date Understanding (N=329)
Baseline	65.9 ±0.8	79.6 ±1.9	<u>92.6</u> ±0.9	93.4 ±0.8	<u>89.9</u> ±0.4
Prompt Transferability	<u>67.0</u> ±0.5 ↑1.1	<u>81.4</u> ±1.0 ↑1.8	90.2 ±0.9 ↓2.4	<u>93.6</u> ±0.8 ↑0.2	86.7 ±0.3 ↓3.2
Model Transferability	**69.4** ±0.4 ↑3.5	**82.0** ±0.6 ↑2.4	**93.0** ±0.6 ↑0.4	**95.0** ±0.6 ↑1.6	**90.5** ±0.6 ↑0.6

Table 2: Generalization performance of RIOT across five datasets. All experiments are evaluated on Gemini-1.5-flash, and optimizer model is GPT-4o. **Baseline**: Applying the zero-shot CoT prompt. **Prompt Transferability**: Prompts are optimized for GPT-3.5-turbo. **Model Transferability**: Prompts are optimized for Gemini-1.5-flash. **Bold** and <u>underlined</u> values indicate best and second-best performance, respectively. Red arrows indicate the absolute performance decrease, while gree arrows indicate the absolute performance increase, both compared to the baseline. N represents the number of samples used for testing.

Method	Accuracy (%)
RIOT	**81.2** ±1.2
w/o TEXT RESIDUAL CONNECTION	68.8 ±3.9 ↓12.4
w/o PERPLEXITY-INFORMED NODE SELECTION	68.2 ±1.8 ↓13.0

Table 3: Ablation study results. **Bold** value indicates the best performance. Red arrows indicate the absolute performance decrease compared to the full model.

Metric	Perplexity	Entropy	Length
Accuracy (%)	**81.2** ±1.2	78.6 ±1.0 ↓2.6	73.6 ±4.3 ↓7.6

Table 4: Comparison of different node selection metrics on GSM8K. **Bold** value indicates the best performance. Red arrows indicate the absolute performance decrease compared to perplexity.

Embedding Model	Accuracy (%)
`text-embedding-3-large`	**81.2** ±1.2
`text-embedding-3-small`	77.8 ±2.7 ↓3.4
`text-embedding-ada-002`	60.2 ±1.5 ↓21.0

Table 5: Impact of embedding model choice on performance. **Bold** value indicates the best performance. Red arrows indicate the absolute performance decrease compared to the full model. Our results show that more recent and larger models achieve better performance.

prompts, enabling more effective learning with limited data and outperforming all baseline methods.

Finding #3: RIOT demonstrates robust cross-task prompt optimization. In Table 1, RIOT improves performance across all five tasks. In contrast, other automatic prompt optimization methods show inconsistent generalization performance. For instance, APE achieves only 57.4% and 72.8% accuracy on the LogiQA 2.0 and Date Understanding datasets, respectively, which are lower than the 59.0% and 76.1% achieved by zero-shot CoT prompting, indicating a degradation in performance. Meanwhile, RIOT demonstrates superior cross-task stability, successfully optimizing the initial prompt in all five tasks.

6.2 Further Analysis

Hyperparametric sensitivity analysis of RIOT. On GSM8K, we systematically analyze the effect of tree width K, hyperparameters b_1, and b_2 on RIOT, as visualized in Figure 2. The results reveal the following key findings: (a) Increasing the tree width W from 2 to 6 leads to a unimodal accuracy curve, initially rising and then declining, with peak accuracy of 81.2% at $K = 3$; (b) The parameter b_1 is negatively correlated with performance. As b_1 increases from 0.25 to 1, accuracy decreases by 11.2%, underscoring the importance of dynamically removing redundant parent node information; (c) The parameter b_2 plays a critical role in balancing performance and computational overhead. As b_2 increases from 0 to 0.5, accuracy fluctuates non-monotonically, while the optimized prompt length rapidly decreases. Further increasing b_2 to 1 results in a decline in accuracy from 81.2% to 73.4%, with the optimized prompt length reduced to 26 tokens. The optimal trade-off between performance and efficiency is achieved at $b_2 = 0.5$, demonstrating that moderate length constraints can effectively eliminate redundant lexical units while preserving essential reasoning information.

Generalization analysis of RIOT. We conduct systematic evaluations to validate the generalization of RIOT across two critical dimensions: **(1) Prompt Transferability**: Assessing whether prompts optimized for one model generalize to another. **(2) Model Transferability**: Evaluating RIOT's ability to optimize prompts for distinct

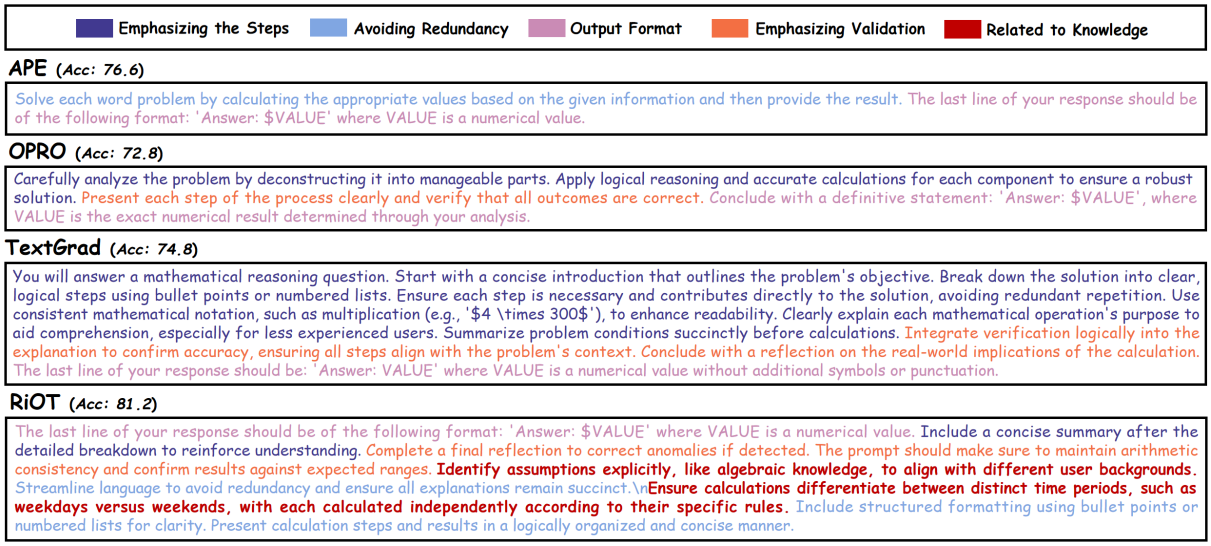

Figure 3: List of optimized prompts on GSM8K by different prompt optimization methods, categorized according to five key dimensions: emphasizing the steps, avoiding redundancy, output format, emphasizing validation, and relating to domain-specific knowledge. RIOT integrates all dimensions, enhancing reliability and aligning solutions with real-world contexts.

target models. In both experiments, prompts optimized by GPT-4o are evaluated on Gemini-1.5-flash (Team et al., 2024), with the first experiment optimizing for GPT-3.5-turbo and the second for Gemini-1.5-flash. Additionally, we evaluate the zero-shot CoT prompt on Gemini-1.5-flash as the baseline. The results are shown in Table 2. Optimizing prompts for GPT-3.5-turbo boosts the performance of Gemini-1.5-flash across three tasks. Furthermore, when Gemini-1.5-flash is used as the target model for prompt optimization with RIOT, its performance surpasses the baseline across five tasks. For instance, on the StrategyQA dataset, accuracy improves by 1.8% in the Prompt Transferability experiment and by 2.4% in the Model Transferability experiment, both compared to the baseline. These results demonstrate the effectiveness of RIOT in optimizing prompts for both model and prompt transferability.

Ablation study on each component of RIOT on model performance. In Table 3, we evaluate individual component of RIOT on GSM8K. In particular, we compare two variants: **(1) w/o Text Residual Connection**: The optimal child node in each iteration is used as the parent node for the next, without incorporating any information from the current prompt, **(2) w/o Perplexity-Informed Node Selection**: Each optimization step generates only a single candidate prompt (*i.e.,* $K = 1$), disabling the selection mechanism. When the number of candidate prompts is limited to one, the effect of Text Residual Connection is minimal, resulting in a 13% decrease in accuracy, highlighting the importance of diversity in automatic prompt optimization. Additionally, when there is a lack of inheritance between parent and child nodes during iterations, accuracy drops by 12.4%, reinforcing our hypothesis that semantic drift occurs during the optimization process. These results show that RIOT benefits from both prompt diversity and sentence-level semantic fusion, and that both components are crucial for stable and effective optimization.

Impact of node selection metrics. To evaluate the impact of different node selection strategies in RIOT, we compare three metrics: perplexity, entropy, and length, which capture different aspects of prompt quality—semantic likelihood, prediction uncertainty, and verbosity. As shown in Table 4, experimental results on GSM8K demonstrate that perplexity-informed selection consistently outperforms the other two, indicating its effectiveness in capturing semantic diversity and informativeness during prompt optimization. Formal definitions of entropy and length are provided in Appendix B.

Better embedding models lead to improved performance. Table 5 shows RIOT's sensitivity to embedding quality on GSM8K, comparing distinct embedding models: text-embedding-3-large (top-tier) (OpenAI, 2023), text-embedding-3-small (mid-tier) (OpenAI, 2023), and text-embedding-ada-002 (base) (Ryan et al., 2022). Accuracy drops

by 3.4% and 21% respectively when downgrading models, revealing RIOT's strong dependency on semantic discrimination capability. This dependency stems from the fact that lower-performing embedding models inadequately discriminate subtle lexical variations, impairing the identification of semantically similar units between parent and child nodes during residual connection. Consequently, suboptimal embedding models require careful calibration of the similarity threshold parameters b_1 and b_2 to mitigate performance loss.

6.3 Case Study

As shown in Figure 3, we compare prompts optimized by different automatic prompt optimization methods on GSM8K. This comparison allows us to further explore RIOT. Unlike baselines, RIOT's optimized version uniquely combines verification mechanisms and domain adaptation, enforcing arithmetic consistency while eliminating redundant reasoning steps. Crucially, the explicit incorporation of temporal rule decoupling and algebraic assumption specification highlights RIOT's ability to dynamically assimilate domain-specific patterns from training samples during optimization. This practical insight enables RIOT to produce solutions with higher real-world alignment.

7 Conclusion

In this paper, we propose Residual Optimization Tree (RIOT), a novel tree-based optimization framework that automatically generates higher-quality prompts. We employ a perplexity-informed node selection strategy to increase diversity in the prompt optimization process and incorporate a text residual connection to preserve the crucial semantic information. Experimental results show that RIOT demonstrates substantial improvements in performance across five reasoning tasks while maintaining generalization.

Limitations

Despite the promising results, our study has two main limitations, which reflect broader challenges in gradient-free automatic prompt optimization. First, our research focuses on textual tasks. However, the growing use of multimodal large language models (MLLMs) in real-world applications—ranging from visual question answering to robotic instruction synthesis—highlights the critical need for multimodal prompt engineering. Whether our method retains its effectiveness in the multimodal domain is an important direction for future exploration. Second, we observe notable performance variation across different task categories, suggesting challenges in cross-task generalization for automatic prompt optimization. This issue is likely due to inherent differences in how LLMs organize and activate task-specific knowledge, pointing to the need for further investigation into the latent knowledge structures within LLMs.

Acknowledgements

This work is funded by National Natural Science Foundation of China (U23A20496, 62302433), Zhejiang Provincial Key Research and Development Project of China (2024C01135), Zhejiang Provincial Natural Science Foundation of China (LQ24F020007) and the Ningbo Natural Science Foundation (2024J020). This work was supported by Ant Group.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Xavier Amatriain. 2024. Prompt design and engineering: Introduction and advanced methods. *arXiv preprint arXiv:2401.14423*.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, et al. 2025. Qwen2. 5-vl technical report. *arXiv preprint arXiv:2502.13923*.

Steven Bird. 2006. Nltk: the natural language toolkit. In *Proceedings of the COLING/ACL 2006 Interactive Presentation Sessions*, pages 69–72.

Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. 2021. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*.

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.

Huajun Chen. 2024. Large knowledge model: Perspectives and challenges. *Data Intelligence*, 6(3):587–620.

Justin Chih-Yao Chen, Swarnadeep Saha, and Mohit Bansal. 2023. Reconcile: Round-table conference improves reasoning via consensus among diverse llms. *arXiv preprint arXiv:2309.13007*.

Songlin Chen, Weicheng Wang, Xiaoliang Chen, Peng Lu, Zaiyan Yang, and Yajun Du. 2024. Llama-lora neural prompt engineering: A deep tuning framework for automatically generating chinese text logical reasoning thinking chains. *Data Intelligence*, 6(2):375–408.

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*.

Mingkai Deng, Jianyu Wang, Cheng-Ping Hsieh, Yihan Wang, Han Guo, Tianmin Shu, Meng Song, Eric Xing, and Zhiting Hu. 2022. RLPrompt: Optimizing discrete text prompts with reinforcement learning. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 3369–3391, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Shihan Dou, Enyu Zhou, Yan Liu, Songyang Gao, Wei Shen, Limao Xiong, Yuhao Zhou, Xiao Wang, Zhiheng Xi, Xiaoran Fan, Shiliang Pu, Jiang Zhu, Rui Zheng, Tao Gui, Qi Zhang, and Xuanjing Huang. 2024. LoRAMoE: Alleviating world knowledge forgetting in large language models via MoE-style plugin. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1932–1945, Bangkok, Thailand. Association for Computational Linguistics.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B Tenenbaum, and Igor Mordatch. 2023. Improving factuality and reasoning in language models through multiagent debate. *arXiv preprint arXiv:2305.14325*.

Tianyu Gao, Adam Fisch, and Danqi Chen. 2021. Making pre-trained language models better few-shot learners. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3816–3830, Online. Association for Computational Linguistics.

Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. 2021. Did aristotle use a laptop? A question answering benchmark with implicit reasoning strategies. *Trans. Assoc. Comput. Linguistics*, 9:346–361.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Songtao Jiang, Yan Zhang, Chenyi Zhou, Yeying Jin, Yang Feng, Jian Wu, and Zuozhu Liu. 2024a. Joint visual and text prompting for improved object-centric perception with multimodal large language models. *arXiv preprint arXiv:2404.04514*.

Songtao Jiang, Tuo Zheng, Yan Zhang, Yeying Jin, Li Yuan, and Zuozhu Liu. 2024b. Med-MoE: Mixture of domain-specific experts for lightweight medical vision-language models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 3843–3860, Miami, Florida, USA. Association for Computational Linguistics.

Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan A, Saiful Haq, Ashutosh Sharma, Thomas T. Joshi, Hanna Moazam, Heather Miller, Matei Zaharia, and Christopher Potts. 2024. DSPy: Compiling declarative language model calls into state-of-the-art pipelines. In *The Twelfth International Conference on Learning Representations*.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. 2017a. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. 2017b. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526.

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.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597, Online. Association for Computational Linguistics.

Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, et al. 2022. Competition-level code generation with alphacode. *Science*, 378(6624):1092–1097.

Zhizhong Li and Derek Hoiem. 2017. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947.

Hanmeng Liu, Jian Liu, Leyang Cui, Zhiyang Teng, Nan Duan, Ming Zhou, and Yue Zhang. 2023. Logiqa 2.0—an improved dataset for logical reasoning in natural language understanding. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*.

Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. 2024. Gpt understands, too. *AI Open*, 5:208–215.

Davide Maltoni and Vincenzo Lomonaco. 2019. Continuous learning in single-incremental-task scenarios. *Neural Networks*, 116:56–73.

Michael McCloskey and Neal J Cohen. 1989. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier.

OpenAI. 2022. Introducing chatgpt. https://openai.com/index/chatgpt/. Accessed: 2025-01-21.

OpenAI. 2023. New embedding models and api updates. https://openai.com/index/new-embedding-models-and-api-updates/. Accessed: 2025-01-21.

OpenAI. 2024. Hello gpt-4o. https://openai.com/index/hello-gpt-4o/. Accessed: 2025-01-21.

Reid Pryzant, Dan Iter, Jerry Li, Yin Lee, Chenguang Zhu, and Michael Zeng. 2023. Automatic prompt optimization with "gradient descent" and beam search. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7957–7968, Singapore. Association for Computational Linguistics.

Guanghui Qin and Jason Eisner. 2021. Learning how to ask: Querying LMs with mixtures of soft prompts. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5203–5212, Online. Association for Computational Linguistics.

Weijieying Ren, Xinlong Li, Lei Wang, Tianxiang Zhao, and Wei Qin. 2024. Analyzing and reducing catastrophic forgetting in parameter efficient tuning. *arXiv preprint arXiv:2402.18865*.

Laria Reynolds and Kyle McDonell. 2021. Prompt programming for large language models: Beyond the few-shot paradigm. In *Extended abstracts of the 2021 CHI conference on human factors in computing systems*, pages 1–7.

Greene Ryan, Sanders Ted, Weng Lilian, and Neelakantan Arvind. 2022. New and improved embedding model. https://openai.com/index/new-and-improved-embedding-model/. Accessed: 2025-01-21.

Claude Elwood Shannon. 1948. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423.

Weijia Shi, Xiaochuang Han, Hila Gonen, Ari Holtzman, Yulia Tsvetkov, and Luke Zettlemoyer. 2023. Toward human readable prompt tuning: Kubrick's the shining is a good movie, and a good prompt too? In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 10994–11005, Singapore. Association for Computational Linguistics.

Taylor Shin, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, and Sameer Singh. 2020. AutoPrompt: Eliciting Knowledge from Language Models with Automatically Generated Prompts. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4222–4235, Online. Association for Computational Linguistics.

Jasper Snoek, Hugo Larochelle, and Ryan P Adams. 2012. Practical bayesian optimization of machine learning algorithms. *Advances in neural information processing systems*, 25.

Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, et al. 2023. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Transactions on Machine Learning Research*.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V. Le, Ed H. Chi, Denny Zhou, and Jason Wei. 2023. Challenging big-bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 13003–13051. Association for Computational Linguistics.

Gemini Team, Petko Georgiev, Ving Ian Lei, Ryan Burnell, Libin Bai, Anmol Gulati, Garrett Tanzer, Damien Vincent, Zhufeng Pan, Shibo Wang, et al. 2024. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*.

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.

Yuxin Wen, Neel Jain, John Kirchenbauer, Micah Goldblum, Jonas Geiping, and Tom Goldstein. 2024. Hard prompts made easy: Gradient-based discrete optimization for prompt tuning and discovery. *Advances in Neural Information Processing Systems*, 36.

Qinwei Xu, Xingkun Xu, Chenyi Zhou, Zuozhu Liu, Feiyue Huang, Shaoxin Li, Lifeng Zhu, Zhian Bai, Yuchen Xu, and Weiguo Hu. 2025. Towards normalized clinical information extraction in chinese

radiology report with large language models. *Expert Systems with Applications*, 271:126585.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V. Le, Denny Zhou, and Xinyun Chen. 2024. Large language models as optimizers. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2024. Tree of thoughts: Deliberate problem solving with large language models. *Advances in Neural Information Processing Systems*, 36.

Mert Yuksekgonul, Federico Bianchi, Joseph Boen, Sheng Liu, Zhi Huang, Carlos Guestrin, and James Zou. 2024. Textgrad: Automatic" differentiation" via text. *arXiv preprint arXiv:2406.07496*.

Tianjun Zhang, Xuezhi Wang, Denny Zhou, Dale Schuurmans, and Joseph E. Gonzalez. 2023. TEMPERA: Test-time prompt editing via reinforcement learning. In *The Eleventh International Conference on Learning Representations*.

Xilin Zhang, Zhixin Mao, Ziwen Chen, and Shen Gao. 2024. Effective tool augmented multi-agent framework for data analysis. *Data Intelligence*, 6(4):923–945.

Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. 2023. Large language models are human-level prompt engineers. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

A Weighted Average Accuracy Across Datasets

To provide a holistic evaluation of each method's performance across all benchmark datasets, we report their Weighted Average Accuracy (WAA). The WAA is defined as:

$$\text{WAA} = \frac{\sum_{i=1}^{n} N_i \cdot A_i}{\sum_{i=1}^{n} N_i} \tag{9}$$

where N_i denotes the number of test samples and A_i denotes the average accuracy on the i-th dataset. n is the number of datasets.

As shown in Figure 4, RIOT achieves the highest WAA of 77.2%, outperforming all baselines. Specifically, it exceeds the Zero-shot CoT method by 8.2% and shows consistent advantages over automatic prompt optimization baselines including APE (5.5%), OPRO (4.7%), TextGrad (4.0%), and DSPy (2.7%).

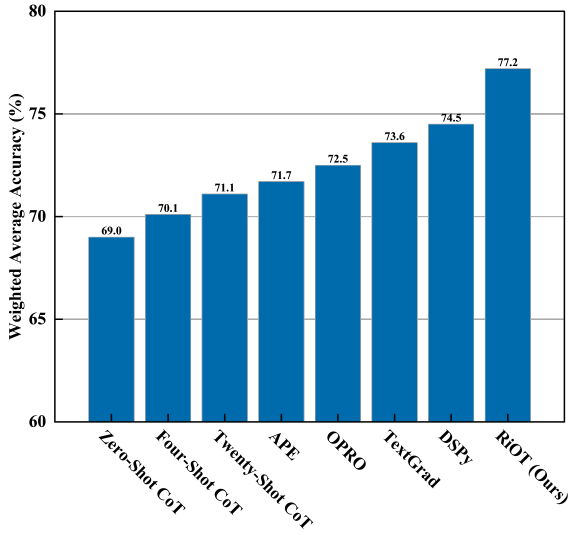

Figure 4: Weighted Average Accuracy (WAA) of different methods across all datasets. RIOT achieves the highest WAA of 77.2%, outperforming all baseline methods.

B Alternative Node Selection Metrics

We consider two alternative metrics to perplexity for scoring candidate prompts: Entropy and Length. Given a prompt sequence $(x_0, x_1, \ldots, x_J)$, the metrics are defined as:

$$\text{Entropy} = -\sum_{j=1}^{J} p_\theta(x_j \mid x_{<j}) \log p_\theta(x_j \mid x_{<j})$$

$$\text{Length} = J$$

where $p_\theta(x_j \mid x_{<j})$ denotes the probability assigned to token x_j given its preceding context.

Method	TextGrad	RiOT (Single)	RiOT (Multi)
Time (s)	989.47	2741.41	1080.56

Table 6: Runtime comparison between TextGrad and RiOT under different threading settings.

C Computational Cost Analysis

In this section, we conduct an explicit analysis of the computational overhead introduced by RIOT. Compared to TextGrad, the additional cost mainly stems from the need to generate K candidate prompts at each optimization step (with $K = 3$ in our experiments) and to perform semantic fusion, which involves computing embeddings and evaluating semantic similarity for residual merging.

Since RIOT adopts TextGrad as its optimization backbone, we compare their runtime to assess the computational complexity. Theoretically, RiOT requires approximately K times the optimization time of TextGrad. In practice, however, this overhead is effectively mitigated by leveraging multithreading, which enables parallel execution of candidate generation steps.

As shown in Table 6, the multithreaded version of RIOT only incurs a modest increase in runtime (approximately 9%) compared to TextGrad, while achieving a notable improvement in accuracy. This demonstrates that the trade-off between performance gain and computational cost is both reasonable and practical for real-world applications.

D Additional Evaluation on a Small-Scale, High-Difficulty Benchmark

To further evaluate the generalization ability of RIOT in complex reasoning scenarios, we conduct an additional experiment on the AMC12 benchmark. AMC12 (American Mathematics Competition 12) is a small-scale but high-difficulty dataset consisting of 83 math problems from the official 2022 and 2023 exams. It requires symbolic manipulation and multi-step reasoning, posing substantial challenges for prompt optimization methods. The dataset is split into training, validation, and test sets in a 20/20/43 ratio. As shown in Table 7, RIOT achieves an accuracy of 46.0%, surpassing all baselines and improving over the zero-shot CoT baseline by +5.1%. These results serve as a further validation of RIOT's robustness and effectiveness in low-resource, high-complexity settings.

Method	Zero-Shot CoT	Four-Shot CoT	Twenty-Shot CoT	APE	OPRO	TextGrad	DSPy	RiOT (Ours)
Accuracy (%)	40.9	37.2	30.7	42.8	42.8	40.9	41.9	**46.0**
Δ	+0.0	-3.7	-10.2	+1.9	+1.9	+0.0	+1.0	**+5.1**

Table 7: Comparion of Automatic Prompt Optimization Methods on the AMC12 benchmark.

E Details of Evaluation Datasets

In this section, we provide detailed information about the five reasoning datasets used in our study. Table 8 summarizes dataset statistics, while Table 11 provides representative examples. To ensure a fair comparison, all experiments follow consistent data sampling protocols: fixed-size development sets with 20 training and 20 validation samples each task. Below we detail each dataset's unique characteristics.

LogiQA 2.0 (Liu et al., 2023) is a complex logical reasoning dataset derived from the Chinese Civil Service Examination. We use its nature language inference subset, which includes 6,871 training samples, 3,807 validation samples, and 3,240 test samples. The problems are categorized into five types: categorical reasoning, sufficient conditional reasoning, necessary conditional reasoning, disjunctive reasoning, and conjunctive reasoning. We sample 160 test samples that cover these five reasoning types from the original dataset.

StrategyQA (Geva et al., 2021) is a commonsense reasoning task that evaluates the model's ability to infer unstated intermediate facts and strategically combine them to derive correct answers, leveraging general world knowledge. The original dataset consists of 2,061 training samples and 229 test samples. Following previous research (Chen et al., 2023; Yuksekgonul et al., 2024; Du et al., 2023; Yao et al., 2024), we sample 100 test samples for evaluation.

Object Counting (Srivastava et al., 2023; Suzgun et al., 2023) measures the model's ability to count objects within a given context while ignoring distractors. This task emphasizes fundamental reasoning over arithmetic by focusing on semantic understanding. The original dataset contains 250 test samples. For our study, we sample 210 test samples for evaluation.

GSM8K (Cobbe et al., 2021) evaluates the model's ability to solve basic mathematical problems requiring multi-step reasoning. This task focuses on contextual understanding and numerical computation. The original dataset consists of 7,473 training samples and 1,319 test samples. Following previous research (Chen et al., 2023; Yuksekgonul et al., 2024; Du et al., 2023; Yao et al., 2024), we sample 100 test samples for evaluation.

Date Understanding (Srivastava et al., 2023; Suzgun et al., 2023) assesses the model's ability to comprehend temporal information, requiring reasoning over explicit or implicit data contexts to identify specific dates. The original dataset contains 369 test samples. For our study, For our study, we sample 329 test samples for evaluation.

Task	Training	Validation	Test
LogiQA 2.0	20	20	160
StrategyQA	20	20	100
Object Counting	20	20	210
GSM8K	20	20	100
Date Understanding	20	20	329

Table 8: Dataset Statistics.

F Details of Manual Prompt Construction

In this section, we provide detailed information about the design and implementation of the zero-shot and few-shot CoT prompts used as baselines in our experiments. Both prompt variants follow a structured template to ensure consistency across tasks while accommodating task-specific requirements. The general template $\mathcal{P}^k_{\text{CoT}}$ is defined as:

$$\mathcal{P}^k_{\text{CoT}} = \mathcal{P}_{\text{task}} + \mathcal{P}_{\text{think}} + \mathcal{P}_{\text{format}} + \{\mathcal{P}^{(i)}_{\text{demo}}\}^k_{i=1} \quad (10)$$

where $\mathcal{P}_{\text{task}}$ defines the precise task definition and the objective, $\mathcal{P}_{\text{think}}$ is the reasoning guidance, $\mathcal{P}_{\text{format}}$ denotes output structure constraints and $\{\mathcal{P}^{(i)}_{\text{demo}}\}^k_{i=1}$ is the demonstration examples. The zero-shot CoT prompts $\{\mathcal{P}^0_{\text{CoT}}\}$ is exemplified in Table 10. The few-shot CoT prompts $\{\mathcal{P}^k_{\text{CoT}}\}$ extend this template by appending demonstrations from the training set, and each demonstration consists of an input-output pair. An example is visualized in Figure 5. These manual prompts represent the most widely adopted approaches in practice and serve as fundamental baselines for evaluating prompt engineering techniques.

Implementation	TRUE / FALSE		GENERATIVE		MULTIPLE-CHOICE
	LogiQA 2.0 (N=160)	StrategyQA (N=100)	Object Counting (N=210)	GSM8K (N=100)	Date Understanding (N=329)
Sentence-based	61.4 ±1.5	74.6 ±1.4	86.9 ±0.6	81.2 ±1.2	78.2 ±0.6
LLM-based	60.1 ±1.4 ↓1.3	72.4 ±1.2 ↓2.2	81.7 ±1.3 ↓5.2	75.4 ±2.8 ↓5.8	80.2 ±0.3 ↑2.0

Table 9: Comparison of performance between two implementations for text residual connection across five tasks. We report accuracy (%) along with standard deviation. Red arrows indicate the absolute performance decrease, while green arrows indicate the absolute performance increase, both compared to the sentence-based implementation. N represents the number of samples used for testing. The sentence-based implementation outperforms the LLM-based implementation.

G An Alternative Implementation of the Text Residual Connection

In our exploration of Text Residual Connection, we experiment with two distinct implementation approaches. The first approach, introduced in the Section 4.2, is the **sentence-based** Implementation, which utilizes sentence tokenization and vector similarity to perform the text residual connection. The second approach, the **LLM-based** Implementation, leverages the strong meta-reasoning capabilities of modern LLMs without explicit lexical analysis. To be specific, we develop a rigorously engineered prompt template to instruct LLMs in performing text residual connection. As shown in Figure 6, the prompt template incorporates four key principles: **(1) Task Specification**: Explicit definition of input-output relationships. **(2) Constraint Enumeration**: Clear operational boundaries. **(3) Exemplar Guidance**: Illustrative examples demonstrating ideal behavior. **(4) Error Prevention**: Anticipating and mitigating common failure modes.

We evaluate the performance of these two implementations across five datasets, using GPT-4o to implement the LLM-based Implementation. The results, presented in Table 9, show that the sentence-based implementation generally outperforms the LLM-based implementation, particularly in tasks involving binary classification (True/False) and generative tasks. Additionally, during the experimental process, we observe that the LLM-based implementation does not always strictly adhere to the task requirements in the same way as the sentence-based implementation. Specifically, the LLMs tend to modify or omit parts of the original content in the prompt during execution. For these reasons, we choose to use the sentence-based implementation for the main experiments in this study, as it provide more consistent and reliable results.

Despite these challenges, the LLM-based implementation remains promising, especially given its end-to-end nature and the lack of the need for hyperparameter tuning. This characteristic makes it easier to integrate directly into existing optimization frameworks, which is a substantial advantage in practical applications. Further development and exploration are required to refine this approach.

Task	Zero-Shot Prompt (Initial Prompt)
LogiQA 2.0	You will solve logical reasoning problems based on the given facts. Think step by step. The last line of your response should be of the following format: "Answer: YES" or "Answer: NO".
StrategyQA	You will answer a commonsense reasoning task. Think step by step. The last line of your response should be of the following format: "Answer: YES" or "Answer: NO".
Object Counting	You will answer a reasoning question. Think step by step. The last line of your response should be of the following format: 'Answer: $VALUE' where VALUE is a numerical value.
GSM8K	You will answer a mathematical reasoning question. Think step by step. The last line of your response should be of the following format: 'Answer: $VALUE' where VALUE is a numerical value.
Date Understanding	You will answer a multiple-choice question related to date understanding. Think step by step. The last line of your response should be of the following format: 'Answer: $VALUE' where VALUE is a single letter.

Table 10: Zero-Shot Prompts (initial prompt for optimization) for various reasoning tasks. Each prompt consists of three components: the task definition $\mathcal{P}_{\text{task}}$, the reasoning guidance $\mathcal{P}_{\text{think}}$, and the output structure constraints $\mathcal{P}_{\text{format}}$.

You will answer a mathemetical reasoning question. Think step by step. The last line of your response should be of the following format: 'Answer: $VALUE' where VALUE is a numerical value.

Here are some examples:
Example 1:
Input: Question: Stefan goes to a restaurant to eat dinner with his family. They order an appetizer that costs $10 and 4 entrees that are $20 each. If they tip 20% of the total for the waiter, what is the total amount of money that they spend at the restaurant?
Output: Answer: 108
Example 2:
Input: Question: 4 friends are running a 4 x 100 relay race. Mary ran first and took twice as long as Susan. Susan ran second and she took 10 seconds longer than Jen. Jen ran third and finished in 30 seconds. Tiffany ran the last leg and finished in 7 seconds less than Mary. How many seconds did it take the team to finish the race?
Output: Answer: 223
Example 3:
Input: Question: Arnold owns three cars. The first car averages 50 miles per gallon of gas. The second car averages 10 miles per gallon of gas. And the third car averages 15 miles per gallon of gas. He splits his 450-mile monthly driving mileage equally amongst his three cars. If gas costs $2 per gallon, how much does he spend on gas each month?
Output: Answer: 56
Example 4:
Input: Question: The gauge on a water tank shows that the tank is 1/3 full of water. To fill the tank, 16 gallons of water are added. How many gallons of water does the tank hold when full?
Output: Answer: 24

Figure 5: An example of few-shot prompts from GSM8K. The purple section represent the demonstration examples $\{\mathcal{P}_{\text{demo}}^{(i)}\}_{i=1}^{k}$. The examples are drawn from the training set.

Task	Example Problem
LogiQA 2.0	**Input**: *Given the fact*: Balance is particularly important when (...) If all the media were to adopt such a perverse interpretation of balanced reporting, the public would be given a picture of a world where each party in every conflict had an equal measure of justice on its side, contrary to our experience of life and, indeed, our common sense. *Does it follow that*: The main point of the argument is that balanced reporting requires impartially revealing injustices where they occur no less than fairly presenting the views of each party in a conflict. **Ground Truth**: Yes.
StrategyQA	**Input**: *Question*: Are months based on the solar cycle? **Ground Truth**: Yes.
Object Counting	**Input**: *Question*: I have a fridge, an oven, a car, a toaster, a microwave, a table, and a bed. How many objects do I have? **Ground Truth**: 7.
GSM8K	**Input**: *Question*: Stefan goes to a restaurant to eat dinner with his family. They order an appetizer that costs $10 and 4 entrees that are$20 each. If they tip 20% of the total for the waiter, what is the total amount of money that they spend at the restaurant? **Ground Truth**: 108.
Date Understanding	**Input**: *Question*: In the US, Thanksgiving is on the fourth Thursday of November. Today is the US Thanksgiving of 2001. What is the date today in MM/DD/YYYY? *Choices*: A. 01/16/2003 B. 11/21/2002 C. 09/04/2002 D. 08/24/2002 E. 11/22/2002 F. 11/23/2002 **Ground Truth**: E.

Table 11: Example problems and their corresponding ground truths for various reasoning tasks. The tasks cover a range of domains, including logical reasoning (LogiQA 2.0), commonsense reasoning (StrategyQA), semantic understanding (Object Counting), mathematical reasoning (GSM8K), and temporal reasoning (Date Understanding).

```
Please perform a text comparison and merging task based on the following requirements:

### Input
1. **Text A**: The original text content that needs to be updated.
2. **Text B**: The reference text content containing additional information.

### Output
- Return an updated version of Text A that meets the following conditions:
  1. **Retain similar sentences**: If a sentence in Text A has a similar counterpart in Text B, retain the sentence from Text A.
  2. **Add new sentences**: Include sentences from Text B that do not have a similar counterpart in Text A.
  3. **Avoid redundancy**: Ensure there are no duplicate sentences in the final output.
  4. **Maintain coherence**: The resulting text should be logically consistent and the sentence order should be clear and natural.

### Example
**Input**:
- Text A: "The sky is blue. Birds are flying."
- Text B: "Birds are flying in the sky. The grass is green."

**Output**:
"The sky is blue. Birds are flying. The grass is green."

### Execution Steps
1. Split Text A and Text B into sentences.
2. Compare each sentence in Text A with sentences in Text B. If a similar sentence exists in Text B, retain the one from Text A.
3. Identify sentences in Text B that do not have a similar counterpart in Text A and add them to the updated text.
4. Combine the updated sentences into a coherent and complete text.

### Task Objective
Please complete the text updating task in a clear and logically consistent manner, and return the final updated text.
```

Figure 6: The prompt template for the LLM-based Implementation of Text Residual Connection.