Data-Augmented Prompt Optimization for LLMs

Anonymous ACL submission

Abstract

With the rapid advancement of Large Language Models (LLMs), their performance has become increasingly dependent on prompt quality, making prompt optimization a critical research area. However, existing techniques often face limitations due to the lack of diverse and challenging data necessary for fully testing and refining prompt effectiveness. To address this challenge, we propose a novel approach that integrates data augmentation into the prompt optimization process, leveraging the synergy between synthetic data generation and prompt optimization. Our framework centers on prompt optimization as the key driver of LLM performance, integrating data augmentation as a means to enhance this process. We introduce a multi-agent system consisting of an auto prompt optimization agent and a synthetic data generation agent, where the former serves as the core component, iteratively refining prompts through a closed-loop feedback mechanism. This dynamic interplay enables the LLM to engage in self-reflection and multi-step reasoning, structurally improving prompt quality over time. Our results demonstrate that enhancing prompt optimization with data augmentation significantly improves model performance. Our approach surpasses existing methods in classification, question answering, and reasoning tasks across multiple benchmarks.

1 Introduction

Large language models (LLMs) have achieved excellent performance in benchmark tasks such as classification cation, question answering, and reasoning tasks, but they depend heavily on prompts construction. Minor adjustments in prompt phrasing or formatting can lead to noticeable changes in model outputs, which highlights the importance of careful prompt design when harnessing the capabilities of these models.

Recent studies have addressed prompt optimization through manual tuning, gradient-based strategies, and automated search (Wang et al., 2023; Sun et al., 2024; Gou et al., 2023). These methods aim to make LLM responses more reliable and consistent by improving prompt structures. Despite these advances, most existing approaches focus on prompt refinement without systematically exploring how additional data might support or guide the optimization process.

Interestingly, we notice that in earlier deep learning research, data augmentation often leads to significant performance boost. However, within LLM prompt engineering, generating new data instances has not been widely explored. We propose that prompt optimization can benefit from an augmentation-based approach, especially as LLMs themselves are capable of producing synthetic text with contextual relevance (Singh et al., 2023; Gilardi et al., 2023; Tang et al., 2023; Gao et al., 2023).

To demonstrate this, we propose a multi-agent system that unifies synthetic data generation with automatic prompt optimization. Specifically, the data generation agent employs iterative refinement techniques and synthetic data to enhance prompt-based autonomous learning. It generates challenging edge cases, which trigger the auto prompt agent to engage in a closed-loop feedback mechanism, enabling continuous optimization.

With these key insights and prior works (Singh et al., 2023; Gilardi et al., 2023; Tang et al., 2023; Gao et al., 2023; Wang et al., 2023; Sun et al., 2024; Gou et al., 2023), our contributions are as follows:

- We are the first to incorporate data augmentation into prompt optimization and verify its performance through empirical evidence.
- We propose a data generation agent-driven framework to enhance the diversity and quality of synthetic data. By seamlessly integrating the data generation agent with automated

prompt engineering, our unified system fosters efficient synergy, leading to substantial performance improvements.

- We have conducted benchmark experiments to validate the performance and stability of our proposed system and tested it across multiple state-of-the-art models.

In the following sections, we present the system's architecture and implementation, detailing how iterative prompt optimization enhances the performance of LLM, and present the benchmark results.

2 Related Work

2.1 Automatic prompt engineering

Automatically discovering optimal prompts has emerged as a central challenge in the era of LLMs. Automatic Prompt Engineering (APE) employs a variety of optimization-based, generative, and template-driven approaches to refine and enhance prompts for large language models (LLMs). Optimization-based methods encompass techniques such as gradient-based objective optimization (Shin et al., 2020), reinforcement learning with reward functions (e.g., RLHF), and evolutionary algorithms that iteratively improve prompt quality. Generative approaches leverage models like GPT or Gemini to generate candidate prompts and evaluate them using automated metrics. Additionally, zero-shot and few-shot learning paradigms enable the creation of effective prompts even with minimal labeled data. Template-driven approaches, on the other hand, rely on structured formats—such as fill-in-the-blank templates—and apply grammatical constraints to ensure clarity, correctness, and consistency.

Building upon these foundations, we propose a novel framework that integrates LLM-driven rewriting with natural language feedback (Pryzant et al., 2023), alongside mechanisms for self-reflection (Shinn et al., 2024) and planning (Wang et al., 2023). This hybrid approach enhances the adaptability and precision of prompts by leveraging iterative refinement processes informed by the model's own assessments and external feedback. Our framework aims to address limitations in existing methods, offering a robust and flexible solution for advancing prompt optimization.

2.2 Data synthesis

Using large language models (LLMs) for data synthesis is a relatively new and rapidly evolving approach. Recent advancements have shown that LLMs possess the capability to generate text with fluency and quality comparable to human output (Li et al., 2023; Eldan and Li, 2023). As a result, the application of LLMs for data synthesis has garnered significant attention (Mukherjee et al., 2023; Li et al., 2023; Eldan and Li, 2023). For instance, prior work (Gao et al., 2022) has explored leveraging pre-trained language models (PLMs) to generate task-specific text data. However, these studies have not fully incorporated advanced methodologies such as chain-of-thought (CoT) reasoning, in-context learning, or data synthesis driven by prompts that integrate task descriptions and label information.

In this study, we systematically experimented with a range of techniques, including in-context learning and prompt-driven data synthesis, combining task descriptions and label information. Our results indicate that integrating these approaches produces higher-quality synthetic data compared to traditional methods. To further enhance the robustness and applicability of the synthetic data, we introduced corner cases, making the generated data more challenging and reflective of complex real-world scenarios. These findings highlight the potential of combining advanced LLM capabilities with tailored prompting strategies to improve data synthesis quality and reliability for prompt optimization.

3 Method

3.1 System Overview

The system adopts a multi-agent mode, dividing the entire closed-loop system into two parts: a data generation agent and an auto prompt agent. Fig 1 shows the general flow of the system.

3.1.1 Data Generation Agent

We propose a new synthetic data generation framework driven by data augmentation, where data augmentation serves as the core mechanism to trigger a series of mutually supporting processes. These processes collectively enhance the complexity,diversity, and usability of synthetic data, moving beyond traditional approaches that simply apply multiple augmentation methods in isolation.

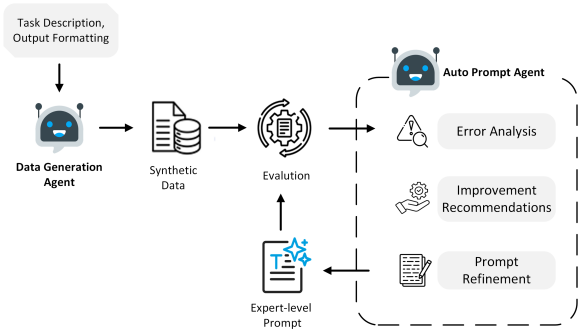

Figure 1: The Data Generation Agent synthesizes data and passes it to the Auto Prompt Agent with specific task evaluation results, which summarizes the correctness of the responses to the synthesized data based on the previous iteration. The prompt is then optimized accordingly, and this process continues iteratively.

At the core of the framework is the ability to generate high-quality synthetic data with minimal prior information. By leveraging few-shot learning, the framework identifies and reproduces diverse situational features through interaction with contextual dialogue and task descriptions. This is achieved by conditioning the generation process on a few-shot dataset $S = \{(x_i, y_i)\}_{i=1}^{M}$ ($M \ll N$), where the model learns to generate new samples x' that align with the desired label distribution $p^*(y)$. The generation process is formalized as:

$$p(x'|y, S) = \int p(x'|z, y)p(z|S)\, dz$$

Here, z represents a latent variable that captures the underlying structure of the few-shot data, enabling the model to generate diverse and realistic samples even with limited initial examples.

To ensure that the generated data adheres to specific task requirements and label constraints, the framework incorporates label and task constraints directly into the generation process. This is achieved by minimizing the Kullback-Leibler (KL) divergence between the generated label distribution $p(y)$ and the target distribution $p^*(y)$, while also optimizing for task-specific performance. The optimization objective is formalized as:

$$\min_{p(x'|y)} \mathrm{KL}(p(y)\|p^*(y)) + \lambda \mathbb{E}_{x',y}[\mathcal{L}(f_\theta(x'), y)]$$

Here, $\mathcal{L}(\cdot, \cdot)$ is the task-specific loss function, and λ is a trade-off parameter that balances label consistency and task performance. This ensures that the generated data is not only diverse but also aligned with the desired task requirements.

The framework further enhances the realism and domain relevance of the generated data by incorporating role-playing into the generation process. By simulating different users, scenarios, or professional perspectives, the model generates data from multiple viewpoints, enriching the expressive dimensions of the synthetic data. This is achieved by introducing a role distribution $p(r)$ and conditioning the generation process on specific roles:

$$x' = f_\theta^r(x), \quad r \sim p(r)$$

Here, f_θ^r represents the generative model for role r, and $p(r)$ controls the distribution of roles. To ensure diversity and realism, adversarial training is employed:

$$\min_\theta \max_\phi \mathbb{E}_{x,r}[\log D_\phi(x, r)] + \mathbb{E}_{x',r}[\log(1 - D_\phi(x', r))]$$

Here, D_ϕ is a discriminator that distinguishes between real and generated data, ensuring that the synthetic data is both diverse and realistic.

To address tasks of varying complexity, the framework introduces a progressive complexity mechanism. By dynamically adjusting semantic cues, reasoning levels, and problem difficulty, the generated data can range from simple recognition tasks to advanced reasoning and cross-domain analysis. This is achieved through a complexity parameter c, which controls the generation process:

$$x' = f_\theta(x, c), \quad c \sim p(c)$$

Here, $p(c)$ is the complexity distribution, which can be adjusted based on task requirements. A sequence of data with increasing complexity can be generated as:

$$\begin{aligned} x'_1 &= f_\theta(x, c_1), \\ x'_2 &= f_\theta(x'_1, c_2), \\ &\vdots \\ x'_L &= f_\theta(x'_{L-1}, c_L) \end{aligned}$$

where $c_1 < c_2 < \cdots < c_L$. This progressive approach ensures that the generated data is suitable for a wide range of tasks, from basic to advanced.

3.1.2 Auto Prompt Agent

In the automatic prompt engineering process, we integrate the rewriting capabilities of a large language model (LLM) with a self-reflection and planning mechanism (Brown et al., 2020; Shinn et al., 2024)

to enable systematic and iterative refinement of prompts. This approach allows the LLM to analyze error patterns, diagnose potential failure points, and iteratively optimize the prompt formula to improve performance. By combining self-assessment and a structured feedback loop, the LLM can identify and mitigate recurring errors, ensuring continuous improvement and standardization of prompts.

The main goal of adopting these techniques is to improve the adaptability, reliability, and efficiency of the LLM in different applications. By enabling the model to dynamically refine its own prompts, we reduce the reliance on manual prompt engineering, simplify task-specific tuning, and improve the overall automation of complex reasoning and structured generation tasks. This not only improves accuracy and consistency, but also minimizes human intervention, making the system more scalable and robust for real-world applications.

To evaluate the effectiveness of each prompt iteration, we conducted a multi-stage validation process using a synthetically generated dataset. If the model produces an incorrect output, it performs causal analysis to identify flaws in the prompt structure and then improves the formula accordingly. This iterative feedback-driven optimization strategy facilitates robust prompt engineering, enabling the system to dynamically adapt to task complexity, domain-specific constraints, and evolving performance benchmarks.

Ultimately, the prompt variant with the highest performance score (determined by a system evaluation metric) is selected, ensuring that the final prompt formulation maximizes task accuracy, consistency, and generality. By automatically improving the optimized prompts, this approach not only improves the effectiveness of LLM in automated reasoning, structured generation, and domain-specific problem solving, but also contributes to the broader goal of making AI systems more interpretable, self-improving, and efficient at adapting to new challenges.

3.2 System Implementation

3.2.1 Escape Mechanism

Since synthetic data is utilized, ensuring the accuracy of all generated questions and answers is inherently challenging. To mitigate the propagation of erroneous data, a skip mechanism is implemented. If three consecutive errors occur during generation, the corresponding question is bypassed. This mechanism prevents the accumulation of flawed data that could mislead the language model, thereby reducing the likelihood of generating suboptimal prompts. Furthermore, by incorporating a self-reflection and adaptive filtering mechanism, the system can analyze error patterns and dynamically adjust the prompt refinement process. This enhances the robustness and reliability of the framework, ensuring more stable and high-quality prompt optimization.

3.2.2 Optimal path strategy

An optimal path strategy is introduced to systematically refine and enhance prompt effectiveness. This strategy involves selecting the highest-performing prompt from the set of generated prompts during each iteration. The selected prompt is then further optimized through iterative refinement using the synthetic dataset.

By iteratively refining the optimal prompt, this approach guarantees continuous improvement in prompt quality. The strategy also leverages reinforcement learning-inspired optimization techniques, where feedback from previous iterations is used to guide the selection and modification of future prompts. This enables the framework to dynamically adapt to task-specific complexities, thereby maximizing both effectiveness and adaptability.

4 Experiments

4.1 Datasets

To comprehensively evaluate our system, we selected nine datasets, enabling a multi-perspective assessment. These tasks emphasize the integration of domain-specific knowledge (e.g., geometric shapes and causal reasoning) with advanced cognitive and reasoning abilities (e.g., interpreting penguin-related data in tabular formats, object counting, cognitive reasoning, and time-series analysis). This diverse selection ensures a well-rounded evaluation of the system's capabilities in handling both domain-specific and general reasoning challenges.

1. **BigBech:** BIG-Bench (Srivastava et al., 2022) is a broad benchmark dataset and framework designed to evaluate the performance of large language models (LLMs). It was developed by multiple research institutions and individuals to evaluate the model's capabilities on a range of complex tasks. These tasks are

usually areas that existing benchmark datasets cannot fully cover, involving reasoning, common sense understanding, creativity, ethics, and many other aspects.We used 3200 logic questions in BIG-Bench, including Penguins, Geometry, Epistemic, Object Count, Temporal, Causal Judge and PrOntoQA.

2. **ProofWriter:**ProofWriter(Tafjord et al., 2021) is a dataset based on Natural Language Inference (NLI) that is used to evaluate the reasoning and deduction capabilities of language models, especially the ability to derive logical proofs. ProofWriter constructs reasoning tasks through logical rules, requiring the model to generate logical proofs or verify the correctness of the conclusions. We randomly sampled 600 pieces of data.

3. **FOLIO:**FOLIO(Han et al., 2024) (First Order Logic Inference Over Natural Language) is a reasoning dataset based on first-order logic, which aims to evaluate the model's comprehensive understanding of logical rules and natural language reasoning.We used 204 examples in FOLIO as the test set.

4.2 EXPERIMENTAL SETUP

4.2.1 Baselines

We compare our method with four other types of methods: Chain of Thoughts (CoT) prompts, Automatic Prompt Engineer (APE), PromptAgent, and Neuro-Symbolic.

Chain of Thought (CoT)is a method designed to improve the reasoning ability of large language models (LLMs). It simulates the logical reasoning process of humans, guides the model to gradually decompose complex problems, and gives more accurate and explanatory answers. The CoT method usually shows significant advantages in tasks such as mathematical calculations, logical reasoning, and multi-step reasoning (Suzgun et al., 2022).

Automatic Prompt Engineer(APE) (Wang et al., 2023) is designed to automatically generate and optimize natural language instructions for large language models (LLMs) to enhance task performance. APE begins by leveraging the model to generate candidate instructions, evaluates their effectiveness based on the execution results of the target model, and iteratively refines instruction quality using the Monte Carlo search method. This approach minimizes manual intervention, significantly improving efficiency and adaptability across various tasks compared to manually crafted prompts. Experiments conducted on 24 NLP tasks demonstrate that instructions generated by APE perform exceptionally well in both zero-shot and few-shot learning scenarios, highlighting its broad applicability and effectiveness.

PromptAgentPromptAgent, developed by (Zhou et al., 2022b), aims to automatically generate high-quality prompts that are comparable to those produced by experts. It frames prompt optimization as a strategic planning problem, utilizing a Monte Carlo Tree Search (MCTS) algorithm to systematically explore the high-level prompt space. Drawing inspiration from human trial-and-error processes, PromptAgent refines prompts by distilling precise expert insights and generating detailed, constructive feedback based on model errors.

Neuro-Symbolic The Neuro-Symbolic method we use is proposed by (Pan et al., 2023), which is a hybrid method that combines neural networks with symbolic reasoning, particularly in scenarios that demand a combination of advanced perception capabilities and robust logical reasoning. It has been widely adopted in benchmark problems of logical reasoning (Ying et al., 2023; Zhang et al., 2023), where the text-processing capabilities of large language models (LLMs) are utilized to efficiently and accurately transform natural language data into symbolic representations for solution, demonstrating its versatility and effectiveness in handling complex reasoning challenges.

4.2.2 Implementation details

Data Generation Agent for Synthetic Data Creation The Data Generation Agent employs structured output to standardize the format of synthetic data, ensuring consistency across all generated samples. To introduce challenging data, a difficulty gradient is implemented to gradually increase the complexity of the synthetic data, thereby enhancing the fine-tuning of prompts in the automatic prompting stage.

During the synthetic data generation process, several challenges arise, particularly in geometric data synthesis:

Parsing Challenges in SVG Path Data: SVG path data is typically represented using floating-point numbers, which may introduce parsing ambiguities, making it difficult for LLMs to accurately interpret and recognize shapes.

Complex Shape Generation: Constructing complex geometric shapes, including irregu-

lar curves and polygons, cannot be effectively achieved using simple rule-based methods, necessitating the adoption of more advanced synthesis techniques. To address these challenges, we provide examples of the generated synthetic data in Appendix B and design a Data Generation Agent based on the Retrieval-Augmented Generation (RAG) (Lewis et al., 2020) framework.

The Data Generation Agent integrates a retriever module to facilitate the search and synthesis of challenging yet structurally accurate SVG paths. Additionally, a reverse-generation strategy is employed, where answers are synthesized first, followed by the generation of corresponding questions. This approach ensures strong consistency and alignment between questions and answers, thereby enhancing the coherence and reliability of the synthetic dataset.

To ensure the generation of high-quality synthetic data, GPT-4o and GPT-4o-mini are utilized due to their proven effectiveness in text synthesis. To further improve accuracy and reliability, a hybrid approach is adopted, combining in-context learning (ICL) with label constraints: Facilitates few-shot learning, improving data diversity and logical consistency in the generated samples. Label Constraints: Define task-specific boundaries, constraining outputs within predefined parameters to maintain structural integrity. Difficulty Gradient Mechanism for Progressive Data Complexity A difficulty gradient mechanism is introduced to systematically increase the complexity of the generated data, progressively challenging prompts with increasingly intricate scenarios. This structured progression (Liu et al., 2024) enhances the robustness and adaptability of the prompt optimization process.

By integrating adaptive difficulty scaling, logical consistency reinforcement, and strategic synthesis methodologies, the proposed framework provides a scalable and effective approach for generating diverse, high-quality synthetic data. This enables systematic prompt refinement, ultimately improving the performance of language models across a wide range of task domains.

Prompt Generation process is designed as an iterative optimization algorithm, enabling the LLM to refine prompts step by step through a structured three-phase approach:

Error Analysis: The LLM identifies and summarizes the root causes of errors by analyzing incorrect answers generated for the given tasks. This phase ensures a precise understanding of the shortcomings in the current prompt.

Improvement Recommendations: Based on the identified errors, the LLM generates targeted improvement suggestions. These recommendations address specific weaknesses in the prompt, such as ambiguity, lack of clarity, or insufficient context, ensuring a more robust and effective design.

Prompt Refinement and Validation: The prompt is updated according to the improvement suggestions, creating a revised version. The LLM then uses the generated synthetic data to test the performance of the updated prompt. By evaluating its effectiveness on the synthetic data, the system determines whether the modifications result in meaningful improvements.

This iterative framework functions as a feedback-driven optimization loop, systematically refining prompts to enhance their clarity, adaptability, and effectiveness. By leveraging synthetic data for validation, the algorithm ensures robust testing and continuous optimization, making it highly adaptable to diverse tasks and applications.

4.3 Results and analyze

4.3.1 Synthetic Data Generation

The synthetic data results generated by the data generation agent are shown in the text box below, highlighting the exceptional advantages of Structured Output. They ensure a highly consistent data format with a well-organized structure and clear hierarchical distinctions. Each output adheres strictly to predefined standards, guaranteeing data stability and integrity while significantly reducing the cost of subsequent cleaning and adjustments. Additionally, the structured format enhances readability and parsability, streamlining analysis and processing while improving data reusability and reliability.

Furthermore, the synthesized data presents a high level of challenge, which is instrumental in refining the Auto Prompt Agent. By generating complex and diverse scenarios, the data pushes the agent to iteratively enhance the prompt, improving its ability to handle a wide range of difficulties effectively. This ensures that the prompt becomes more adaptable and robust in addressing various question complexities. These results strongly validate the effectiveness of Structured Output in standardizing data generation and improving overall data quality, while also fostering continuous ad-

vancements in prompt optimization, ultimately laying a solid foundation for future applications.

```
"target_scores": {
"rectangle": 1,
"sector": 0,
"triangle": 0,
"circle": 0,
"heptagon": 0,
"hexagon": 0,
"kite": 0,
"line": 0,
"octagon": 0,
"pentagon": 0
}
Generated SVG Path
<path d="M 23.45,45.78 L
78.32,45.78 L 78.32,78.56 L
23.45,78.56 L 23.45,45.78"/>
```

4.3.2 Comparison with various prompting baselines

We tested our methods by different LLMs such as GPT-4o, GPT-4o-mini, Gemini-1.5-flash, and Gemini-1.5-pro on different datasets, including BIG-Bench, ProofWriter, and FOLIO.

BIG-Bench. We evaluated our method on key BIG-Bench tasks, including Penguins, Geometry, Epistemic Reasoning, Object Counting, Temporal Reasoning, and Causal Judgment. As shown in Table 1 and Table 2, GPT-4o and GPT-4o-mini demonstrate particularly strong performance in Temporal Reasoning, Object Counting, and Causal Judgment. While Geometry exhibits comparable accuracy across GPT-4o and GPT-4o-mini, our method achieves the highest overall accuracy for GPT-4o-mini, Gemini-1.5-flash, and Gemini-1.5-pro. For GPT-4o, our method performs at a near-best level, trailing PromptAgent by only 0.01, yet still demonstrating that LLMs benefit from synthetic data generation for reasoning improvements, whereas other methods primarily rely on existing datasets. These results further highlight the advantages of LLM-driven data augmentation in enhancing logical, numerical, and causal reasoning capabilities.

FOLIO, PrOntoQA, and ProofWriter. We evaluated our method on FOLIO, PrOntoQA, and ProofWriter using GPT-4o and GPT-4o-mini, assessing the models' ability to perform structured logical reasoning. As shown in Table 3, our method achieves the highest overall F1 score and outperforms all other approaches on FOLIO and PrOntoQA. In PrOntoQA, our approach surpasses all methods, demonstrating its capability to generate structured logical proofs. Similarly, for FOLIO, our method outperforms both CoT and the neuro-symbolic baseline, further validating its effectiveness in formal logic inference.

While neuro-symbolic reasoning remains the best performer on ProofWriter, our method achieves highly competitive results on ProofWriter, trailing by only 0.004 on GPT-4o-mini and 0.02 on GPT-4o, underscoring its strong adaptability to structured reasoning tasks. Crucially, unlike neuro-symbolic approaches that rely on predefined rule-based datasets, our method is trained entirely on LLM-generated synthetic data, demonstrating the effectiveness of LLM-driven data augmentation for enhancing logical inference across diverse reasoning benchmarks.

4.3.3 Prompt generalization

The whole process of the prompt improvement on the Penguin data is shown in Appendix B. Follow the three steps of Error Analysis, Summary of Aspects for Improvement, and Improvement Recommendations: Revised Prompt. The prompts generated by our method outperform other baselines in both structure and result accuracy, validating the effectiveness of our approach in optimizing prompt design and enhancing overall performance. By employing a LEAST-TO-MOST(Zhou et al., 2022a) approach in prompt generation, the LLM to reason in a structured and incremental manner. After each iteration, the prompts are evaluated using real datasets. If the accuracy falls below a predefined threshold, the prompt is refined based on incorrectly answered questions to enhance its effectiveness. All prompts can be viewed in Appendix A

4.4 Evaluation Metrics

We evaluate the quality of synthetic data primarily through manual review and the performance of prompts trained on the synthetic data. The key metric for assessing prompt effectiveness is the F1 score, which we define as the percentage of generated outputs that match the expected results. As a balanced measure of both precision and recall, the F1 score provides a comprehensive evaluation of prompt quality, ensuring that the system not only generates accurate responses but also effectively

Table 1: Comparison of **BIG-Bench** tasks performance across GPT-4o and GPT-4o-mini.

	GPT-4o				GPT-4o-mini			
Tasks	CoT	APE	PromptAgent	Ours	CoT	APE	PromptAgent	Ours
Penguins	0.798	0.848	0.961	**0.964**	0.758	0.837	0.898	**0.921**
Geometry	0.791	0.653	**0.830**	0.822	0.686	0.445	0.720	**0.732**
Epistemic	0.793	0.848	**0.916**	0.863	0.852	0.816	**0.860**	0.851
Object Count	0.852	0.860	0.882	**0.911**	0.815	0.863	0.843	**0.875**
Temporal	0.980	0.992	0.984	**0.993**	0.949	0.972	0.946	**0.980**
Causal Judge	0.678	0.740	0.778	**0.790**	0.636	0.756	0.846	**0.880**
Average	0.815	0.824	**0.892**	0.891	0.783	0.782	0.852	**0.873**

Table 2: Comparison of **BIG-Bench** tasks performance across Gemini-1.5-flash and Gemini-1.5-pro.

	Gemini-1.5-flash				Gemini-1.5-pro			
Tasks	CoT	APE	PromptAgent	Ours	CoT	APE	PromptAgent	Ours
Penguins	0.704	0.376	0.674	**0.773**	**0.818**	0.402	0.736	0.793
Geometry	0.683	0.494	**0.703**	0.689	0.591	0.566	0.583	**0.643**
Epistemic	0.855	**0.888**	0.816	0.870	0.826	0.887	0.838	**0.893**
Object Count	0.901	0.847	0.863	**0.923**	**0.928**	0.786	0.726	0.913
Temporal	0.940	0.994	0.942	**0.984**	0.989	0.860	**0.984**	0.980
Causal Judge	0.668	0.694	0.679	**0.732**	0.615	0.657	0.742	**0.783**
Average	0.792	0.716	0.780	**0.829**	0.795	0.693	0.768	**0.834**

Table 3: Comparison of methods on ProofWriter, FOLIO, and PrOntoQA across GPT-4o and GPT-4o-mini.

	GPT-4o				GPT-4o-mini			
Tasks	Baseline	Neuro-symbolic	CoT	Ours	Baseline	Neuro-symbolic	CoT	Ours
ProofWriter	0.585	**0.816**	0.723	0.796	0.526	**0.797**	0.618	0.793
FOLIO	0.712	0.792	0.726	**0.832**	0.512	0.732	0.693	**0.811**
PrOntoQA	0.804	0.852	0.956	**0.963**	0.746	0.793	0.893	**0.913**
Average	0.700	0.820	0.802	**0.864**	0.595	0.774	0.735	**0.839**

captures all relevant information. Its robustness makes it an essential indicator of overall performance and reliability.

5 Conclusion

Our research results demonstrate that the designed system can efficiently leverage the Data Generation Agent and Auto Prompt Agent to achieve automated prompt generation based on high-quality and diverse synthetic data. Through an advanced Strategic Data Augmentation Framework, the LLM can generate more diverse and high-quality data, providing a rich and reliable data foundation for automated prompt engineering. This system not only enhances the automation of prompt generation but also improves the applicability and generalization of data in complex tasks. Our research offers a scalable, adaptive, and data-efficient solution for real-world AI applications requiring strong logical reasoning, further advancing the application of large models in intelligent reasoning and task automation.

Limitations

Our multi-agent system efficiently generates synthetic data based on user requirements and autonomously iterates to optimize high-quality task-solving prompts. However, the system still has certain limitations. Currently, our evaluation is limited to standard benchmarks. Future work could extend the validation to real-world datasets, such as financial and medical data, to comprehensively assess the system's effectiveness and adaptability.

References

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.

Ronen Eldan and Yuanzhi Li. 2023. Tinystories: How small can language models be and still speak coherent english? *Preprint*, arXiv:2305.07759.

Jiahui Gao, Renjie Pi, Yong Lin, Hang Xu, Jiacheng Ye, Zhiyong Wu, Weizhong Zhang, Xiaodan Liang, Zhenguo Li, and Lingpeng Kong. 2022. Self-guided noise-free data generation for efficient zero-shot learning. *arXiv preprint arXiv:2205.12679*.

Jiahui Gao, Renjie Pi, Yong Lin, Hang Xu, Jiacheng Ye, Zhiyong Wu, Weizhong Zhang, Xiaodan Liang, Zhenguo Li, and Lingpeng Kong. 2023. Self-guided noise-free data generation for efficient zero-shot learning. *Preprint*, arXiv:2205.12679.

Fabrizio Gilardi, Meysam Alizadeh, and Maël Kubli. 2023. Chatgpt outperforms crowd workers for text-annotation tasks. *Proceedings of the National Academy of Sciences*, 120(30):e2305016120.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Minlie Huang, Nan Duan, and Weizhu Chen. 2023. Tora: A tool-integrated reasoning agent for mathematical problem solving. *arXiv preprint arXiv:2309.17452*.

Simeng Han, Hailey Schoelkopf, Yilun Zhao, Zhenting Qi, Martin Riddell, Wenfei Zhou, James Coady, David Peng, Yujie Qiao, Luke Benson, Lucy Sun, Alex Wardle-Solano, Hannah Szabo, Ekaterina Zubova, Matthew Burtell, Jonathan Fan, Yixin Liu, Brian Wong, Malcolm Sailor, Ansong Ni, Linyong Nan, Jungo Kasai, Tao Yu, Rui Zhang, Alexander R. Fabbri, Wojciech Kryscinski, Semih Yavuz, Ye Liu, Xi Victoria Lin, Shafiq Joty, Yingbo Zhou, Caiming Xiong, Rex Ying, Arman Cohan, and Dragomir Radev. 2024. Folio: Natural language reasoning with first-order logic. *Preprint*, arXiv:2209.00840.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474.

Yifei Li, Zeqi Lin, Shizhuo Zhang, Qiang Fu, Bei Chen, Jian-Guang Lou, and Weizhu Chen. 2023. Making language models better reasoners with step-aware verifier. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5315–5333.

Michael Xieyang Liu, Frederick Liu, Alexander J Fiannaca, Terry Koo, Lucas Dixon, Michael Terry, and Carrie J Cai. 2024. " we need structured output": Towards user-centered constraints on large language model output. In *Extended Abstracts of the CHI Conference on Human Factors in Computing Systems*, pages 1–9.

Subhabrata Mukherjee, Arindam Mitra, Ganesh Jawahar, Sahaj Agarwal, Hamid Palangi, and Ahmed Awadallah. 2023. Orca: Progressive learning from complex explanation traces of gpt-4. *arXiv preprint arXiv:2306.02707*.

Liangming Pan, Alon Albalak, Xinyi Wang, and William Yang Wang. 2023. Logic-lm: Empowering large language models with symbolic solvers for faithful logical reasoning. *arXiv preprint arXiv:2305.12295*.

Reid Pryzant, Dan Iter, Jerry Li, Yin Tat Lee, Chenguang Zhu, and Michael Zeng. 2023. Automatic prompt optimization with "gradient descent" and beam search. *Preprint*, arXiv:2305.03495.

Taylor Shin, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, and Sameer Singh. 2020. Autoprompt: Eliciting knowledge from language models with automatically generated prompts. *Preprint*, arXiv:2010.15980.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2024. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36.

Avi Singh, John D Co-Reyes, Rishabh Agarwal, Ankesh Anand, Piyush Patil, Xavier Garcia, Peter J Liu, James Harrison, Jaehoon Lee, Kelvin Xu, et al. 2023. Beyond human data: Scaling self-training for problem-solving with language models. *arXiv preprint arXiv:2312.06585*.

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. 2022. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *arXiv preprint arXiv:2206.04615*.

Hongda Sun, Weikai Xu, Wei Liu, Jian Luan, Bin Wang, Shuo Shang, Ji-Rong Wen, and Rui Yan. 2024. Determlr: Augmenting llm-based logical reasoning from indeterminacy to determinacy. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9828–9862.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V Le, Ed H Chi, Denny Zhou, et al. 2022. Challenging big-bench tasks and whether chain-of-thought can solve them. *arXiv preprint arXiv:2210.09261*.

Oyvind Tafjord, Bhavana Dalvi Mishra, and Peter Clark. 2021. Proofwriter: Generating implications,

proofs, and abductive statements over natural language. *Preprint*, arXiv:2012.13048.

Changli Tang, Wenyi Yu, Guangzhi Sun, Xianzhao Chen, Tian Tan, Wei Li, Lu Lu, Zejun Ma, and Chao Zhang. 2023. Salmonn: Towards generic hearing abilities for large language models. *arXiv preprint arXiv:2310.13289*.

Xinyuan Wang, Chenxi Li, Zhen Wang, Fan Bai, Haotian Luo, Jiayou Zhang, Nebojsa Jojic, Eric P Xing, and Zhiting Hu. 2023. Promptagent: Strategic planning with language models enables expert-level prompt optimization. *arXiv preprint arXiv:2310.16427*.

Lance Ying, Katherine M Collins, Megan Wei, Cedegao E Zhang, Tan Zhi-Xuan, Adrian Weller, Joshua B Tenenbaum, and Lionel Wong. 2023. The neuro-symbolic inverse planning engine (nipe): Modeling probabilistic social inferences from linguistic inputs. *arXiv preprint arXiv:2306.14325*.

Hanlin Zhang, Jiani Huang, Ziyang Li, Mayur Naik, and Eric Xing. 2023. Improved logical reasoning of language models via differentiable symbolic programming. *arXiv preprint arXiv:2305.03742*.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, et al. 2022a. Least-to-most prompting enables complex reasoning in large language models. *arXiv preprint arXiv:2205.10625*.

Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. 2022b. Large language models are human-level prompt engineers. *arXiv preprint arXiv:2211.01910*.

A Optimzed Prompts for different tasks

In this section, we demonstrate optimized prompts by Chain-of-Thought (CoT), Automatic Prompt Engineering (APE), PromptAgent, and our method with F1 scores respectively.

Table 4: Comparison of Optimized Prompts for **Object Counting task**, including CoT, APE, PromptAgent, and our method

Approach	Optimized Prompt	F1 score
CoT	Your task is to count the total number of objects mentioned in the question. Follow these simple steps to ensure accurate counting: **Steps to Follow:** 1. **Identify Items**: Read the question carefully and list all objects mentioned. 2. **Count Quantities**: For each item, check if a quantity is provided. If no quantity is mentioned, assume it is one. 3. **Add Totals**: Add up the quantities of all items to calculate the total count. 4. **Verify the Total**: Double-check to ensure no item is missed or counted twice. **Example:** - Question: "Count the apples, oranges, and bananas. There are 2 apples, 1 orange, and 3 bananas." - Step 1: Identify items: apples, oranges, bananas. - Step 2: Count quantities: 2 apples, 1 orange, 3 bananas. - Step 3: Add totals: 2 + 1 + 3 = 6. - Step 4: Verify: All items are accounted for, total is 6. - **Output**: "The total count is 6." Use this step-by-step method for every question to ensure accurate and clear results.	0.928
APE	Calculate the overall total of all items even those spoken in groups.	0.863
PromptAgent	Carefully examine the provided information. Identify and catalog each mentioned item, ensuring that explicitly stated quantities are accurately recorded. If no quantity is specified for an item, assume it as a single unit. However, for items with defined quantities, count each unit separately and include it in the total. If collective terms or categories are mentioned, break them down into their individual components and associate each with its stated count. When computing the total for such categories, ensure that the sum reflects all individual units rather than just the number of groups or types. Each item should be counted independently, but related items belonging to a common category should be grouped together, with their specific quantities contributing precisely to the final total. Avoid assumptions regarding the classification or nature of items—adhere to standard, widely accepted definitions. Finally, summarize the count by explicitly listing the quantity of each identified item or category, and provide a comprehensive total of individual units rather than just category counts, unless otherwise specified.	0.882

(Continued on next page)

(Continued from previous page)

Approach	Optimized Prompt	F1 score
Our	Task Requirements: The task involves counting the total number of objects listed in a question. Each distinct object should be considered as part of the total count, regardless of its type or variation. The output should be formatted correctly as specified. Problem Rule Application: Identify all items listed in the question. Count each item exactly once, regardless of type, to determine the total number of objects. Ensure accuracy by verifying that all listed items have been included in the final count. Provide the final result in the required format: The number should be presented in both word form and numerical form, separated by a comma (e.g., "nine, 9"). No extra symbols, characters, or explanations should be included. Judgment Criteria: (Strictly follow these rules) Complete Identification: Extract and recognize all objects in the given list. Do not overlook any item mentioned in the question. Accurate Counting: Each item must be counted exactly once. Ensure no items are omitted or double-counted. Verification Process: Double-check the list to confirm that all objects are included. Cross-verify the final count to avoid errors.	0.923

Table 5: Comparison of Optimized Prompts for **Penguins In A Table task**, including CoT, APE, PromptAgent, and our method

Approach	Optimized Prompt	F1 score
CoT	You are tasked with answering questions about a table of penguins and their attributes. Use step-by-step reasoning to ensure accuracy in calculations and comparisons. The table is as follows: “‘ Name, Age, Height (cm), Weight (kg) Louis, 7, 50, 11 Bernard, 5, 80, 13 Vincent, 9, 60, 11 Gwen, 8, 70, 15 “‘ **Reasoning Steps for Each Question:** 1. Identify the target attribute (age, height, or weight) and the type of operation (comparison, ranking, filtering). 2. Extract the relevant rows or columns based on the question’s requirements. 3. Perform the required operation step-by-step using the extracted data. 4. Clearly summarize the answer based on the operation’s result. Example Workflow: - Question: "Who is the tallest penguin?" - Step 1: Identify the target attribute: Height. - Step 2: Extract the height values and corresponding names: [(Louis, 50), (Bernard, 80), (Vincent, 60), (Gwen, 70)]. - Step 3: Find the maximum height: Bernard (80 cm). - Step 4: Output the result: "Bernard is the tallest penguin with a height of 80 cm." Follow this workflow for every question to ensure clarity and correctness.	0.818
APE	Carefully scrutinize the provided table or tables. Understand the query in relation to the information given. Pinpoint the pertinent data and carry out the vital computations or comparisons to determine the right answer from the given choices.	0.848

(Continued on next page)

(Continued from previous page)

Approach	Optimized Prompt	F1 score
PromptAgent	Answer questions about a table of penguins and their attributes, considering both the penguin table and any additional relevant tables. Please provide step-by-step reasoning for your answers, and ensure to clarify any criteria used for filtering or sorting data. Here is a table where the first line is a header and each subsequent line is a penguin: name, age, height (cm), weight (kg) Louis, 7, 50, 11 Bernard, 5, 80, 13 Vincent, 9, 60, 11 Gwen, 8, 70, 15 For example: the age of Louis is 7, the weight of Gwen is 15 kg, the height of Bernard is 80 cm. What is the name of the last penguin sorted by alphabetic order? Options: (A) Louis (B) Bernard (C) Vincent (D) Gwen (E) James **Instructions**: 1. List the names of the penguins. 2. Sort the names alphabetically and present the sorted list clearly. 3. Identify the last name in the sorted list and indicate the corresponding option letter from the provided options. 4. If the last name does not match any of the options, select the name that is closest to the last name in the original list of penguins. At the end, show the answer option bracketed between <answer> and </answer>.	0.961
Our	Answer questions about a dynamic, comprehensive table of penguins and their attributes that allows penguins and other animals to be added and removed. Perform calculations and comparisons based on the questions asked. Read the question carefully to determine which attribute is being compared (age, height, weight). When comparing an attribute, extract the name and that attribute, and then compare, ignoring the other attributes. Ensure the extracted value is from the correct column corresponding to the requested attribute. When using the table, align the data so that the first number is age, the second is height, and the third is weight. Understand the question correctly, find the key words from it, and then perform calculations or comparisons based on the key words The current table is as follows: Name, Age, Height (cm), Weight (kg) Louis, 7, 50, 11 Bernard, 5, 80, 13 Vincent, 9, 60, 11 Gwen, 8, 70, 15 **Question Rules to Apply:** - Identify the rows or columns that meet the specified conditions. - Retrieve the value of the required attribute from the identified rows or columns. When we modify this table (by adding new penguins or removing existing penguins or adding giraffes), we first confirm whether the information we added is a penguin or a giraffe, and then solve the problem of comparing, ranking, and filtering based on attributes between penguins or giraffes, depending on the problem.	0.964

Table 6: Comparison of Optimized Prompts for **Geometric Shapes task**, , including CoT, APE, PromptAgent, and our method

Approach	Optimized Prompt	F1 score
CoT	Your task is to identify the geometric shape represented by the given SVG path data. Follow these steps to ensure accuracy: **Steps to Identify the Shape:** 1. **Check for 'A' Instructions**: - If the path contains 'A', determine: - **Circle**: 2 or more 'A' instructions. - **Sector**: 1 'A' instruction. 2. **Count 'L' Instructions**: - If there are no 'A' instructions, count the 'L' instructions to determine the polygon's shape: - **Line**: 2 'L'. - **Triangle**: 3 'L'. - **Rectangle**: 4 'L'. - **Pentagon**: 5 'L'. - **Hexagon**: 6 'L'. - **Heptagon**: 7 'L'. - **Octagon**: 8 'L'. - **Kite**: 4 'L'. 3. **Provide the Shape Name**: Output only the name of the shape (e.g., "circle", "triangle", "hexagon"). **Example:** - Input: ‘"M 10 10 L 20 10 L 20 20 L 10 20 Z"‘ - Step 1: No 'A' instructions. - Step 2: Count 'L' instructions: 4 'L'. - Step 3: Shape is a **Rectangle**. - **Output**: "rectangle". Use this step-by-step process for all inputs to determine the correct shape.	0.791
APE	Determine the shape each SVG path element is drawing, then pair it with the corresponding letter from the available choices. In this case, C symbolizes hexagon, G is for pentagon, I signifies sector, and B stands for heptagon.	0.650

(Continued on next page)

(Continued from previous page)

Approach	Optimized Prompt	F1 score
PromptAgent	Analyze the SVG path data to identify the geometric shape it represents. Follow these comprehensive and refined steps to ensure accurate identification: 1. **Holistic Path Closure**: Determine if the path forms a closed shape by checking if the last point connects back to the starting point. If multiple ‘M‘ commands are present, analyze the segments collectively to identify any closed loops. Treat the entire path as a single entity for a thorough analysis. 2. **Segment and Side Analysis**: Identify the types of segments used in the path: - **Line Segments**: Count the number of distinct line segments to determine the number of sides. Ensure accurate counting by considering all segments collectively. - **Arc Segments**: For paths using the ‘A‘ command, note that these represent elliptical arcs. Pay attention to the parameters to distinguish between circles and ellipses. 3. **In-depth Geometric Properties**: - For line segments, analyze the relative lengths of sides and angles between them. Consider properties such as parallel sides, equal side lengths, and right angles to distinguish between different types of polygons. Evaluate the overall shape formed by all segments. - For arc segments, examine the parameters of the ‘A‘ command: - Check if the radii are equal, which indicates a circle. - If the radii differ, consider the shape as an ellipse. 4. **Shape Identification and Classification**: Use the geometric properties to classify the shape: - For polygons, identify specific types like rectangles, kites, and trapezoids based on their properties. Pay special attention to the number of sides and the relationships between them. Consider the entire path as a single shape to ensure accurate classification. - For arcs, determine if the shape is a circle or an ellipse based on the radii. 5. **Options Selection and Interpretation**: Choose the most appropriate shape from the given options. Consider multiple interpretations of the path data, especially when multiple ‘M‘ commands are present, to ensure accurate classification. If the path represents multiple shapes, prioritize the most complex or relevant shape. 6. **Ambiguity Resolution**: In cases where the path data could represent multiple shapes, provide a rationale for selecting the most complex or relevant shape. Consider the context and any additional information that might influence the classification. 7. **Visual Verification**: If possible, visualize the path to confirm the identified shape. This step can help resolve any remaining ambiguities and ensure the accuracy of the classification. 8. **Iterative Refinement**: If the initial classification is uncertain, revisit the analysis steps to refine the identification. Consider alternative interpretations and re-evaluate the geometric properties. 9. **Contextual Considerations**: Take into account any contextual information or additional data that might influence the shape classification, especially in ambiguous cases. Provide your answer by selecting the correct option and enclosing it within `<answer>` and `<answer>` tags. Example: - SVG Path: ‘path d="M 8.10,55.86 L 1.74,25.57 L 12.08,23.40 L 18.44,53.69 L 8.10,55.86"‘ - Analysis: The path forms a closed quadrilateral with opposite sides parallel and equal, indicating a rectangle. - Answer: `<answer>H</answer>` - SVG Path: ‘path d="M 16.33,5.98 A 8.87,8.87 275.02 1,0 14.78,23.64 A 8.87,8.87 275.02 1,0 16.33,5.98"/‘ - Analysis: The path uses elliptical arcs with equal radii, forming a closed loop, indicating a circle. - Answer:	0.830

(Continued from previous page)

Approach	Optimized Prompt	F1 score
our	Given the following SVG path data: "input" and options, identify the geometric shape it represents and provide ONLY the name of the shape as the 'target'. **Task Requirements:** 1. Count the instructions in the SVG path 2. Judge the shape of the graphic according to the judgment criteria 3. Provide the exact name of the shape as output. You need to count how many instructions L are in the SVG path: **Problem Rule Application:** 1. Visualize the path data to understand the overall structure. 2. Find out whether there is instruction A in the instruction. If so, determine whether it is a circle or a sector according to the number of instructions A. If not, determine how many sides it is 3. For polygons, pay attention to the number of edges to identify the shape. The following are the number of instructions corresponding to different shapes: # - **triangle**: 3 L # - **rectangle**: 4 L # - **hexagon**: 6 L # - **pentagon**: 5 L # - **octagon**: 8 L # - **heptagon**: 7 L # - **kite**: 4 L # - **line**: 2 L # - **circle**: 2 or more A # - **sector**: 1 A **Judgment criteria:**(Please strictly abide by this rule) No need to pay attention to "M" instructions !! First identify whether there is an instruction "A" in the SVG path. If so, first determine whether it is a circle or a sector. !! If there is no instruction "A", determine the number of sides of the polygon based on the instruction "L". A polygon with n sides requires n "L" instructions.(Please strictly abide by this rule)	0.822

Table 7: Comparison of Optimized Prompts for **Causal Judgment tasks**, including CoT, APE, PromptAgent, and our method

Approach	Optimized Prompt	F1 score
CoT	Task: Respond to inquiries about causal attribution by identifying the key causes and their contributions to the outcome. Follow the steps below to ensure accurate and clear reasoning: **Steps to Analyze Causation:** 1. **Identify Key Entities**: Read the question carefully and highlight the specific entities or factors being discussed. 2. **Determine Relevant Causes**: Analyze the context to identify immediate and incidental causes contributing to the outcome. - Immediate causes: Directly lead to the outcome. - Incidental causes: Indirectly influence the outcome but may still contribute. 3. **Evaluate Interactions**: Consider how multiple causes might interact to produce the observed effect (e.g., synergy or independent contributions). 4. **Provide the Answer**: Clearly state the primary and secondary causes, as well as their roles in creating the outcome. Avoid unsupported assumptions. Use this structured reasoning approach to analyze each inquiry and provide a clear and logical explanation.	0.678

(Continued on next page)

(Continued from previous page)

Approach	Optimized Prompt	F1 score
APE	For each situation, decide if the result was caused deliberately or not. If the individual or party behind the event was aware of the potential result and chose to go ahead, select 'Yes'. If they didn't intend the result to happen, even if they knew it could possibly occur, select 'No'.	0.756
PromptAgent	When addressing questions about causal attribution, ensure a comprehensive analysis by considering both individual and collective actions that contribute to an outcome. Clearly differentiate between necessary and sufficient causes, and recognize that multiple causes can simultaneously contribute to an outcome. Emphasize the importance of understanding both general and specific intentions, especially when outcomes are unintended. Define "intentional" actions as those where the actor or group had control over maintaining or altering the conditions necessary for the outcome, even if the specific result was not desired. Address scenarios where unintended consequences arise from intentional actions, and provide answers that reflect a nuanced understanding of how different elements interact to produce a result. Use diverse examples to illustrate key concepts like "direct causation," "simultaneity," and "unintended consequences," ensuring a balanced consideration of necessary and sufficient causes. Simplify complex scenarios by breaking them down into clear, manageable components, and provide definitions or examples of key terms to guide your analysis. Additionally, clarify definitions of key terms such as "necessary," "sufficient," "intentional," and "unintended consequences" to ensure precise understanding. Highlight the importance of interactions between multiple causes, especially in complex scenarios, and offer strategies for analyzing scenarios where simultaneity is crucial. Explore the nuances of intentional actions and unintended consequences more deeply, and encourage the use of diverse examples to illustrate different aspects of causation. Pay special attention to the role of individual actions in maintaining necessary conditions and the distinction between collective and individual causation. Emphasize that in collective decision-making, the outcome can be intentional if it aligns with the group's goals, even if individual members disagree. Reinforce the distinction between necessary and sufficient causes, ensuring the model understands that necessary causes alone do not determine the outcome. Clarify that following a protocol does not remove intentionality if the outcome aligns with organizational priorities. Highlight that intentionality can be attributed if the outcome was a foreseeable consequence of the action, regardless of individual opposition.	0.846

(Continued on next page)

(Continued from previous page)

Approach	Optimized Prompt	F1 score
Our	Task Requirements Determine whether a given event (cause) directly leads to another event (effect). Assess the causal relationship based on logical reasoning, ensuring a clear and definitive answer. The final output must be only "Yes" or "No", strictly adhering to the required format. Problem Rule Application Identify the cause and effect within the question. Assess necessity: Determine if the cause is essential for the effect to occur. Evaluate causation: If the cause did not happen, would the effect still occur? If the effect only happens when the cause is present, then the cause directly leads to the effect. If the effect can still happen independently, then the relationship is not causal. Judgment Criteria Direct Causation: If the cause directly leads to the effect and is a necessary condition, answer "Yes". If the effect would not have occurred without the cause, answer "Yes". Example: "Dropping a glass caused it to shatter." $\rightarrow$ Yes. No Direct Causation: If the effect can occur without the cause, answer "No". If the cause is only correlated but not necessary, answer "No". Example: "Wearing a red shirt caused the stock market to rise." $\rightarrow$ No. Verification Process: Check whether the absence of the cause results in the absence of the effect. Ensure logical consistency in the causal assessment.	0.880

Table 8: Comparison of Optimized Prompts for **Epistemic task**, including CoT, APE, PromptAgent, and our method

Approach	Optimized Prompt	F1 score
CoT	Task: Analyze the logical relationship between a given premise and hypothesis. Your goal is to determine if the premise guarantees the truth of the hypothesis. Choose one of the following answers: 'entailment' or 'non-entailment'. **Steps to Follow:** 1. **Understand the Premise and Hypothesis**: Carefully read the premise and hypothesis to identify the key information in both statements. 2. **Analyze the Logical Relationship**: Determine whether the information in the premise confirms the truth of the hypothesis. - If the premise logically supports and guarantees the hypothesis, choose 'entailment'. - If the premise does not confirm the hypothesis, or if there is uncertainty, choose 'non-entailment'. 3. **Provide the Answer**: Based on your analysis, output the correct answer ('entailment' or 'non-entailment'). Use this step-by-step approach for all premise and hypothesis pairs to ensure accurate reasoning.	0.855
APE	Determine whether the hypothesis is directly implied by the premise or not. If the premise's statement is a direct claim or conviction of the individual mentioned in the hypothesis, choose 'entailment'. However, if the premise is formed on the belief or supposition of someone other than the subject in the hypothesis, opt for 'non-entailment'.	0.888

(Continued on next page)

(Continued from previous page)

Approach	Optimized Prompt	F1 score
PromptAgent	Determine the relationship between two sentences by evaluating whether the first sentence provides direct or logically implied evidence for the second. Choose from the options 'entailment' or 'non-entailment'. Consider the following: - **Entailment**: The first sentence directly or through logical implication confirms the truth of the second sentence, even if it involves a chain of beliefs or perceptions, as long as the chain logically supports the hypothesis. - **Non-entailment**: The first sentence does not confirm the truth of the second sentence, often involving unsupported assumptions, beliefs, or suspicions that do not logically lead to the hypothesis. Guidelines for Analysis: 1. **Clarify Belief Chains and Logical Implications**: Understand how belief chains work and when they logically support the hypothesis. Pay attention to verbs indicating beliefs, assumptions, or suspicions (e.g., "thinks," "assumes," "suspects") versus those indicating direct evidence (e.g., "learns," "knows," "remembers"). Consider how these verbs interact in belief chains and what they imply about the subject's own beliefs. 2. **Evaluate Direct and Implied Evidence**: Determine if the premise provides direct or logically implied evidence for the hypothesis, considering how belief chains can logically support the hypothesis. Recognize that indirect beliefs about another person's recognition can imply one's own belief about a situation, especially when the belief chain is logical and straightforward. 3. **Consider Perspective and Source of Information**: Note any differences in perspective or source of information (e.g., who remembers or assumes something) and how these perspectives contribute to the logical implication of the hypothesis. 4. **Conduct a Comprehensive Analysis**: Use a step-by-step approach to ensure all relevant details and logical implications are considered in the analysis. Balance the emphasis on direct evidence with the recognition of logical implications from indirect beliefs. Example: Premise: "Charlotte thinks that Richard recognizes that a boy is standing in a pool getting splashed with water." Hypothesis: "Charlotte thinks that a boy is standing in a pool getting splashed with water." Options: (A) entailment (B) non-entailment Analysis: 1. **Understanding the Premise**: The premise indicates that Charlotte thinks Richard recognizes a specific situation involving a boy in a pool. 2. **Understanding the Hypothesis**: The hypothesis states that Charlotte thinks a boy is in a pool getting splashed with water. 3. **Assessing the Relationship**: The premise implies that Charlotte has a belief about the situation (through Richard's recognition), which logically supports the hypothesis. Charlotte's belief about Richard's recognition suggests she also believes in the situation's occurrence. 4. **Conclusion**: The relationship is one of entailment because Charlotte's belief about Richard's recognition logically implies her belief in the situation. Therefore, the correct answer is: `<answer>A</answer>` Identify the relation between the following premises and hypotheses, choosing from the options 'entailment' or 'non-entailment'. At the end, show the answer option bracketed between `<answer>` and `</answer>`.	0.916

(Continued on next page)

(Continued from previous page)

Approach	Optimized Prompt	F1 score
our	Task Requirements: Analyze a given premise (primary sentence) and determine whether it fully supports the truth of a hypothesis (subsequent sentence). Classify the relationship as either "Entailment" or "Non-Entailment" based on the logical and factual connections between the two. Provide the classification only as the final output. Problem Rule Application: Entailment: The premise explicitly confirms the hypothesis with clear, direct evidence. No additional information, assumptions, or interpretations are required to validate the hypothesis. Non-Entailment: The premise does not fully or explicitly confirm the hypothesis. If there is ambiguity, uncertainty, or missing logical links, label it as Non-Entailment. Judgment Criteria: (Strictly follow these rules) Language of Uncertainty: Words like "assumes," "believes," "thinks," "feels," "suspects" indicate subjectivity and should not be considered definitive proof. These terms suggest a possibility rather than an explicit factual connection. Specific vs. General Statements: A specific premise (e.g., mentioning a “full face mask”) does not necessarily contradict a general hypothesis (e.g., referencing a “mask” in general). However, if the premise is too general to confirm the specific claim, classify as Non-Entailment. Objective Reasoning: Only use the logical and factual ties within the given statements. Do not rely on external knowledge, assumptions, or interpretations unless directly supported by the premise. Decision Process: Determine whether the premise fully supports the hypothesis without needing extra inference → Entailment. If the premise only partially supports or fails to confirm the hypothesis → Non-Entailment.	0.893

Table 9: Comparison of Optimized Prompts for **Temporal task** including CoT, APE, PromptAgent, and our method.

Approach	Optimized Prompt	F1 score
CoT	Your task is to determine the available time slot for an event, based on a schedule of occupied times. Follow these steps to ensure accuracy: **Steps to Identify Free Time Slots:** 1. **List Occupied Periods**: Organize all occupied time slots in chronological order. 2. **Find Gaps**: Identify gaps between the occupied periods where no activities are scheduled. 3. **Check Constraints**: Ensure that the free time slots fall within operational constraints (e.g., facility closing times). 4. **Select the Slot**: Choose the correct free time slot that satisfies all criteria. **Output Result Format:** - Present the selected free time slot in a clear format, such as "Xpm to Ypm" or "Xam to Yam". Use this step-by-step method to ensure that the identified time slot is accurate and does not overlap with any occupied periods.	0.989

(Continued on next page)

(Continued from previous page)

Approach	Optimized Prompt	F1 score
APE	Identify the period when the individual was unnoticed and had the possibility to visit the specified place before its closing time.	0.994
PromptAgent	Analyze the timeline of events to determine possible time frames during which certain events could have occurred, even if they were not explicitly observed. Start by constructing a comprehensive timeline, clearly listing all observed and unobserved time slots. Identify gaps where the subject is unobserved, ensuring these gaps fit within any given constraints, such as opening and closing times. Emphasize the importance of constraints by verifying them after identifying potential gaps. Use a step-by-step reasoning approach to systematically evaluate all available information, and include a final review to check for potential errors or overlooked details before finalizing the answer. Define key terms like "unobserved" and "constraints" to ensure clarity in the task requirements. Provide examples to illustrate the reasoning process and expected output format, guiding the model in analyzing timelines and identifying possible time frames for unobserved events. Additionally, incorporate a checklist to ensure all steps are followed, and highlight common pitfalls to avoid during the analysis. Finally, include a summary of the reasoning process to reinforce understanding and ensure the model's conclusions are well-supported. To further enhance the model's performance, include additional examples that cover a wider range of scenarios and constraints, such as overlapping time slots or multiple constraints. Provide explicit guidance on handling complex constraints and ambiguous information. Incorporate interactive feedback mechanisms to help the model learn from mistakes and improve over time. Ensure the prompt is concise and focused, avoiding unnecessary repetition while maintaining clarity and comprehensiveness. Additionally, introduce a section for handling exceptions or unusual cases, offering strategies for dealing with incomplete or conflicting data. This will help the model adapt to a broader range of real-world scenarios and improve its robustness in timeline analysis tasks.	0.984

(Continued on next page)

(Continued from previous page)

Approach	Optimized Prompt	F1 score
our	**Task Requirements:** Determine the possible time period during which an event could have occurred, based on a detailed schedule of occupied times. Your goal is to identify the correct time slot that fits all the provided criteria without any overlap. **Problem Rule Explanation:** 1. Analyze the schedule to identify all time slots during which the person is occupied. 2. Determine the available time slots by identifying gaps between these occupied periods. 3. Consider any additional constraints, such as closing times, that may limit the available time slots. **Problem Rule Application:** - List all the occupied time slots chronologically. - Identify gaps between these occupied slots where the person is free. - Ensure that the free time slots do not conflict with constraints like closing times. **Result Verification:** - Confirm that the identified time slot is completely free and adheres to any constraints. - Double-check against all occupied periods to ensure there is no overlap. - Avoid selecting time slots that are partially occupied or overlap with any scheduled activities. **Output Result Format:** - Present the correct time slot in a straightforward manner, using the format "Xpm to Ypm" or "Xam to Yam" as appropriate. - Ensure the output is clear and free of any extraneous symbols or text. **Common Mistakes to Avoid:** - Do not include time slots that extend beyond the closing time of the facility. - Avoid selecting time slots that overlap with any scheduled activities. - Ensure the selected time slot is entirely free and does not partially overlap with any occupied period. **General Rules and Analysis:** - Identify all occupied periods and list them chronologically. - Look for gaps between these periods where the person is not scheduled for any activity. - Verify that these gaps fall within any operational constraints, such as closing times. - Ensure the selected time slot is entirely free and does not overlap with any occupied periods. By following these guidelines, you can accurately determine the available time slot for the event in question. Avoid errors by ensuring that the selected time slot is entirely free and does not overlap with any occupied periods.	0.993

B Prompt Self-improvement

B.1 Error Analysis

1. **Misinterpretation of the Question**: The model may have misinterpreted the question, focusing on the structure of the data rather than the specific request for the height of the tallest penguin. This could lead to confusion and an irrelevant answer.

2. **Inattention to Numerical Data**: The model might have overlooked the numerical values provided in the table, failing to recognize that it needed to compare the heights of the penguins to determine the tallest one.

3. **Irrelevant Output Generation**: The answer "rectangle" does not relate to the context of the question. This suggests that the model may have generated a response based on unrelated patterns or associations rather than the specific data presented.

4. **Lack of Contextual Understanding**: The model may not have fully grasped the context of the data table, leading to a failure in recognizing that the question was asking for a specific value derived from the table.

5. **Failure to Process Tabular Data**: The model might struggle with processing tabular data effectively, which can lead to incorrect conclusions or irrelevant outputs when asked to analyze such formats.

B.2 Improvement Recommendations

- Clarify the question to emphasize the need for numerical comparison.
- Provide explicit instructions to focus on extracting specific values from the data.
- Ensure the model is trained to recognize and process tabular data more effectively.
- Avoid ambiguity in the question to prevent misinterpretation of the request.

B.3 Improvement Recommendations: Revised Prompt

Prompt:
Here is a table of penguins with their respective attributes:

```
name, age, height (cm), weight (kg)
Louis, 7, 50, 11
Bernard, 5, 80, 13
Vincent, 9, 60, 11
Gwen, 8, 70, 15
```

Task Requirements:
Analyze the table to identify the tallest penguin based on the 'Height (cm)' column.

Problem Rule Explanation:
Compare the numerical values in the 'Height (cm)' column to determine the maximum height.

Problem Rule Application:
Identify the maximum value from the 'Height (cm)' column.

Result Verification:
Ensure that the identified height corresponds to the tallest penguin.