ADAPT: Adaptive Data-driven Automatic Prompt Tuning

Anonymous ACL submission

Abstract

The effectiveness of Large Language Model (LLM) applications depends heavily on prompt engineering, which suffers from high labeling costs and low robustness. Existing prompt optimization methods typically rely on generic optimization prompts to extract insights from task interactions. However, lacking sufficient understanding of the target environment, these prompts often produce overly generic insights that fail to capture the dominant reasoning patterns and failure modes of the task. To address this, we propose a training-free prompt optimization framework, Adaptive Data-driven Automatic Prompt Tuning (ADAPT). Given a few task examples, ADAPT first leverages an LLM's meta-cognitive ability to propose a task-specific knowledge schema automatically. It then uses this schema to guide a cognitively inspired prompt tuning stage, enabling more effective knowledge and insight extraction without human intervention. Experiments across diverse tasks show that ADAPT significantly outperforms human-curated Automatic Prompt Tuning (APT) prompts. Source code will be released upon publication.

1 Introduction

Large Language Models (LLMs) have shown remarkable general capabilities across a wide range of tasks, including knowledge-intensive tasks (GSM8K (Cobbe et al., 2021) AIME (Zhang et al., 2023a), HotpotQA (Yang et al., 2018), MATH-500 (Lightman et al., 2024)), embodied tasks (Robotouille (Gonzalez-Pumariega et al., 2025), FEVER (Thorne et al., 2018), ALFWorld (Shridhar et al., 2021)), and code generation and debugging (CodeAgent (Zhang et al., 2024b), MetaGPT (Hong et al., 2024), SWE-bench (OpenAI, 2024)).

LLMs rely on prompt designs that engineer the reasoning performance. Simple methods include Chain-of-Thought (Wei et al., 2022), Least-to-Most (Zhou et al., 2023), Self-Consistency (Wang et al., 2023), and Tree-of-Thoughts (Yao et al., 2023). Despite their impressive capabilities, LLM agents often underperform in specialized, real-world domains. These scenarios typically demand human expertise to design prompts that involve domain-specific knowledge, problem-solving strategies, and the use of external tools. Human-crafted prompts not only incur high labeling expenses and inconsistent quality but also lack robustness in evolving environments where continual adjustments are required to remain effective or adaptability across different LLMs of different families/versions, tasks, and domains.

Automatic Prompt Tuning (APT) has emerged to mitigate manual intervention by employing algorithms to autonomously search, generate, or optimize prompts. Prior work, such as training-free GRPO (Lab, 2025), GEPA (Agrawal et al., 2025), and ACE (Zhang et al., 2025), has shown similar results to parameter optimization on LLMs' parameters by attaching knowledge induced from experiences on given tasks or environments. This type of training-free prompt optimization benefits from lower computation overhead and is free from the current technical challenges associated with continuous learning, such as catastrophic forgetting and overfitting.

However, these methods exhibit limitations when deployed in multi-domain environments: they typically rely on generic insight-extraction prompt templates, and therefore fail to effectively extract and accumulate task-specific optimization knowledge. In contrast, humans typically first develop an understanding of domain-specific knowledge before providing guidance for specialized tasks, enabling more targeted and effective recommendations than generic problem-solving advice.

In this paper, we propose Adaptive Data-driven Automatic Prompt Tuning (ADAPT), where we adopt an experience learning pipeline that mimics the above human learning process. Specifi-

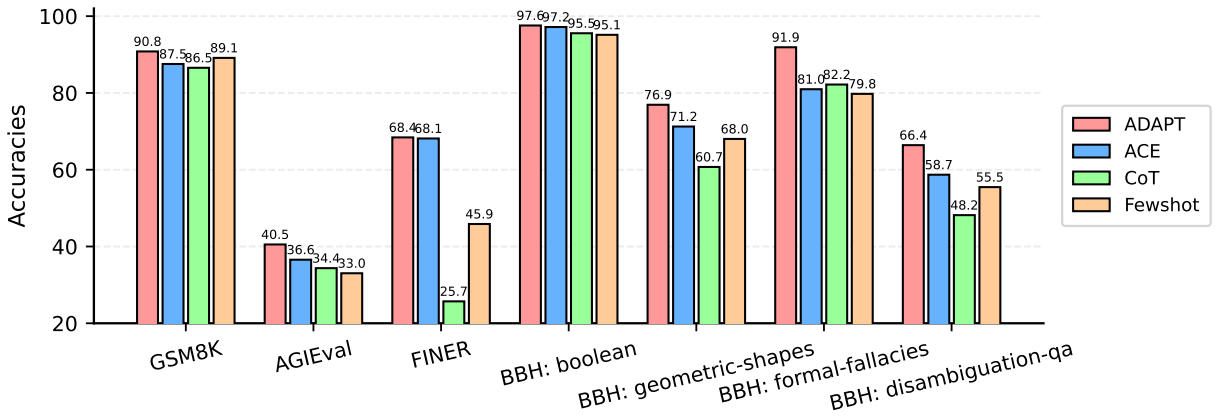

Figure 1: Performance Comparison of our ADAPT and baseline methods (ACE, CoT, and Few-shot) on various benchmarks encompassing knowledge-intensive tasks

cally, ADAPT is a two-stage structured cognitive workflow. In its first stage, guided by a task's environment specifications, instructions, and some training examples, ADAPT uses a meta-prompt to explicitly extract a knowledge schema for the task. This knowledge schema contains adaptive categories of knowledge items, including relevant rules, methodologies, and concepts. ADAPT then builds optimization prompts containing the knowledge schema and interactively updates and expands the rules and insights within the knowledge schema from those extracted from the task environment. Here, the knowledge schema acts as a blueprint, the final outcome of ADAPT is a task-specific optimized prompt that integrates more relevant, effective, and validated knowledge items.

Our main contributions are summarized as follows: (1) We extend the scope of Automatic Prompt Tuning (APT) to optimize the APT prompt templates themselves, enabling more effective extraction and accumulation of task-specific optimization knowledge and consistently improving existing APT frameworks. (2) We introduce explicit knowledge schema into prompt optimization, leading to prompts that embody more robust, generalizable, and human-interpretable problem-solving strategies. Specifically, we design a two-stage approach that builds knowledge schema before data-driven prompt optimization to enhance generalization. (3) We conduct extensive experiments across heterogeneous benchmarks and demonstrate that our approach significantly outperforms state-of-the-art baselines, showing that it is a promising approach for building systems that adapt to novel domains and environments without human intervention.

2 Adaptive Data-Driven Automatic Prompt Tuning

2.1 System Architecture Overview

We design a two-stage framework to optimize Automatic Prompt Tuning (APT) prompt templates for task-specific environments. Given a task with an environment specification, in the first stage, ADAPT performs an *environment examination* process, which generates a knowledge schema S for the task. In the second stage, ADAPT synthesizes task-specific optimization prompts based on S, which are then iteratively updated through interactions with the task environment. The overview of ADAPT's pipeline is shown in Figure 2.

2.1.1 Stage 1: Environment Examination

The first stage is responsible for analyzing the unstructured raw input E (environmental specification, instructions, and examples) into a structured knowledge schema S using our examination prompt P_{exam}:

$$S = \{s_1, s_2, \ldots, s_k\} = P_{exam}(E)$$

where S encompasses environment features across multiple categories, which will be detailed in Section 2.2. This stage mirrors human metacognitive processes in problem investigation by forcing an explicit model of task-specific knowledge schema before finding and connecting the task-relevant knowledge.

2.1.2 Stage 2: Synthesis of Optimization Prompts

In the second stage, we use S to synthesize task-specific optimization prompts P. In this paper,

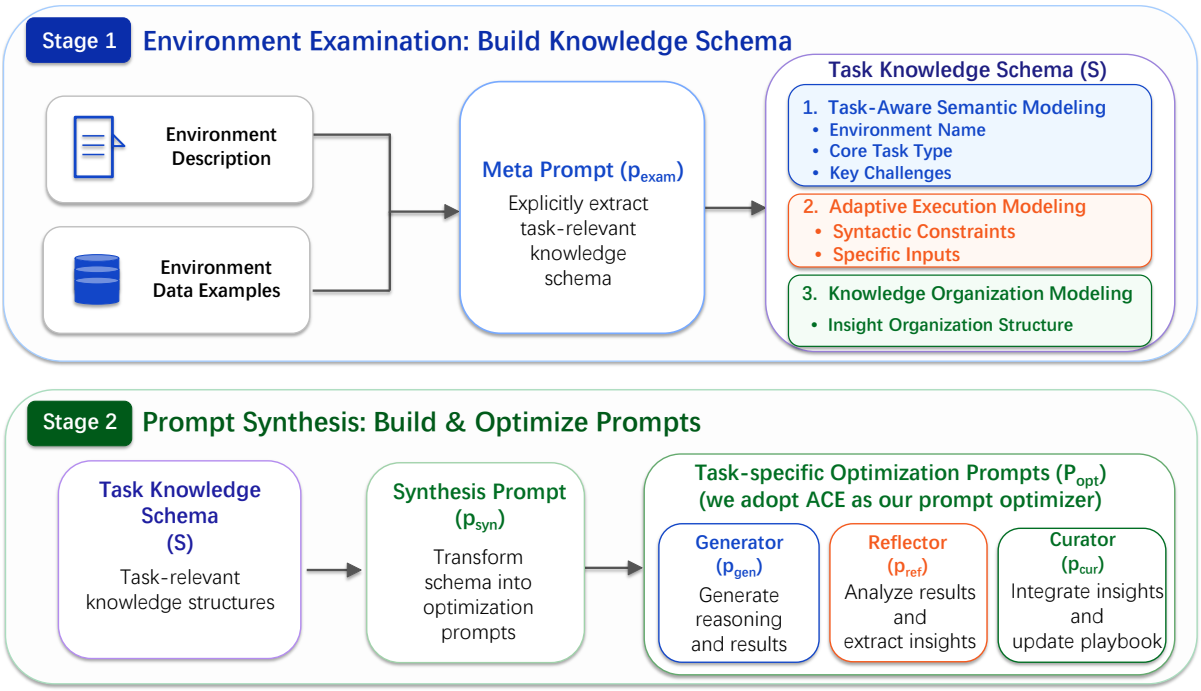

Figure 2: Overview of ADAPT's two-stage pipeline: structured knowledge extraction and task-specific prompt optimization

we develop ADAPT based on Agentic Context Engineering (ACE) (Zhang et al., 2025), although our approach can be applied to other APT approaches. To generate task-specific ACE optimization prompts, we design a synthesis prompt P_{syn}:

$$P_{opt} = P_{syn}(S),$$

where $P_{opt} = \{P_{gen}, P_{ref}, P_{cur}\}$ are the optimization prompts that used to be manually curated in ACE. We adopt ACE as our prompt optimizer. The optimization prompts are then used to generate task-prompts for the target task. For ACE, these prompts are designed with clear separation of responsibilities: The Generator prompt P_{gen} produces reasoning/execution trajectories in response to new queries, thereby exposing effective strategies and failure patterns. The Reflector prompt P_{ref} analyzes these trajectories to evaluate successes and failures, extracting concrete, actionable empirical insights. The Curator prompt P_{cur} translates these insights into structured context updates (delta entries) and integrates them into the existing insights using lightweight, deterministic, non-LLM-based merging logic, thereby enabling scalable context adaptation.

2.2 Knowledge Schema

A knowledge schema is a structured representation or category system that defines what types of information should be extracted from example tasks to guide prompt construction. For instance, in a mathematical reasoning task, a schema could include "operation type," "number relationships," and "common error patterns." The schema acts as a blueprint—it tells the LLM not just that it should learn from examples, but what exactly to look for and how those pieces of knowledge relate (e.g., unit consistency constraints may override otherwise plausible arithmetic operations). A poorly designed schema (e.g., a generic "Heuristic rules" category) misses critical interactions; a well-designed schema (e.g., separating "arithmetic checks," "problem parsing," and "solution strategies") captures the task's underlying structure.

Figure 3 illustrates an example knowledge schema extracted from GSM8K.

2.2.1 Dynamic Knowledge Schema Induction

ADAPT dynamically induces knowledge schemas that transform unstructured environment descriptions and interaction examples into structured, task-relevant representations. Rather than relying on manually predefined task categories, the framework automatically identifies the structures and constraints most relevant for task prompt optimization.

The induced schema captures core semantics of the target environment, including dominant reason-

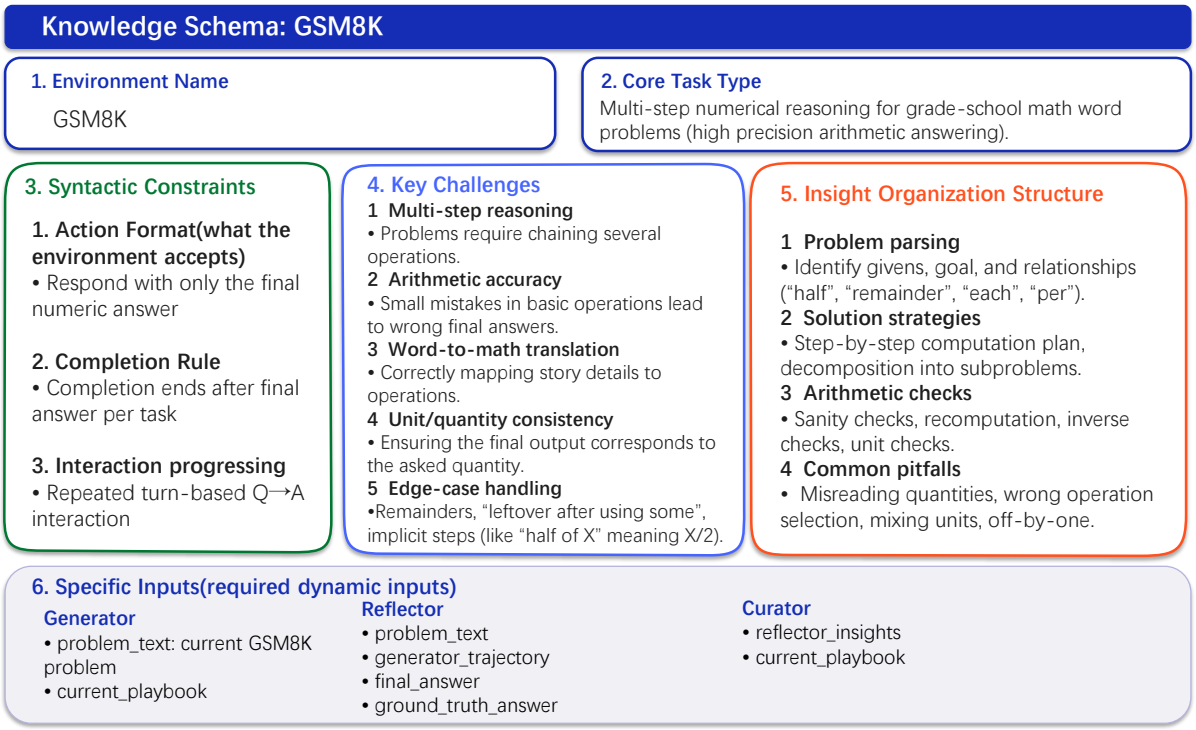

Figure 3: Knowledge schema example of GSM8K.

ing structures, representative reasoning patterns, and major sources of ambiguity or failure. It also summarizes key task challenges, enabling the optimizer to better align refinement with task-specific reasoning requirements.

Beyond semantic understanding, the schema organizes executable environment constraints and runtime information discovered from task specifications and interaction examples. This includes output formatting requirements, interaction protocols, completion conditions, and dynamically changing inputs that cannot be inferred statically. ADAPT further structures these signals according to the characteristics of the target environment, rather than relying on fixed constraint categories.

Finally, ADAPT induces an organizational structure for accumulating knowledge during iterative refinement. Instead of storing heterogeneous insights within generic categories, the framework dynamically constructs task knowledge structures that reflect emerging reasoning patterns and optimization behaviors. This adaptive organization enables accumulated knowledge to gradually evolve into reusable reasoning principles.

2.2.2 Example: GSM8K Knowledge Schema

Figure 3 presents an example Knowledge Schema instantiated for the GSM8K environment. The schema captures multiple optimization-relevant aspects of the task, including environment semantics, reasoning challenges, execution constraints, runtime inputs, and task-specific optimization structures for multi-step mathematical reasoning.

The **Core Task Type** and **Key Challenges** components provide semantic grounding for optimization by explicitly characterizing the reasoning structure and dominant failure modes of the environment, such as multi-step reasoning, arithmetic accuracy, word-to-math translation, unit consistency. This explicit characterization allows the optimizer to directly focus on task-relevant reasoning behaviors instead of repeatedly rediscovering common reasoning patterns through iterative prompt refinement.

The **Syntactic Constraints** and **Specific Inputs** components formalize the execution interface of the environment, including output requirements, interaction mechanics, and runtime variables. For GSM8K, the syntactic constraints mainly enforce strict answer formatting rules, requiring the model to output only the final numeric answer without explanations or additional text within a repeated turn-based question-answering process. Explicitly modeling these constraints helps the optimizer avoid unnecessary reasoning outputs and reduces opti-

mization effort spent on interface-level behaviors unrelated to core mathematical reasoning quality.

Finally, the **Insight Organization Structure** component defines how optimization knowledge is organized across refinement iterations. Instead of accumulating heterogeneous heuristics within generic categories, ADAPT derives environment-specialized optimization structures tailored to mathematical reasoning tasks, including problem parsing, solution strategies, arithmetic checks, and common pitfalls. This organization enables optimization knowledge to gradually evolve into reusable task-level reasoning principles rather than isolated local heuristics.

3 Related work

3.1 Prompt Engineering

A significant body of early work has explored task-independent prompt designs to engineer the reasoning performance of LLMs. This includes approaches such as Chain-of-Thought (Wei et al., 2022), Least-to-Most (Zhou et al., 2023), Self-Consistency (Wang et al., 2023), and Tree-of-Thoughts (Yao et al., 2023). Although these approaches are impressive in general applications, LLMs often underperform in domain-specific tasks that require specific knowledge, problem-solving strategies, or mastering the use of necessary tools.

3.2 Metacognition-Based Prompt Engineering

Several recent works have been proposed for particular tasks, utilizing the metacognitive abilities in LLMs. Buffer of Thoughts (Yang et al., 2024b) derives high-level guidelines for multi-query tasks from previously completed tasks and stores them in a buffer for future reuse, enabling learning from experience and improving efficiency by distilling level-2 slow thinking into level-1 fast thinking. Skill-based CoT (Didolkar et al., 2024) explores the metacognitive capabilities of LLMs in mathematical problem-solving by labeling questions with corresponding skills, clustering them to reduce redundancy, and retrieving skill-relevant examples for in-context learning during inference. Induction-augmented generation (Zhang et al., 2023b) identifies key concepts in questions and uses inductive prompting templates to extract their closely related concepts and common attributes, facilitating more accurate reasoning processes.

While these approaches utilize metacognitive ability to solve particular tasks in micro problem-solving steps, our approach leverages metacognitive capabilities to construct high-level learning schemas for general problem-solving optimization.

3.3 Training-free Prompt Optimization

Prior work, including training-free GRPO (Lab, 2025), GEPA (Agrawal et al., 2025), and Agentic Context Engineering (Zhang et al., 2025), demonstrates that similar results to those achieved through parameter optimization on LLMs can be obtained by attaching task-specific knowledge to prompts, induced from experiences in given tasks or environments. This type of training-free prompt optimization goes beyond result-oriented metrics and incorporates structured understanding of environmental constraints. It benefits from lower computational overhead and is free from current technical challenges associated with continuous learning, such as catastrophic forgetting and overfitting.

Training-free GRPO (Lab, 2025) leverages the group relative semantic advantage instead of numerical ones like GRPO (Zhihong Shao, 2024) within each group of rollouts. It iteratively distills high-quality experiential knowledge during multi-epoch learning on a small set of ground-truth data, and seamlessly integrates such knowledge as the learned token prior during LLM API calls to guide model behavior.

Genetic-Pareto (GEPA) (Agrawal et al., 2025) proposes a reflection-driven prompt evolution method for compound AI systems composed of multiple LLM prompts. In each iteration, it runs a small number of rollouts, uses execution traces and evaluation-derived textual feedback to produce natural-language reflections, and then applies targeted updates to specific module prompts.

Agentic Context Engineering (ACE) (Zhang et al., 2025) proposes a context adaptation framework for LLM applications that continuously improves system performance without updating model weights, by constructing and maintaining the input context over time.

Due to the sensitivity of prompts, these approaches necessitate domain-specific prompt engineering to maintain operational efficacy in different task domains. This motivates approaches such as ours that aim to fully automate prompt optimization across domains and dynamically changing environments, utilizing the metacognitive ability of LLMs.

3.4 Optimization via Searching and Training

This work belongs to training-free prompt optimization, which prevents the overhead of optimization via searching and training. rStar (Qi et al., 2024) introduces a self-play mutual reasoning approach that significantly improves the reasoning capabilities of small language models without fine-tuning. This method employs a costly Monte Carlo Tree Search (MCTS) with a set of five reasoning-inducing prompts. STaR (Zelikman et al., 2022) demonstrates that iterative training on reasoning histories leading to correct answers enables models to solve increasingly complex problems. Similarly, (Snell et al., 2024) fine-tunes small models to perform more reasoning steps using reinforcement learning with beam search, lookahead search, and best-of-N verifiers. ReST-MCTS (Zhang et al., 2024a) integrates process reward guidance with tree search MCTS to collect higher-quality reasoning traces, while AFlow (Anonymous, 2024) iteratively refines task-specific workflows.

4 Experiments

4.1 Experiment settings

Our experiments were conducted using DeepSeek V3.2 (DeepSeek-AI, 2024) and QWen3-8B (Yang et al., 2024a). We use the more advanced DeepSeek V3.2 for prompt optimization and QWen3-8B for reasoning during evaluation. For QWen3-8B, we use the non-thinking mode with the following decoding configurations: temperature=0.9, top_p=0.7, top_k=20, and max_tokens=6000. Results are reported as the mean accuracy over three independent runs, with error bars indicating the standard deviation across runs.

4.2 Datasets

We evaluate ADAPT on diverse benchmarks covering mathematical reasoning, logical reasoning, financial information extraction, and general reasoning tasks.

FINER (Loukas et al., 2022) is a financial named entity recognition benchmark based on XBRL financial documents, containing 139 fine-grained entity types. We use FINER to evaluate whether ADAPT can automatically capture task constraints and optimization strategies. Following our setup, we sample 700 examples from the test split.

GSM8K (Cobbe et al., 2021) is a benchmark of grade-school math word problems designed to evaluate multi-step numerical reasoning and arithmetic abilities. We evaluate on the standard 1K test set.

AGIEval (LSAT-AR) (Zhong et al., 2024) is a logical reasoning benchmark derived from LSAT Analytical Reasoning problems, requiring structured reasoning under complex constraints such as ordering, grouping, and matching.

BBH (Suzgun et al., 2022) (BIG-Bench Hard) is a collection of challenging reasoning tasks designed to evaluate advanced language model capabilities. We evaluate on four representative BBH subtasks.

4.3 Baseline

In our experiments, the LLM during training-free optimization (using the Generator, Reflector, and Curator prompts) is kept identical (DeepSeek V3.2) between the baseline (Vanilla ACE) and ADAPT and its ablations. QWen3-8B is used for evaluation across all methods. This single-variable control ensures that the performance gains of ADAPT stem solely from its ability to transform raw environment specifications into high-fidelity, executable prompts.

ACE Vanilla (Zhang et al., 2025) We adopt the standardized, environment-agnostic prompts provided in the original Agentic Context Engineering (ACE) framework as our primary baseline. These prompts rely on generalized descriptions of agentic roles (Generation, Reflection, and Curation) without pre-aligned knowledge of the specific environment's API syntax or domain-specific pitfalls. This human-curated baseline represents ACE's static optimization setting, which can only learn environmental constraints through trial-and-error.

Chain-of-Thought (CoT) (Wei et al., 2022). Chain-of-Thought (CoT) prompting is employed as a reasoning baseline to evaluate the model's ability to handle multi-step logical deduction. This method encourages the Large Language Model (LLM) to generate a series of intermediate reasoning steps before reaching the final answer. In our experiments, CoT is implemented using standard "Let's think step by step" triggers or manually designed reasoning templates.

Few-shot. Few-shot prompting serves as a contextual baseline to measure the model's capacity for in-context learning. This approach provides the model with a limited set of high-quality input-output exemplars (3 pairs) within the prompt to define the task's expected format and semantic patterns. Our implementation of few-shot prompt-

Benchmark / Task	ADAPT	ACE	COT	Fewshot
BBH: disambiguation_qa	**66.40%** ±1.03	58.70% ±0.95	48.17% ±1.54	55.47% ±1.21
BBH: formal_fallacies	**91.90%** ±0.42	80.97% ±1.15	82.19% ±1.33	79.76% ±0.97
BBH: boolean	**97.57%** ±0.18	97.17% ±0.32	95.55% ±0.33	95.14% ±0.42
BBH: geometric_shapes	**76.92%** ±0.67	71.25% ±0.94	60.72% ±1.96	68.02% ±1.74
AGIEval: LSAT-AR	**40.53%** ±1.30	36.56% ±1.48	34.36% ±1.11	33.03% ±1.24
GSM8K	**90.81%** ±0.25	87.54% ±0.58	86.55% ±0.73	89.13% ±0.69
FINER	**68.42%** ±0.24	68.14% ±0.12	25.71% ±2.85	45.86% ±1.14

Table 1: Accuracy comparison between ADAPT and baselines across various benchmarks. For all methods, we use QWen3-8B as evaluation LLM. For ADAPT and ACE methods, we use DeepSeek v3.2 as training LLM to optimize their prompts for the target task.

ing utilizes fixed, representative examples selected from the training pool.

ADAPT (Ours). We replace the vanilla ACE prompts with our synthesized prompt P in Section 2.1. It is important to note that no manual tuning is performed per-task in our approach: all prompts are synthesized automatically from raw environment specifications in a zero-shot manner. Once synthesized, these prompts are plugged into the same ACE execution loop (comprising the Generator, Reflector, and Curator).

4.4 Comparison Results

As shown in Table 1, ADAPT consistently improves upon vanilla ACE across diverse reasoning benchmarks, with particularly large gains on BBH:disambiguation_qa (+7.7%) and BBH:formal_fallacies (+10.93%). The improvements on AGIEval: LSAT-AR and GSM8K further demonstrate that ADAPT generalizes effectively across complex reasoning environments requiring multi-step logical inference and constraint-sensitive reasoning. On the FINER financial NER benchmark, ADAPT achieves 68.42% accuracy, slightly surpassing the expert-engineered ACE prompts (68.14%), suggesting that ADAPT can automatically induce effective task-specific optimization strategies even in domain-specialized settings. The consistent improvements across diverse benchmarks suggest that schema-guided prompt optimization enhances both cross-task generalization and reasoning robustness in complex environments.

A notable observation is that vanilla ACE occasionally underperforms standard Chain-of-Thought prompting, such as on BBH:formal_fallacies (80.97% vs. 82.19%). This suggests that generic optimization prompts are often insufficient for complex reasoning tasks and may introduce noisy or ineffective optimization behaviors. In contrast, ADAPT improves optimization quality by specializing optimization prompts toward the target environment, enabling more effective extraction and accumulation of task-specific optimization knowledge.

4.5 Ablation Study

Benchmark / Task	Accuracy
BBH: disambiguation_qa	
ADAPT (full)	**66.40%**
- w/o task examples	50.61%
- w/o task instruction	55.87%
- w/o two-stage pipeline	56.28%
BBH: formal_fallacies	
ADAPT (full)	**91.90%**
- w/o task examples	77.33%
- w/o task instruction	83.40%
- w/o two-stage pipeline	79.35%

Table 2: Ablation study for ADAPT on BBH: disambiguation_qa and BBH: formal_fallacies.

We conduct ablation studies on two BBH datasets: disambiguation_qa and formal_fallacies, and the results are summarized in Table 2. We observe consistent performance degradation across all ablated settings, suggesting that each component of ADAPT contributes to maintaining task-aligned optimization behavior during prompt refinement. Specifically: (1) Removing task examples weakens the task understanding captured by the induced knowledge schema, reducing the quality of

task-specific optimization insights. (2) Collapsing schema induction and prompt optimization into a single stage removes explicit schema structuring, reducing the effectiveness of prompt specialization. (3) Removing task-specific instructions weakens the environment signals used for accurate knowledge schema induction, resulting in poorer task adaptation.

4.6 Cost Analysis

Benchmark / Task	ADAPT	ACE
AGIEval	283,337	159,124
GSM8K	361,595	276,525
BBH: boolean	50,548	43,278
BBH: disambiguation_qa	67,423	60,483
BBH: formal_fallacies	159,006	98,414

Table 3: Completion tokens comparison between ADAPT and ACE.

As shown in Table 3, ADAPT consistently uses more completion tokens than ACE across all evaluation tasks, indicating that it tends to generate longer output sequences during inference and generation, thereby incurring higher computational overhead. This increase is particularly pronounced in complex reasoning scenarios: for example, on AGIEval (283,337 vs. 159,124) and BBH: formal_fallacies (159,006 vs. 98,414), ADAPT appears to produce more extensive intermediate reasoning and verification steps. In contrast, the gap is smaller on relatively simpler tasks such as BBH: boolean (50,548 vs. 43,278), suggesting that the additional cost is task-dependent. Overall, ADAPT achieves stronger performance at the cost of increased generation overhead. This highlights the trade-off between reasoning quality and inference efficiency in practical deployment.

4.7 Discussion

4.7.1 Hallucination Risk and Robustness of Examination.

Although the ADAPT framework significantly reduces information entropy through knowledge schema extraction, it still fundamentally relies on the underlying large language model's (LLM's) comprehension capability. During the environment examination phase, the model remains susceptible to hallucination risks—such as misinterpreting environment documentation or fabricating nonexistent constraints. Currently, we employ highly capable LLMs such as GPT-5.2 to extract meta-parameters and generate corresponding prompts. In the future, we will introduce an environment feedback loop to refine these parameters, enabling the system to extract high-quality schema even when the model's comprehension capability is limited.

4.7.2 Efficiency and Cost.

Although the instruction set synthesized by ADAPT is longer than generic, concise prompts, its highly structured and static nature makes it inherently compatible with the input caching mechanisms of modern large language model APIs. In practical deployment, the ADAPT-generated prompt serves as a fixed context prefix, incurring only a one-time overhead during cold start. Subsequent large-scale task inferences can then trigger cache hits, significantly improving adherence to environmental constraints while reducing marginal token costs.

5 Conclusion

In this paper, we proposed ADAPT, a training-free, fully automated prompt optimization framework that improves robustness and transfer across model architectures and versions, task types, domains, and shifting data conditions. Experiments across diverse benchmarks show that ADAPT consistently outperforms human-curated APT prompts, indicating its effectiveness for building adaptive LLM systems without continual manual prompt maintenance.

Limitations

Our method incurs higher token consumption and computational overhead compared to conventional Automatic Prompt Tuning methods, a common drawback among advanced prompting techniques that rely on more complex test-time optimization.

Another limitation of ADAPT is that the quality of the induced knowledge schema still depends on the reasoning and comprehension capabilities of the underlying LLM. However, as foundation models continue to improve in reasoning, comprehension, and self-verification capabilities, we expect the reliability and robustness of schema induction to improve accordingly.

References

Lakshya A Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J Ryan, Meng Jiang, Christopher Potts, Koushik Sen, Alexandros G. Dimakis, Ion Stoica, Dan Klein, Matei Zaharia, and Omar Khattab. 2025. Gepa: Reflective prompt evolution can outperform reinforcement learning. *Preprint*, arXiv:2507.19457.

Anonymous. 2024. AFlow: Automating agentic workflow generation. In *The Thirteenth International Conference on Learning Representations (ICLR)*.

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

DeepSeek-AI. 2024. Deepseek-v3 technical report. https://arxiv.org/pdf/2412.19437.

Aniket Rajiv Didolkar, Anirudh Goyal, Nan Rosemary Ke, Siyuan Guo, Michal Valko, Timothy P Lillicrap, Danilo Jimenez Rezende, Yoshua Bengio, Michael Curtis Mozer, and Sanjeev Arora. 2024. Metacognitive capabilities of LLMs: An exploration in mathematical problem solving. In *AI for Math Workshop @ ICML 2024*.

Gonzalo Gonzalez-Pumariega, Leong Su Yean, Neha Sunkara, and Sanjiban Choudhury. 2025. Robotouille: An asynchronous planning benchmark for LLM agents. In *The Thirteenth International Conference on Learning Representations*.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. 2024. MetaGPT: Meta programming for a multi-agent collaborative framework. In *The Twelfth International Conference on Learning Representations*.

Tencent Youtu Lab. 2025. Training-free group relative policy optimization. *Preprint*, arXiv:2510.08191.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2024. Let's verify step by step. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Lefteris Loukas, Manos Fergadiotis, Ilias Chalkidis, Eirini Spyropoulou, Prodromos Malakasiotis, Ion Androutsopoulos, and Paliouras George. 2022. Finer: Financial numeric entity recognition for xbrl tagging. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (ACL 2022)*. Association for Computational Linguistics.

OpenAI. 2024. Introducing SWE-bench Verified. https://openai.com/index/introducing-swe-bench-verified/. Accessed: 2025-09-16.

Zhenting Qi, Mingyuan Ma, Jiahang Xu, Li Lyna Zhang, Fan Yang, and Mao Yang. 2024. Mutual reasoning makes smaller llms stronger problem-solvers. In *Arxiv*.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. 2021. ALFWorld: Aligning Text and Embodied Environments for Interactive Learning. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2024. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *Preprint*, arXiv:2408.03314.

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. 2022. Challenging big-bench tasks and whether chain-of-thought can solve them. *arXiv preprint arXiv:2210.09261*.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. FEVER: a large-scale dataset for fact extraction and VERification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 809–819, New Orleans, Louisiana. Association for Computational Linguistics.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed H. Chi, Quoc V Le, and Denny Zhou. 2022. Chain of thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, and 40 others. 2024a. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*.

Ling Yang, Zhaochen Yu, Tianjun Zhang, Shiyi Cao, Minkai Xu, Wentao Zhang, Joseph E Gonzalez, and Bin Cui. 2024b. Buffer of thoughts: Thought-augmented reasoning with large language models. *arXiv preprint arXiv:2406.04271*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems*.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. 2022. Star: Bootstrapping reasoning with reasoning. In *Advances in Neural Information Processing Systems*, volume 35, pages 15476–15488. Curran Associates, Inc.

Dan Zhang, Sining Zhoubian, Yisong Yue, Yuxiao Dong, and Jie Tang. 2024a. Rest-mcts*: Llm self-training via process reward guided tree search. Thirty-eighth Conference on Neural Information Processing Systems (NeurIPS).

Kechi Zhang, Jia Li, Ge Li, Xianjie Shi, and Zhi Jin. 2024b. CodeAgent: Enhancing code generation with tool-integrated agent systems for real-world repo-level coding challenges. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13643–13658, Bangkok, Thailand. Association for Computational Linguistics.

Qizheng Zhang, Changran Hu, Shubhangi Upasani, Boyuan Ma, Fenglu Hong, Vamsidhar Kamanuru, Jay Rainton, Chen Wu, Mengmeng Ji, Hanchen Li, Urmish Thakker, James Zou, and Kunle Olukotun. 2025. Agentic context engineering: Evolving contexts for self-improving language models. *Preprint*, arXiv:2510.04618.

Xingyuan Zhang, Philip Becker-Ehmck, Patrick van der Smagt, and Maximilian Karl. 2023a. Action inference by maximising evidence: Zero-shot imitation from observation with world models. In *Thirty-seventh Conference on Neural Information Processing Systems (NeurIPS)*.

Zhebin Zhang, Xinyu Zhang, Yuanhang Ren, Saijiang Shi, Meng Han, Yongkang Wu, Ruofei Lai, and Zhao Cao. 2023b. IAG: Induction-augmented generation framework for answering reasoning questions. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 1–14, Singapore. Association for Computational Linguistics.

Qihao Zhu Runxin Xu Junxiao Song Mingchuan Zhang Y.K. Li Y. Wu Daya Guo Zhihong Shao, Peiyi Wang. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models.

Wanjun Zhong, Ruixiang Cui, Yiduo Guo, Yaobo Liang, Shuai Lu, Yanlin Wang, Amin Saied, Weizhu Chen, and Nan Duan. 2024. AGIEval: A human-centric benchmark for evaluating foundation models. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 2299–2314, Mexico City, Mexico. Association for Computational Linguistics.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, and Ed Chi. 2023. Least-to-most prompting enables complex reasoning in large language models. In *The Eleventh International Conference on Learning Representations*.

A Prompts

Autonomous Knowledge Extractor:

Your primary task is to utilize the provided environmental block and interaction example block to first extract the necessary structured knowledge.

 # RAW ENVIRONMENT DOCUMENTATION INPUT (Mandatory):

You must process this text first to determine the environment's parameters.

Description of the environment:

{environment_description}

Example of the environment's interaction:

{task_interaction}

APT METHOD DESCRIPTION(Principles and Modules)

{apt_method_description}

 # KNOWLEDGE EXTRACTION

You must internally derive and structure the following six critical parameters from the RAW ENVIRONMENT DOCUMENTATION INPUT for the current extraction task.

1. [Environment name]: (e.g., AppWorld, FINER).

2. [Core Task Type]: The primary function or core domain of the agent's tasks, typically a single, concise phrase describing the environment's main operation (e.g., Multi-step application control, High-precision numerical analysis, Text-based Embodied Reasoning).

3. [Key Challenges]: Identified failure modes, ambiguities, or hard requirements (e.g., unit conversion, date sensitivity).

4. [Insight Organization Structure]: Proposed structural categories for the Agent's knowledge base, which MUST be applicable to the environment. (e.g., STRATEGIES, COMMAND-USAGE, Heuristics, PITFALLS).

5. [Syntactic Constraints]: All generalized syntactic rules and operational constraints extracted from the documentation and interaction example, including:

-- Action Format: The singular output format the environment accepts.

-- Completion Rule: How the Agent formally concludes a task (e.g., won: [True] signal, calling a specific task_complete API).

-- Loop Mechanics: Any special variables driving the environment's progress (e.g., page_index, page_source, T-A-O loop).

6. [Specific Inputs]: A list of inputs for each APT module (Refer to APT METHOD DESCRIPTION). You must autonomously decide which dynamic inputs from the RAW ENVIRONMENT DOCUMENTATION INPUT necessitate a placeholder to be passed into the prompt template. Each input should have a description.

Figure 4: ADAPT prompt for step1

GENERATE TAILORED PROMPTS:

You must generate the complete text for the Prompts template of APT method (refer to APT METHOD DESCRIPTION), ensuring all customizations are made based on the Extracted structured task knowledge. Every prompt template should include system prompt, task description, instruction, input, output schema (only show once under each prompt template). For each of the aspects above, you must provide a complete description. Input use {} as placeholders, {} appear in output should be replaced with {{}}. Except when used as input placeholders, parameter names should not be wrapped. Precede the input with its name and a brief description.

 # Each part description:

System Prompt (Identity & Cognitive Foundation): Define the agent's professional persona, core values, and universal reasoning principles. It should establish a stable "mindset" (e.g., analytical rigor, adherence to formal systems) without referencing any task-specific data or goals.

Task Description (Objective & Contextual Boundary): Define the problem domain and the high-level goal. It should describe the "what" (the input nature and the intended outcome) and set the contextual boundaries, ensuring the agent knows the scope of the current environment.

Instruction (Operational Guidelines & Constraints): Provide a list of Actionable Guidelines in a concise, bulleted format. This section should manage the "how-to" logic: incorporating specific insight strategies, referencing knowledge IDs, avoiding failure patterns, and following environment-specific constraints (e.g., precedence rules or formatting requirements). It should act as a set of professional "tips and guardrails".

 # Extracted structured task knowledge:

{knowledge_schema}

 # APT METHOD DESCRIPTION(Include principles, modules with output format):

{apt_method_description}

Figure 5: ADAPT prompt for step2