ETHICS STATEMENT

This work studies prompt optimization techniques for language models (LLMs) to better elicit their capabilities in solving target tasks. The primary potential risks of this research are related to the misuse of LLMs, for example, generating misleading, harmful, or biased content.

In our experiments, we only use publicly available datasets and pre-trained LLMs, and no private or sensitive data were involved. Specific statements on LLM usage can be found in Appendix A. We emphasize that our methods are intended for research and benchmarking purposes, and we encourage responsible use to mitigate potential societal risks.

REPRODICIBILITY STATEMENT

We are committed to ensuring the reproducibility of our work. To facilitate replication, we provide the following details:

Computational Resources The following describes the experimental environment, including detailed information on both hardware and software configurations.

- **Hardware**. All experiments were conducted on a computing node equipped with four NVIDIA Tesla V100-SXM2 GPUs (32GB memory each), an Intel Xeon Gold 6248 CPU @ 2.50GHz with 20 cores, and 226 GB of RAM.
- **Software**. The system runs Ubuntu 20.04.6 LTS with Linux kernel version 5.4.0. All models were implemented in Python 3.10.18 using PyTorch 2.0.0 with CUDA 11.7.

Hyperparameter Details In order to isolate the effect of our proposed method and ensure a fair comparison, we mainly followed the default configurations used in baseline methods and intentionally introduced no additional trainable parameters. Specifically, the detailed hyperparameter settings are given below.

- **Initial Population Size**. Following the setup of EvoPrompt, which uses both human-written and LLM-generated prompts, we adopted a similar strategy in spirit but tailored it to our fully automated framework. (1) We identify a fixed set of components through preliminary study mentioned at ref . (2) For each component, we use an LLM to generate 10 candidate values based on prompt templates. (3) We then randomly combine these values to create 10 initial prompts, which together form the initial population for the evolutionary process.
- **Temperature**. Since the stochasticity of LLM outputs is sensitive to temperature settings, we set the temperature to 0.5 to strike a balance between exploration and exploitation. This choice aligns with prior work such as EvoPrompt.
- **Sample Allocation**. For data splits, we followed the protocols of APE and EvoPrompt. Specifically, if the dataset has a predefined training/testing split, we used it as-is. For datasets without predefined splits, we randomly selected 100 examples as the test set and used the remaining examples for training.
- **Randomness Control**. To ensure reproducibility. Unless otherwise noted, we use 3 random seeds (5, 10 and 15) in the training phrase, and reported the results on the test set.

LIMITATIONS

While our framework can adaptively design well-matched prompts for any LLM across diverse downstream tasks, several limitations remain. (1) Due to substantial computational costs, we cannot comprehensively evaluate all models and domains. Instead, we focused on widely used datasets to balance fairness and coverage. (2) Although we report monetary cost based on actual token usage, variations in token pricing across input and output types cannot be precisely captured by the API. Analysis indicates that most of the cost arises from including memory content as input tokens, while output token consumption remains relatively modest, particularly when "thinking mode" is disabled. Future work will explore prompt compression to further optimize resource use. (3) We evaluated only representative component values from each category due to resource constraints. Nevertheless, even with this limited set, our approach continues to outperforms or remains competitive with baselines, demonstrating its effectiveness and suggesting that its benefits will likely increase as LLMs support longer contexts.

REFERENCES

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and Torsten Hoefler. Graph of thoughts: Solving elaborate problems with large language models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(16):17682–17690, March 2024. ISSN 2159-5399. doi: 10.1609/aaai.v38i16.29720. URL http://dx.doi.org/10.1609/aaai.v38i16.29720.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA, 2020. Curran Associates Inc. ISBN 9781713829546.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks, 2023. URL https://arxiv.org/abs/2211.12588.

DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL https://arxiv.org/abs/2501.12948.

DeepSeek Chat. Deepseek chat web interface, 2025. URL https://chat.deepseek.com/. Accessed: 2025-08.

Yihe Deng, Weitong Zhang, Zixiang Chen, and Quanquan Gu. Rephrase and respond: Let large language models ask better questions for themselves, 2024. URL https://arxiv.org/abs/2311.04205.

Shizhe Diao, Pengcheng Wang, Yong Lin, Rui Pan, Xiang Liu, and Tong Zhang. Active prompting with chain-of-thought for large language models, 2024. URL https://arxiv.org/abs/2302.12246.

Longyu Feng, Mengze Hong, and Chen Jason Zhang. Auto-demo prompting: Leveraging generated outputs as demonstrations for enhanced batch prompting. *arXiv preprint arXiv:2410.01724*, 2024.

Chrisantha Fernando, Dylan Banarse, Henryk Michalewski, Simon Osindero, and Tim Rocktäschel. Promptbreeder: self-referential self-improvement via prompt evolution. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

Yao Fu, Hao Peng, Ashish Sabharwal, Peter Clark, and Tushar Khot. Complexity-based prompting for multi-step reasoning, 2023. URL https://arxiv.org/abs/2210.00720.

Qingyan Guo, Rui Wang, Junliang Guo, Bei Li, Kaitao Song, Xu Tan, Guoqing Liu, Jiang Bian, and Yujiu Yang. Evoprompt: Connecting llms with evolutionary algorithms yields powerful prompt optimizers, 2025. URL https://arxiv.org/abs/2309.08532.

Jia He, Mukund Rungta, David Koleczek, Arshdeep Sekhon, Franklin X Wang, and Sadid Hasan. Does prompt formatting have any impact on llm performance? *arXiv preprint arXiv:2411.10541*, 2024.

Tushar Khot, Harsh Trivedi, Matthew Finlayson, Yao Fu, Kyle Richardson, Peter Clark, and Ashish Sabharwal. Decomposed prompting: A modular approach for solving complex tasks, 2023. URL https://arxiv.org/abs/2210.02406.

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

Xiaoxi Li, Guanting Dong, Jiajie Jin, Yuyao Zhang, Yujia Zhou, Yutao Zhu, Peitian Zhang, and Zhicheng Dou. Search-o1: Agentic search-enhanced large reasoning models. *CoRR*, abs/2501.05366, 2025. doi: 10.48550/ARXIV.2501.05366. URL https://doi.org/10.48550/arXiv.2501.05366.

Ranjita Naik, Varun Chandrasekaran, Mert Yuksekgonul, Hamid Palangi, and Besmira Nushi. Diversity of thought improves reasoning abilities of llms, 2024. URL https://arxiv.org/abs/2310.07088.

Krista Opsahl-Ong, Michael J Ryan, Josh Purtell, David Broman, Christopher Potts, Matei Zaharia, and Omar Khattab. Optimizing instructions and demonstrations for multi-stage language model programs. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 9340–9366, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.525. URL https://aclanthology.org/2024.emnlp-main.525/.

Reid Pryzant, Dan Iter, Jerry Li, Yin Lee, Chenguang Zhu, and Michael Zeng. Automatic prompt optimization with "gradient descent" and beam search. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 7957–7968, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.494. URL https://aclanthology.org/2023.emnlp-main.494/.

Victor Sanh, Albert Webson, Colin Raffel, Stephen H. Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, Manan Dey, M Saiful Bari, Canwen Xu, Urmish Thakker, Shanya Sharma Sharma, Eliza Szczechla, Taewoon Kim, Gunjan Chhablani, Nihal Nayak, Debajyoti Datta, Jonathan Chang, Mike Tian-Jian Jiang, Han Wang, Matteo Manica, Sheng Shen, Zheng Xin Yong, Harshit Pandey, Rachel Bawden, Thomas Wang, Trishala Neeraj, Jos Rozen, Abheesht Sharma, Andrea Santilli, Thibault Fevry, Jason Alan Fries, Ryan Teehan, Tali Bers, Stella Biderman, Leo Gao, Thomas Wolf, and Alexander M. Rush. Multitask prompted training enables zero-shot task generalization. In *International Conference on Learning Representations (ICLR)*, 2022. URL https://arxiv.org/abs/2110.08207.

Jie-Jing Shao, Xiao-Wen Yang, Bo-Wen Zhang, Baizhi Chen, Wen-Da Wei, Guohao Cai, Zhenhua Dong, Lan-Zhe Guo, and Yu feng Li. Chinatravel: A real-world benchmark for language agents in chinese travel planning, 2024. URL https://arxiv.org/abs/2412.13682.

Weijia Shi, Julian Michael, Suchin Gururangan, and Luke Zettlemoyer. Nearest neighbor zero-shot inference. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (eds.), *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 3254–3265, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-main.214. URL https://aclanthology.org/2022.emnlp-main.214/.

Kashun Shum, Shizhe Diao, and Tong Zhang. Automatic prompt augmentation and selection with chain-of-thought from labeled data. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 12113–12139, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.811. URL https://aclanthology.org/2023.findings-emnlp.811/.

Xingchen Wan, Ruoxi Sun, Hootan Nakhost, and Sercan Ö. Arı k. Teach better or show smarter? on instructions and exemplars in automatic prompt optimization. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 58174–58244. Curran Associates, Inc., 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/6b031defd145b02bed031093d8797bb3-Paper-Conference.pdf.

Lei Wang, Wanyu Xu, Yihuai Lan, Zhiqiang Hu, Yunshi Lan, Roy Ka-Wei Lee, and Ee-Peng Lim. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Proceedings* *of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2609–2634, Toronto, Canada, July 2023a. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.147. URL https://aclanthology.org/2023.acl-long.147/.

Ming Wang, Yuanzhong Liu, Xiaoyu Liang, Songlian Li, Yijie Huang, Xiaoming Zhang, Sijia Shen, Chaofeng Guan, Daling Wang, Shi Feng, et al. Langgpt: Rethinking structured reusable prompt design framework for llms from the programming language. *arXiv preprint arXiv:2402.16929*, 2024.

Xinyuan Wang, Chenxi Li, Zhen Wang, Fan Bai, Haotian Luo, Jiayou Zhang, Nebojsa Jojic, Eric P. Xing, and Zhiting Hu. Promptagent: Strategic planning with language models enables expert-level prompt optimization, 2023b. URL https://arxiv.org/abs/2310.16427.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models, 2023c. URL https://arxiv.org/abs/2203.11171.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2023. URL https://arxiv.org/abs/2201.11903.

Jinyu Xiang, Jiayi Zhang, Zhaoyang Yu, Fengwei Teng, Jinhao Tu, Xinbing Liang, Sirui Hong, Chenglin Wu, and Yuyu Luo. Self-supervised prompt optimization, 2025. URL https://arxiv.org/abs/2502.06855.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V. Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers, 2024. URL https://arxiv.org/abs/2309.03409.

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

Mert Yuksekgonul, Federico Bianchi, Joseph Boen, Sheng Liu, Pan Lu, Zhi Huang, Carlos Guestrin, and James Zou. Optimizing generative ai by backpropagating language model feedback. *Nature*, 639:609–616, 2025.

Wenxuan Zhang, Yue Deng, Bing Liu, Sinno Pan, and Lidong Bing. Sentiment analysis in the era of large language models: A reality check. In Kevin Duh, Helena Gomez, and Steven Bethard (eds.), *Findings of the Association for Computational Linguistics: NAACL 2024*, pp. 3881–3906, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-naacl.246. URL https://aclanthology.org/2024.findings-naacl.246/.

Yue Zhang, Leyang Cui, Deng Cai, Xinting Huang, Tao Fang, and Wei Bi. Multi-task instruction tuning of llama for specific scenarios: A preliminary study on writing assistance, 2023. URL https://arxiv.org/abs/2305.13225.

Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. Automatic chain of thought prompting in large language models, 2022. URL https://arxiv.org/abs/2210.03493.

Ruochen Zhao, Xingxuan Li, Shafiq Joty, Chengwei Qin, and Lidong Bing. Verify-and-edit: A knowledge-enhanced chain-of-thought framework. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 5823–5840, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.320. URL https://aclanthology.org/2023.acl-long.320/.

Yuxiang Zheng, Dayuan Fu, Xiangkun Hu, Xiaojie Cai, Lyumanshan Ye, Pengrui Lu, and Pengfei Liu. Deepresearcher: Scaling deep research via reinforcement learning in real-world environments, 2025. URL https://arxiv.org/abs/2504.03160.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, and Ed Chi. Least-to-most prompting enables complex reasoning in large language models, 2023a. URL https://arxiv.org/abs/2205.10625.

Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. Large language models are human-level prompt engineers, 2023b. URL https://arxiv.org/abs/2211.01910.

A USE OF LLMS

Large Language Models (LLMs) were used in two ways in this work. First, LLMs served as base models in our experiments on prompt optimization, where we studied how different prompts can elicit their capabilities to solve target tasks. Second, LLMs were employed as auxiliary tools for minor writing support, such as grammar checking and phrasing improvements. Specific details about the LLMs used in our experiments can be found in Appendix B. No LLMs were used to generate substantive ideas, analyses, or content of the paper.

B DETAILS OF DATASETS AND LLMS USED

Datasets For fair comparison, we followed the datasets and evaluation metrics used in prior baselines whenever possible. Specifically, we include 4 classic NLP benchmarks (*MR*, *Subj*, *CoLA*, *SST-5*) and two widely used question-answering datasets (*SQuAD*, *TREC*) to validate basic capabilities; several domain-specific benchmarks to probe specialized performance, including *Financial Sentiment Evaluation* dataset (*FinFE*), *Financial PhraseBank* (*FinPB*), reasoning related dataset (*Casual Judgement*). Besides, one multi-domain datasets (*AG's News*) and one natural language generation dataset (*SAMSum*) are also used to assess overall robustness. To evaluate output quality beyond simple accuracy, we report ROUGE-Avg on *SAMSum* and the Matthews correlation coefficient (MCC) on *CoLA*. To balance computational cost while maximizing coverage, we selected datasets according to a "maximize capability diversity" principle — for example, in addition to the main experiments we ran Qwen2.5-7B-Instruct on *Subj*, *AG's News*, and *FinFE* to cover several of the categories above. Detailed results are presented in the experimental analysis section.

LLMs To demonstrate the adaptability of the proposed method for LLMs, we selected *DeepSeek-R1-Distill-Llama-8B* and *Qwen2.5-7B-Instruct* from open-source LLMs, as well as *GPT-4o-mini* from closed-source LLMs, as the base models for our experiments. The experiments on *DeepSeek-R1-Distill-Llama-8B* evaluate both the performance of the DeepSeek model itself and, to some extent, the capabilities of the underlying Llama architecture, which is primarily trained on English-language data. Experiments on *Qwen2.5-7B-Instruct* assess the framework's performance on a model predominantly trained on Chinese-language data, demonstrating applicability to non-English corpora. *GPT-4o-mini* was included because it is a widely used closed-source model in prior studies and allows cost-effective experimentation within our budget.

C ALGORITHM DETAILS

Algorithm 1 An Overview of **DelvePO**

Require: A population of prompts $\boldsymbol{P}$, size of population N, task-related dataset $\boldsymbol{D}$, number of epochs m, number of iterations n, working memory $\boldsymbol{M} = \{\boldsymbol{M}_{\text{components}}, \boldsymbol{M}_{\text{prompts}}\}$

Ensure: Best prompt p^*

Initialization: $\boldsymbol{P} = \{p_1, p_2, \cdots, p_N\}, \boldsymbol{M}_{\text{prompts}} \leftarrow f_{sort}(\boldsymbol{P}), \boldsymbol{M}_{\text{components}} \leftarrow \emptyset$
for epoch = 1 to m **do**
    $\boldsymbol{P}_{\text{evo}} \leftarrow \emptyset$
    for step = 1 to n **do**
        Selection: $p \leftarrow f_{r.w.s.}(\boldsymbol{P})$
        Task-Evolution: $\mathcal{T}_{\text{evo}} \leftarrow \phi^{\mathcal{T}}(p, \boldsymbol{M}_{\text{components}} \mid \mathcal{T})$
        Solution-Evolution: $\mathcal{S}_{\text{evo}} \leftarrow \phi^{\mathcal{S}}(p, \boldsymbol{M}_{\text{prompts}} \mid \mathcal{T}_{\text{evo}})$
        Evaluation: $p' \leftarrow \phi^{\mathcal{LLM}}(\mathcal{S}_{\text{evo}}),\ s' \leftarrow f_{eval}(p', \boldsymbol{D})$
        Memory-Evolution: $\boldsymbol{M}_{\text{evo}} \leftarrow \phi^{\mathcal{M}}\big(\boldsymbol{M}, \langle p, p', s \geq s' \rangle\big)$
        $\boldsymbol{P}_{\text{evo}} \leftarrow \{\boldsymbol{P}_{\text{evo}}, p'\}$
    end for
    Update: $\boldsymbol{P} \leftarrow \text{Top-}N\, \{\boldsymbol{P}, \boldsymbol{P}_{\text{evo}}\}$
end for
Return the best prompt p^*: $p^* \leftarrow \arg\max_{p \in \boldsymbol{P}} f_{eval}\big(\phi^{\mathcal{LLM}}(p, \boldsymbol{D})\big)$

The sampling function used in our framework is roulette wheel selection, denoted as $f_{r.w.s.}(\cdot)$, which is commonly used in the evolution algorithm. $\phi^{\mathcal{T}}$, $\phi^{\mathcal{S}}$, $\phi^{\mathcal{M}}$ refer to the Task-Evolution, Solution-Evolution, Memory-Evolution methods, respectively. Similarly, $\mathcal{T}$, $\mathcal{S}$, and $\mathcal{M}$ mean the corresponding Task, Solution, Memory. Based on the components, we designed a task-agnostic template described in Figure 4, through which any kind of LLMs can construct an initial content set of components based on a simple description of the target task input by the user.

================= Task-agnostic Template for Component =================

Hi there, I have a task to do which can be described as ***Downstream Task Related Information***. Now I want you to give me N contents related to Component. [OPTIONAL Example]. Please list your answers in the following format: ['content 1', 'content 2',...]

-------------------------- Downstream Task (**Causal judgement**) for role --------------------------

<Query>: Hi there, I have a task to do which can be described as "answer questions about causal attribution". Now I want you to give me 10 related roles who are expertise in these questions. For example, 'Casusal Analysis Experts', etc. Please list your answers in the following format: ["content 1", "content 2",...]

<Response>: ["Cognitive Scientist", "Social Psychologist", "Computational Linguist", "AI Ethicist", "Behavioral Economist", "Decision Theorist", "Philosophy of Mind Researcher", "Causal Inference Data Scientist", "Educational Psychologist", "Human-Computer Interaction Specialist"]

Figure 4: Task-agnostic template for generating component values corresponding to the given component types. *The following part of the figure is the prompt to generate content for Component "role" using the casual judgement task as an example.*

D Additional Experiments

Table 5: The results on different downstream tasks for Qwen2.5-7B-Instruct.

Method	Classical NLP			Question-Answering	Domain-specific	Multi-domain	Avg.
	Subj	SST-5	CoLA	TREC	FinFE	AG's News	
APE	69.00 (3.06)	47.00 (1.10)	79.05 (1.73)	43.40 (1.14)	64.30 (2.70)	83.43 (1.90)	64.38
EvoPrompt	<u>77.03</u> (4.74)	<u>57.67</u> (1.19)	<u>79.69</u> (1.42)	<u>67.55</u> (2.08)	<u>64.67</u> (1.22)	<u>85.73</u> (1.29)	<u>72.06</u>
DelvePO	**80.07** (0.65)	**60.00** (1.69)	**81.40** (1.07)	**70.77** (1.74)	**69.97** (0.87)	**89.27** (0.97)	**75.25**

Table 6: Average monetary cost (USD) for one epoch of optimization on GPT-4o-mini.

Methods	Subj	CoLA	FinPB	AG's News
Promptbreeder	1.17	1.31	0.97	1.52
APE	0.57	0.56	0.61	0.79
EvoPrompt	0.83	0.64	0.74	1.23
DelvePO	1.27	1.08	1.30	1.10

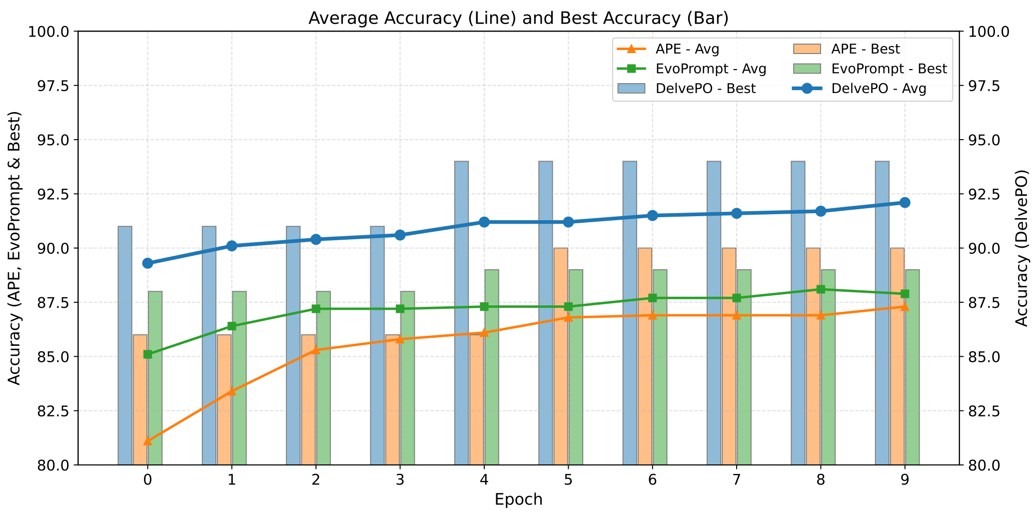

Figure 5: Robustness of DelvePO as the number of epochs increases (Take the dataset MR as an example).

E Detailed Information about Components

To ensure that the types of components are as comprehensive and representative as possible, we first surveyed a broad set of related literature (Yuksekgonul et al., 2025; He et al., 2024; Feng et al., 2024; Opsahl-Ong et al., 2024; Diao et al., 2024; Wang et al., 2024; 2023b) and extracted a variety of factors that have been shown to influence the performance of prompts, forming our component pool. We then categorized all components in the pool based on the semantics implied in their original sources, which resulted in five categories: "Role and Expertise", "Task Content", "Constraints and Norms", "Process and Behavior" and "Context and Examples". From each category, we selected the most representative component as our predefined component types. The complete component pool and its categorization are provided in Table 7.

Despite this extensive literature review, we acknowledge that some important aspects may remain uncovered. This observation motivated our design: as more non-AI experts begin to use LLMs, domain specialists should be able to adaptively define new components through our mechanism, thereby supporting both effective task performance and improved interpretability. It is worth noting that for each component type, we can add a "null" option when generating its values, allowing the presence or absence of the component to be controlled and makes the optimized prompts more flexible.

Table 7: The categories and types of components in the component pool

Categories	Related Items
Role and Expertise	Role; Role description; Scenario; Domain knowledge; Term Clarification
Task Content	Task description; Instruction; Goal
Constraints and Norms	Output format; Constraints; Principle; Style; Length; Tone; Priority & Emphasis; Exception handling; Target audience
Process and Behavior	Workflow; CoT; Action; Skill; Suggestions; Initialization
Context and Examples	Examples; Reference prompt; Attachment

F Template for Injection & Prompts for Evaluation on LLMs

Template_For_Injection $^{\text{General Form}}$ =
··· *<component1>*{content1}*</component1>*. Given the Input, ··· *<component2>*{content2}*</component2>* ···

Template_For_Injection $^{\text{AG's News}}$ =
You are a <role>{role}</role>. Given the News, your task is to <task_description>{task_description}</task_description>.

Template_For_Injection $^{\text{Simplification}}$ =
You are a <role>{role}</role>. Given the English Sentence, your task is to <task_description>{task_description}</task_description>.

Figure 6: Template for initializing prompt populations. *It is also used in the construction of Prompts Memory, that is, injecting discrete components into the template to obtain a continuous form prompt. The above shows the general form, while the two below provide illustrative examples.*

Prompt_For_LLM $^{\text{General Form}}$ =
<INSTRUCTION>: $\cdots$ {**content1**}. Given the Input, $\cdots$ {**content2**} $\cdots$
<Input>: {**input**}
<OUTPUT FORMAT>: Output the final result starting with the tag <res> and ending with the tag </res>. [OPTIONAL REQUIREMENTS]

Prompt_For_LLM $^{\text{AG's News}}$ =
<INSTRUCTION>: You are a {**role**}. Given the News, your task is to {**task_description**}.
<News>: {**input**}
<OUTPUT FORMAT>: Output the final result starting with the tag <res> and ending with the tag </res>. The final result must come from the following: [World, Sports, Business, Tech].

Prompt_For_LLM $^{\text{Simplification}}$ =
<INSTRUCTION>: You are a {**role**}. Given the English Sentence, your task is to {**task_description**}.
<English Sentence>: {**input**}
<OUTPUT FORMAT>: Output the final result starting with the tag <res> and ending with the tag </res>.

Figure 7: Complete prompt template for LLMs (including three parts: instruction, input, and output). *Here we also display two practical prompts for AG's News and Simplification Tasks*.

G THE DETAILED PROMPTS OF TASK-EVOLUTION

Please follow the instructions step-by-step to get **final result**.

Step 1 Conclude **Insights** from the provided Memory Components, which consists of multiple elements. Each element contains two lists: the first contains several markup pairs in the format <component>content</component>. For example, in the pair <role>role_description</role>, the content ("role_description") describes the component ("role"). All markup pairs follow this structure. By default, the first list in each element is considered to perform better than the second.
Memory Components: $\{M_{\text{components}}\}$

Step 2 Based on the **Insights** from Step 1 and the Current Prompt, select one or more component(s) from Component Set that could potentially improve performance to form **final result**. Separate the final result with a special token '|' and ensure that each of final result is unique and appears only once. The final result must start with the tag <res> and end with the tag </res> . For example, the final result must follow the format: <res>component1|...</res>.
Current Prompt: { p }
Component Set: {components}

Figure 8: The prompts for sub-task I

Please follow the instructions step-by-step to get **final result**.

Step 1 Conclude **Insights** from the provided Memory Components, which consists of multiple elements. Each element contains two lists: the first contains several markup pairs in the format <component>content</component>. For example, in the pair <role>role_description</role>, the content ("role_description") describes the component("role"). All markup pairs follow this structure. By default, the first list in each element is considered to perform better than the second.
Memory Components: $\{M_{\text{components}}\}$

Step 2 Given a list named Old Values, where each element contains a pair of contents, use the **Insights** from Step 1 to select one content from each pair in original order. The **final result** must start with the tag <res> and end with the tag </res> . For example, the final results must follow the format: <res>content1|...</res>.
Old Values: {old_values}

Figure 9: The prompts for sub-task II

H THE DETAILED PROMPTS OF SOLUTION-EVOLUTION

Please follow the instructions step-by-step to get **final result**.

Step 1 Conclude the **Insights** from the Memory Prompts, which consists of multiple items. Each item includes two parts: the first part contains several markup pairs in the format <component>content</component>. For example, in the pair <role>role_description</role>, the content ("role_description") describes the component ("role"). Other markup pairs follow this same structure. The second part of each item represents its corresponding performance. The entire Memory Prompts is sorted in descending order based on performance.
Memory Prompts: $\{M^{\text{discrete}}_{\text{prompts}}\}$

Step 2 Given a list named Old Values, use the **Insights** from Step 1 to generate a new mutated content for each content to form a new list, i.e. **final result**, referring to Description, adhering to Rules below.

> Description:
>
> - In Old Values, each element is a markup pair like <component>content</component> containing content that needs to mutate.
>
> Rules:
>
> 1. Mutation Requirements:
>   - For each element like <component>content</component>, generate a **new one content** that:
>   - If the component is <role>, the new content must be a **noun phrase** describing a person.
>   - If the component is <task_description>, the new content must be a **verb phrase** describing a task.
>   - Is **distinct** from the original content.
>   - Preserves lexical identity (noun/verb phrase) matching the component.
>   - If the original content had the **highest score**, the new content must prioritize **improved performance potential** (e.g., higher efficiency, enhanced properties).
>   - Otherwise, the new content may be derived from those contents linked to its corresponding component in the Memory Prompts (optional but allowed).
> 2. Output Format:
>   - Start with <res> and end with </res>.
>   - Separate mutated contents **strictly** with '|' (no extra characters).
>   - Never include original contents in the output.

Old Values: {old_values}

Figure 10: The prompts for Sub-solution I - Prompts Memory in discrete form

Please follow the instructions step-by-step to get **final result**.

Step 1 Conclude the **Insights** from the Memory Prompts, which contains multiple items. Each item has two parts: a sentence enclosed in <prompt> and </prompt>, and its corresponding performance score. The sentence includes markup pairs in the format <component>content</component>, where the content describes the component. For example, <role>role_description</role> indicates that "role_description" explains the "role" component. All items are sorted in descending order by performance.

Memory Prompts: $\{M_{\text{prompts}}^{\text{continuous}}\}$

Step 2 Based on the Current Prompt and **Insights** from Step 1, generate a new mutated content for each markup pair whose component matches those listed in Mutate Factors to form the **final result**, referring to Description, adhering to Rules below.

> Description:
>
> - In Current Prompt, markup pair like <component>content</component> contains content that needs to mutate.
> - In Mutate Factors, each element is a component appeared in Current Prompt.
>
> Rules:
>
> 1. Mutation Requirements:
>   - For each markup pair like <component>content</component>, if the component in Mutate Factors, generate a **new one content** that:
>   - If the component is <role>, the new content must be a **noun phrase** describing a person.
>   - If the component is <task_description>, the new content must be a **verb phrase** describing a task.
>   - Is **distinct** from the original content.
>   - Preserves lexical identity (noun/verb phrase) matching the component.
>   - If the original content had the **highest score**, prioritize generating contents with **improved performance potential** (e.g., higher efficiency, enhanced properties).
>   - Otherwise, the new content may derive from those contents linked to its component in the Memory Prompts (optional but allowed).
> 2. Output Format:
>   - Start with <prompt> and end with </prompt>.
>   - **Only mutate contents within markup pairs specified in Mutate Factors**.
>   - Preserve all other values outside markup pairs.
>   - Replace original contents with mutated ones directly within their components.

Current Prompt: {prompt}
Mutate Factors: {mutate_factors}

Figure 11: The prompts for Sub-solution I - Prompts Memory in continuous form

Please follow the instructions step-by-step to get **final result**.

Step 1 Conclude the **Insights** from the Memory Prompts, which consists of multiple items. Each item includes two parts: the first part contains several markup pairs in the format <component>content</component>. For example, in the pair <role>role_description</role>, the content ("role_description") describes the component ("role"). Other markup pairs follow this same structure. The second part of each item represents its corresponding performance. The entire Memory Prompts is sorted in descending order based on performance.
Memory Prompts: $\{M_{\text{prompts}}^{\text{discrete}}\}$

Step 2 Given a list named Old Values, where each element contains a pair of contents, use the **Insights** from Step 1 to generate a new mutated content for each pair to form a new list, i.e. **final result**, referring to Description, adhering to Rules below.
Old Values: {old_values}

Description:

- In Old Values, each element contains a pair of contents like **[a, b]**.

Rules:

1. Mutation Requirements:
    - For each pair of contents like **[a, b]**, generate a **new one content** that:
        - If **a** and **b** are enclosed with <role> & </role>, the new content must be a noun phrase used to describe a person.
        - If **a** and **b** are enclosed with <task_description> & </task_description>, the new content must be a verb phrase used to describe a task.
        - Is **distinct** from both **a** and **b**.
        - Preserve corresponding lexical identity.
        - If the original pair has the **highest score**, prioritize generating contents with **improved performance potential** (e.g., higher efficiency, enhanced properties).
        - Otherwise, derive the new content from those contents linked to its component in the Memory Prompts (optional but allowed).
2. Output Format:
    - Start with <res> and end with </res>.
    - Separate mutated contents **strictly** with '|' (no extra characters).
    - Never include original pairs in the output.

Figure 12: The prompts for Sub-solution II - Prompts Memory in discrete form

Please follow the instructions step-by-step to get **final result**.

Step 1 Conclude the **Insights** from the Memory Prompts, which contains multiple items. Each item has two parts: a sentence enclosed in <prompt> and </prompt>, and its corresponding performance score. The sentence includes markup pairs in the format <component>content</component>, where the content describes the component. For example, <role>role_description</role> indicates that "role_description" explains the "role" component. All items are sorted in descending order by performance.

Memory Prompts: $\{M^{\text{continuous}}_{\text{prompts}}\}$

Step 2 Based on the Prompt 1 and **Insights** from Step 1, generate a new mutated content for each markup pair whose component matches those listed in Mutate Factors to form the Prompt 2, referring to Description, adhering to Rules below.

> Description:
>
> - In Prompt 1, markup pair like <component>content</component> contains content that needs to mutate.
> - In Mutate Factors, each element is a content appeared in Prompt 1.
>
> Rules:
>
> 1. Mutation Requirements:
>   - For each markup pair like <component>content</component>, if the component in Mutate Factors, Generate a **new one content** that:
>   - If the component is <role>, the new content must be a **noun phrase** describing a person.
>   - If the component is <task_description>, the new content must be a **verb phrase** describing a task.
>   - Is **distinct** from the original content.
>   - Preserves lexical identity (noun/verb phrase) matching the component.
>   - If the original content had the **highest score**, prioritize generating contents with **improved performance potential** (e.g., higher efficiency, enhanced properties).
>   - Otherwise, the new content may derive from those contents linked to its component in the Memory Prompts (optional but allowed).
> 2. Output Format:
>   - Start with <prompt> and end with </prompt>.
>   - **Only mutate contents within markup pairs specified in Mutate Factors.**
>   - Preserve all other values outside markup pairs.
>   - Replace original contents with mutated ones directly within their components.

Prompt 1: {prompt1}

Mutate Factors: {mutate_factors}

Step 3 Based on the Prompt 3 and **Insights** from Step 1, generate a new mutated content for each markup pair whose component matches those listed in Mutate Factors to form the Prompt 4, referring to Description, adhering to Rules below.

> Description:
>
> - In Prompt 3, markup pair like <component>content</component> contains content that needs to mutate.

Figure 13: The prompts for Sub-solution II - Prompts Memory in continuous form

> - In Mutate Factors, each element is a content appeared in Prompt 3.
>
> Rules:
>
> 1. Mutation Requirements:
>   - For each markup pair like <component>content</component>, if the component in Mutate Factors, Generate a **new one content** that:
>   - If the component is <role>, the new content must be a **noun phrase** describing a person.
>   - If the component is <task_description>, the new content must be a **verb phrase** describing a task.
>   - Is **distinct** from the original content.
>   - Preserves lexical identity (noun/verb phrase) matching the component.
>   - If the original content had the **highest score**, prioritize generating contents with **improved performance potential** (e.g., higher efficiency, enhanced properties).
>   - Otherwise, the new content may derive from those contents linked to its component in the Memory Prompts (optional but allowed).
> 2. Output Format:
>   - Start with <prompt> and end with </prompt>.
>   - **Only mutate contents within markup pairs specified in Mutate Factors**.
>   - Preserve all other values outside markup pairs.
>   - Replace original contents with mutated ones directly within their components.

Prompt 3: {prompt3}
Mutate Factors: {mutate_factors}

Step 4 Generate **final result** by selecting contents from pairs in Prompt 2 and Prompt 4 under identical markup components, referring to Description, adhering to Rules below.

> Description:
>
> - Pairs from Prompt 2 and Prompt 4 have identical components (e.g., <role>, <task_description>).
>
> Rules:
>
> 1. Selection Criteria:
>   - For each tagged pair (e.g., <role>a</role> and <role>b</role>):
>   - Use **Insights** from Step 1 to **select one content** (a or b) that has **higher performance improvement potential** (e.g., clarity, specificity, alignment with goals).
>   - If the component is <role>, the new content must be a **noun phrase** describing a person.
>   - If the component is <task_description>, the new content must be a **verb phrase** describing a task.
>   - Preserve the lexical identity of the component.
>   - Never modify text **outside** markup pairs.
> 2. Output Format:
>   - Start with <prompt> and end with </prompt>.
>   - Retain the structure of Prompt 3 but replace tagged pairs with the selected contents.
>   - If multiple tagged pairs exist, update all while maintaining non-tagged values verbatim.

Figure 14: The prompts for Sub-solution II - Prompts Memory in continuous form (extended from Figure 13)

I Case Study Details

To quickly verify the generalizability of our framework, we conducted multi-turn dialogues with DeepSeek Chat via the web interface provided by DeepSeek (DeepSeek Chat, 2025).

Throughout the process, we take simplification task (Zhang et al., 2023) as the example, which allows for easy observation and interpretation of the outputs, and randomly set 4 components. The whole process can be find in Appendix I. For Task-Evolution, we provide two input information (see Figure 15, 17) for the prompt of two sub tasks (see Figure 8, 9). And the corresponding outputs are shown in Figure 16, 18. From the final results, we can derive that under the guidance of direction (i.e., Memory Components), The LLMs could find reasonable direction for evolutionary operator.

Accordingly, for Solution-Evolution, we provide four input information (see Figure 19, 21, 23, 25) for four kinds of scenarios. And the corresponding responses are shown in Figure 20, 22, 24, 26, respectively. Based on the observation from the responses, we also find that the procedure designed in this paper can accurately guide the model to deduce corresponding insights and further output reasonable results based on the insights.

By the way, this case study provides new users with a quick understanding of our framework. It can also serve as a practical guide, allowing anyone to construct a prototype using the system prompts provided in the case study without writing any code, which can help users optimize domain-specific prompts. We sincerely invite researchers to try it out and share their valuable feedback for further improvement.

Step 1 **Memory Components**:

```
[
  [
    [<action>Clarify Sentence Meaning</action>, <requirements>Adapt to
Audience</requirements>],
    [<action>Reduce Linguistic Complexity</action>, <requirements>Improve
Structure</requirements>]
  ],
  [
    [<role>Readability Specialist</role>],
    [<role>Sentence Simplifier</role>]
  ]
]
```

Step 2 **Current Prompt**:

```
[
    <role>Sentence Simplifier</role>,
    <task_description>Rewrite the provided sentences in a simpler
form</task_description>,
    <action>Condense Sentences</action>,
    <requirements>Adapt to Audience</requirements>
]
```

Component Set:

```
[role, task_description, action, requirements]
```

Figure 15: The input for sub-task I

Step 1: Conclude **Insights** from the **Memory Components**

1. In the first tuple, <action>Clarify Sentence Meaning</action> is preferred over <action>Reduce Linguistic Complexity</action>, and <requirements>Adapt to Audience</requirements> is preferred over <requirements>Improve Structure</requirements>.
2. In the second tuple, <role>Readability Specialist</role> is preferred over <role>Sentence Simplifier</role>.

Insights:

- **Role**: "Readability Specialist" outperforms "Sentence Simplifier."
- **Action**: "Clarify Sentence Meaning" outperforms "Reduce Linguistic Complexity" (and by extension, other actions like "Condense Sentences").
- **Requirements**: "Adapt to Audience" is already the preferred choice.

Step 2: Select **Components** to improve performance based on **Current Prompt** and **Insights**

- The **Current Prompt** uses <role>Sentence Simplifier</role>, but the **Insights** suggest <role>Readability Specialist</role> is better.
- The **Current Prompt** uses <action>Condense Sentences</action>, but the **Insights** suggest <action>Clarify Sentence Meaning</action> is better.
- The **Current Prompt** already uses the preferred <requirements>Adapt to Audience</requirements>, so no change is needed here.

final result: <res>role|action</res>

Figure 16: The responses for sub-task I

Step 1 **Memory Components**:

```
[
    [
        [<action>Clarify Sentence Meaning</action>, <requirements>Adapt to
Audience</requirements>],                [<action>Reduce Linguistic
Complexity</action>, <requirements>Improve Structure</requirements>]
    ],
    [
        [<role>Readability Specialist</role>],
        [<role>Sentence Simplifier</role>]
    ],
    [
        [<action>Optimize Readability</action>, <task_description>Reduce the
complexity of the provided sentences</task_description>],
        [<action>Reword for Better Clarity</action>, <task_description>Reword the
provided sentences for better clarity</task_description>]
    ]
]
```

Step 2 **Old Values**:

```
[
    [<role>Readability Specialist</role>, <role>Paraphrasing Specialist</role>],
    [<action>Simplify Sentence Structure</action>, <action>Break Down
Complexity</action>]
]
```

Figure 17: The input for sub-task II

Step 1: Conclude Insights from Memory Components

From the Memory Components:

1. **First Element:**
   - The pair <action>Clarify Sentence Meaning</action> with <requirements>Adapt to Audience</requirements> is preferred over <action>Reduce Linguistic Complexity</action> with <requirements>Improve Structure</requirements>.
   - **Insight:** Prioritizing audience adaptation and meaning clarification over general structural improvement and complexity reduction.
2. **Second Element:**
   - <role>Readability Specialist</role> is preferred over <role>Sentence Simplifier</role>.
   - **Insight:** "Readability Specialist" is considered a more effective or broader role than "Sentence Simplifier".
3. **Third Element:**
   - <action>Optimize Readability</action> is preferred over <action>Reword for Better Clarity</action>.
   - **Insight:** Optimizing readability takes precedence over merely rewording for clarity.

Step 2: Apply Insights to Select from Old Values

Old Values:

1. [<role>Readability Specialist</role>, <role>Paraphrasing Specialist</role>]
   - Based on Insight 2, "Readability Specialist" is already preferred, and since "Paraphrasing" is similar to "Sentence Simplifier", "Readability Specialist" aligns better with the insight.
2. [<action>Simplify Sentence Structure</action>, <action>Break Down Complexity</action>]
   - Based on Insight 1 and 3:
     - "Simplify Sentence Structure" relates more directly to optimizing readability and clarifying sentence meaning.
     - "Break Down Complexity" could be interpreted similarly to "Reduce Linguistic Complexity", which was not preferred.
   - So, choose "Simplify Sentence Structure".

Final Output (final results):

```
<res>
  <role>Readability Specialist</role>
  <action>Simplify Sentence Structure</action>
</res>
```

Figure 18: The responses for sub-task II

Step 1 **Memory Prompts**[discrete]:

```
[
    [
        [<role>Sentence Simplifier</role>, <task_description>Clarify the meaning of the given sentences by simplifying them</task_description>, <action>Optimize Readability</action>, <requirements>Improve Structure</requirements>],
        25.0
    ]
    [
        [<role>Readability Specialist</role>, <task_description>Reduce the complexity of the provided sentences</task_description>, <action>Clarify Sentence Meaning</action>, <requirements>Simplify Vocabulary</requirements>],
        21.0
    ]
    [
        [<role>Grammar & Clarity Editor</role>, <task_description>Modify the given sentences to be more straightforward</task_description>, <action>Reduce Linguistic Complexity</action>, <requirements>Adapt to Audience</requirements>],
        16.0
    ]
    [
        [<role>AI Sentence Simplifier</role>, <task_description>Reword the provided sentences for better clarity</task_description>, <action>Simplify Sentence Structure</action>, <requirements>Ensure Conciseness</requirements>],
        14.0
    ]
]
```

Step 2 **Old Values**:

```
[
    <role>Sentence Simplifier</role>,
    <action>Reduce Linguistic Complexity</action>
]
```

Figure 19: The input for Sub-solution I - Prompts Memory in discrete form

Based on the Memory Item List and the Old Values provided, here's how we can generate the Mutated Values:

1. For `<role>Sentence Simplifier</role>`:
    - This role already has the highest performance score (25.0), so we need to create a new value that's different but still relevant.
    - Possible mutation: "Clarity Optimizer" (derived from the action "Optimize Readability" in the top-performing item)
2. For `<action>Reduce Linguistic Complexity</action>`:
    - This action comes from an item with performance score 16.0 (not the highest), so we can derive a new value from higher-performing items.
    - The top-performing item has action "Optimize Readability" (score 25.0)
    - The second-highest has action "Clarify Sentence Meaning" (score 21.0)

Mutated Values:

```
<res>
    <role>Clarity Optimizer</role>,
    <action>Optimize Readability</action>
</res>
```

Figure 20: The responses for Sub-solution I - Prompts Memory in discrete form

Step 1 **Memory Prompts**continuous:

```
[
    [
        <prompt>You are a <role>Sentence Simplifier</role>. Given the English
Sentence, your task is to <task_description>Clarify the meaning of the given sentences
by simplifying them</task_description>. To accomplish this, you need to
<action>Optimize Readability</action>. Please make sure to <requirements>Improve
Structure</requirements> throughout the process.</prompt>,
        25.0
    ]
    [
        <prompt>You are a <role>Readability Specialist</role>. Given the English
Sentence, your task is to <task_description>Reduce the complexity of the provided
sentences</task_description>. To accomplish this, you need to <action>Clarify Sentence
Meaning</action>. Please make sure to <requirements>Simplify Vocabulary</requirements>
throughout the process.</prompt>,
        21.0
    ]
    [
        <prompt>You are a <role>Grammar & Clarity Editor</role>. Given the English
Sentence, your task is to <task_description>Modify the given sentences to be more
straightforward</task_description>. To accomplish this, you need to <action>Reduce
Linguistic Complexity</action>. Please make sure to <requirements>Adapt to
Audience</requirements> throughout the process.</prompt>,
        16.0
    ]
    [
        <prompt>You are a <role>Technical Content Simplifier</role>. Given the English
Sentence, your task is to <task_description>Reword the provided sentences for better
clarity</task_description>. To accomplish this, you need to <action>Simplify Sentence
Structure</action>. Please make sure to <requirements>Ensure
Conciseness</requirements> throughout the process.</prompt>,
        14.0
    ]
]
```

Step 2 **Current Prompt**:

```
<prompt>
    You are a <role>Sentence Simplifier</role>. Given the English Sentence, your task
is to <task_description>Rewrite the provided sentences in a simpler
form</task_description>. To accomplish this, you need to <action>Condense
Sentences</action>. Please make sure to <requirements>Adapt to Audience</requirements>
throughout the process.
</prompt>
```

Step 2 **Mutate Factors**:

```
<res> role | requirements </res>
```

Figure 21: The input for Sub-solution I - Prompts Memory in continuous form

Here is the **Final Prompt** with mutated values for the specified tags (role and requirements), while keeping the rest unchanged:

```
<prompt>
    You are a <role>Clarity Enhancer</role>. Given the English Sentence, your task is
to <task_description>Rewrite the provided sentences in a simpler
form</task_description>. To accomplish this, you need to <action>Condense
Sentences</action>. Please make sure to <requirements>Improve Structure</requirements>
throughout the process.
</prompt>
```

Mutation Justification:

1. **<role>**
   - Original: *"Sentence Simplifier"*
   - Mutated: *"Clarity Enhancer"* (Inspired by high-performing roles like *"Readability Specialist"* and *"Grammar & Clarity Editor"*, but distinct.)
2. **<requirements>**
   - Original: *"Adapt to Audience"*
   - Mutated: *"Improve Structure"* (Drawn from the top-performing prompt's <requirements>, ensuring relevance while avoiding repetition.)

All other tags (task_description, action) remain **unchanged** as they were not in the Mutate Factors. The new values align with high-scoring patterns while being unique.

Figure 22: The responses for Sub-solution I - Prompts Memory in continuous form

Step 1 **Memory Prompts**[discrete]:

```
[
    [
        [<role>Sentence Simplifier</role>, <task_description>Clarify the meaning of
the given sentences by simplifying them</task_description>, <action>Optimize
Readability</action>, <requirements>Improve Structure</requirements>],
        25.0
    ]
    [
        [<role>Readability Specialist</role>, <task_description>Reduce the complexity
of the provided sentences</task_description>, <action>Clarify Sentence
Meaning</action>, <requirements>Simplify Vocabulary</requirements>],
        21.0
    ]
    [
        [<role>Grammar & Clarity Editor</role>, <task_description>Modify the given
sentences to be more straightforward</task_description>, <action>Reduce Linguistic
Complexity</action>, <requirements>Adapt to Audience</requirements>],
        16.0
    ]
    [
        [<role>AI Sentence Simplifier</role>, <task_description>Reword the provided
sentences for better clarity</task_description>, <action>Simplify Sentence
Structure</action>, <requirements>Ensure Conciseness</requirements>],
        14.0
    ]
]
```

Step 2 **Old Values**:

```
[
    [<role>Sentence Simplifier</role>, <role>Paraphrasing Specialist</role>],
    [<action>Reduce Linguistic Complexity</action>, <action>Break Down
Complexity</action>]
]
```

Figure 23: The input for Sub-solution II - Prompts Memory in discrete form

Step 1: Conclude the Insights from the Memory Prompts

From the **Memory Prompts**, we can observe the following insights:

1. **Roles**: The roles are centered around simplifying, clarifying, or modifying sentences to improve readability and reduce complexity. Examples include "Sentence Simplifier," "Readability Specialist," "Grammar & Clarity Editor," and "AI Sentence Simplifier."
2. **Task Descriptions**: The tasks involve simplifying, rewording, or modifying sentences for clarity and straightforwardness. Examples include "Clarify the meaning of the given sentences by simplifying them" and "Reduce the complexity of the provided sentences."
3. **Actions**: The actions focus on optimizing readability, clarifying meaning, reducing linguistic complexity, and simplifying sentence structure. Examples include "Optimize Readability" and "Clarify Sentence Meaning."
4. **Requirements**: The requirements emphasize improving structure, simplifying vocabulary, adapting to the audience, and ensuring conciseness. Examples include "Improve Structure" and "Simplify Vocabulary."

The list is sorted in descending order of performance, with "Sentence Simplifier" being the highest-performing role.

Step 2: Mutate a new value from each pair in Old Values

Old Value Pair 1:

```
[<role>Sentence Simplifier</role>, <role>Paraphrasing Specialist</role>]
```

- **Sentence Simplifier**: From the Memory Prompts, this role involves simplifying sentences for clarity.
- **Paraphrasing Specialist**: Not explicitly in the Memory Prompts, but likely involves rewording sentences while preserving meaning.
- **Mutation**: Combine the ideas of simplifying and rewording to create a new role that emphasizes clarity and adaptation. For example, "Clarity Adaptation Editor" or "Simplification & Paraphrasing Expert."
    - Chosen mutation: "**Clarity Adaptation Editor**"

Old Value Pair 2:

```
[<action>Reduce Linguistic Complexity</action>, <action>Break Down
Complexity</action>]
```

- **Reduce Linguistic Complexity**: From the Memory Prompts, this involves making language less complex.
- **Break Down Complexity**: Similar, but more focused on decomposing complex ideas.
- **Mutation**: Combine the ideas of reducing and breaking down complexity into a more general or refined action. For example, "Streamline Language" or "Simplify and Deconstruct."
    - Chosen mutation: "**Streamline Language**"

Final Mutated Values:

```
<res>
    [<role>Clarity Adaptation Editor</role>, <action>Streamline Language</action>]
</res>
```

Figure 24: The responses for Sub-solution II - Prompts Memory in discrete form

Step 1 **Memory Prompts**[continuous]:

```
[
    [
        <prompt>You are a <role>Sentence Simplifier</role>. Given the English Sentence,
your task is to <task_description>Clarify the meaning of the given sentences by
simplifying them</task_description>. To accomplish this, you need to <action>Optimize
Readability</action>. Please make sure to <requirements>Improve Structure</requirements>
throughout the process.</prompt>,
        25.0
    ]
    [
        <prompt>You are a <role>Readability Specialist</role>. Given the English
Sentence, your task is to <task_description>Reduce the complexity of the provided
sentences</task_description>. To accomplish this, you need to <action>Clarify Sentence
Meaning</action>. Please make sure to <requirements>Simplify Vocabulary</requirements>
throughout the process.</prompt>,
        21.0
    ]
    [
        <prompt>You are a <role>Grammar & Clarity Editor</role>. Given the English
Sentence, your task is to <task_description>Modify the given sentences to be more
straightforward</task_description>. To accomplish this, you need to <action>Reduce
Linguistic Complexity</action>. Please make sure to <requirements>Adapt to
Audience</requirements> throughout the process.</prompt>,
        16.0
    ]
    [
        <prompt>You are a <role>Technical Content Simplifier</role>. Given the English
Sentence, your task is to <task_description>Reword the provided sentences for better
clarity</task_description>. To accomplish this, you need to <action>Simplify Sentence
Structure</action>. Please make sure to <requirements>Ensure Conciseness</requirements>
throughout the process.</prompt>,
        14.0
    ]
]
```

Step 2 **Prompt 1**:

```
<prompt>You are a <role>Sentence Simplifier</role>. Given the English Sentence, your task
is to <task_description>Rewrite the provided sentences in a simpler
form</task_description>. To accomplish this, you need to <action>Condense
Sentences</action>. Please make sure to <requirements>Adapt to Audience</requirements>
throughout the process.
</prompt>
```

Mutate Factors:

```
<res> role | requirements </res>
```

Step 3 **Prompt 3**:

```
<prompt>You are a <role>Linguistic Analyst</role>. Given the English Sentence, your task
is to <task_description>Rewrite the provided sentences in a simpler
form</task_description>. To accomplish this, you need to <action>Condense
Sentences</action>. Please make sure to <requirements>Preserve Meaning</requirements>
throughout the process.
</prompt>
```

Mutate Factors:

```
<res> role | requirements </res>
```

Figure 25: The input for Sub-solution II - Prompts Memory in continuous form

Step 1: Insights from the Memory Prompts

1. **Highest Scoring Item (Score: 25.0)**:
   - **Role**: Sentence Simplifier
   - **Task Description**: Clarify the meaning of the given sentences by simplifying them
   - **Action**: Optimize Readability
   - **Requirements**: Improve Structure
2. **Other Items**:
   - **Role**: Readability Specialist (Score: 21.0)
   - **Role**: Grammar & Clarity Editor (Score: 16.0)
   - **Role**: Technical Content Simplifier (Score: 14.0)

Step 2: Mutate Prompt 1

Mutate Factors: <res> role | requirements </res>

- **Role**: "Sentence Simplifier" is already the highest-scoring role, so we generate a new value: "Clarity Enhancer".
- **Requirements**: "Adapt to Audience" is not the highest-scoring (highest is "Improve Structure"), so we replace it with "Improve Structure".

Prompt 2:

```
<prompt>You are a <role>Clarity Enhancer</role>. Given the English Sentence, your task is
to <task_description>Rewrite the provided sentences in a simpler form</task_description>.
To accomplish this, you need to <action>Condense Sentences</action>. Please make sure to
<requirements>Improve Structure</requirements> throughout the process.</prompt>
```

Step 3: Mutate Prompt 3

Mutate Factors: <res> role | requirements </res>

- **Role**: "Linguistic Analyst" is not in the Memory Item List, so we replace it with the highest-scoring role: "Sentence Simplifier".
- **Requirements**: "Preserve Meaning" is not the highest-scoring, so we replace it with "Improve Structure".

Prompt 4:

```
<prompt>You are a <role>Sentence Simplifier</role>. Given the English Sentence, your task
is to <task_description>Rewrite the provided sentences in a simpler
form</task_description>. To accomplish this, you need to <action>Condense
Sentences</action>. Please make sure to <requirements>Improve Structure</requirements>
throughout the process.</prompt>
```

Step 4: Combine Prompt 2 and Prompt 4

Combine the best-performing markup pairs from Prompt 2 and Prompt 4:

- **Role**: "Clarity Enhancer" (from Prompt 2)
- **Task Description**: "Rewrite the provided sentences in a simpler form" (common in both)
- **Action**: "Condense Sentences" (common in both)
- **Requirements**: "Improve Structure" (common in both)

Final Prompt:

```
<prompt>You are a <role>Clarity Enhancer</role>. Given the English Sentence, your task is
to <task_description>Rewrite the provided sentences in a simpler form</task_description>.
To accomplish this, you need to <action>Condense Sentences</action>. Please make sure to
<requirements>Improve Structure</requirements> throughout the process.</prompt>
```

Figure 26: The responses for Sub-solution II - Prompts Memory in continuous form