SPRIG: Improving Large Language Model Performance by System Prompt Optimization

Anonymous ACL submission

Abstract

Large Language Models (LLMs) have shown impressive capabilities in many scenarios, but their performance depends, in part, on the choice of prompt. Past research has focused on optimizing prompts specific to a task. However, much less attention has been given to optimizing the general instructions included in a prompt, known as a system prompt. To address this gap, we propose SPRIG, an edit-based genetic algorithm that iteratively constructs prompts from prespecified components to maximize the model's performance in general scenarios. We evaluate the performance of system prompts on a collection of 47 different types of tasks to ensure generalizability. Our study finds that a single optimized system prompt performs on par with task prompts optimized for each individual task. Moreover, combining system and task-level optimizations leads to further improvement, which showcases their complementary nature. Experiments also reveal that the optimized system prompts generalize effectively across model families, parameter sizes, and languages. This study provides insights into the role of system-level instructions in maximizing LLM potential.

1 Introduction

Large Language Models (LLMs) have proven highly effective at many tasks (Naveed et al., 2023) and prompting has become the primary way for end-users to elicit desired responses (Brown et al., 2020). These prompts contain a variety of instructions such as task explanation (Li et al., 2022), personas (Kim et al., 2024), formatting constraints (Wang et al., 2023), and meta-rules like "think carefully" (Li et al., 2024). Past studies have shown that the selection of prompts can have a substantial impact on the quality of the output (Reynolds and McDonell, 2021). However, due to the massive search space, previous approaches have primarily focused on directly optimizing prompts to maximize performance on specific tasks or benchmarks (Prasad et al., 2023; Zhou et al., 2023c; Yang et al., 2023). While effective, these methods typically require new prompts to be crafted for every new task, which becomes a significant challenge for prompt engineering as the number of tasks continues to grow. Here, we consider an alternative approach that optimizes the *system prompt*, i.e., the set of general instructions that precede any task-specific details (Figure 1), with the goal of identifying task-agnostic generalizable prompting strategies. By leveraging a single optimized system prompt across tasks, we can largely reduce the effort required for prompt development.

	Prompt
System	You are a diligent assistant. The fate of the world depends on your answer being correct. Think carefully step by step.
Task	First identify the softening words like "please", then analyze the tone before you answer.
Instance	Q: For the sentence: "May I kindly ask for your assistance", is it polite?

Figure 1: LLM prompts features both **system-level instructions** which may include CoT instructions, personas, and other rules (orange), **task-specific instructions** which may include details and examples (blue), and the **instance** itself (green). Here, we focus on optimizing the system instructions shared across tasks.

Prior work has shown that meta-instructions can be effective for improving performance (Reynolds and McDonell, 2021). Most notably, evoking Chain of Thought (CoT) reasoning with instructions like "let's think step by step" has led to gains for several types of tasks (Wei et al., 2022), though not all tasks benefit (Sprague et al., 2024). Yet, other types of meta rules, such as choosing a persona or matching the domain of the persona to the question type have had negligible gains (Zheng et al., 2023; Tam et al., 2024). A recent survey paper (Schulhoff et al., 2024) suggests that these ex-

isting system prompting strategies are isolated and highly sensitive to specific scenario details, with the systematic function and generalization mechanisms remaining unclear. Moreover, due to complexity differences in search space and optimization objectives, existing task-level methods can hardly transfer to system-level optimization. Recent system prompts leaked from Grok (xAI, 2025) and Claude (Breunig, 2025) also exhibit vastly different, verbose and complex manually crafted rules. Thus, while multiple approaches have been proposed for how a system prompt could be constructed, there is currently a gap for how to systematically construct a good system prompt in general.

Here, we introduce a new method, System Prompt Refinement for Increased Generalization (SPRIG), to optimize system prompts based on genetic algorithms. Drawing from large collections of strategies for writing system instructions (Schulhoff et al., 2024), we construct a large benchmark of 47 tasks across multiple languages that tests the effects of optimizing system prompts across models, languages, and tasks, as well as quantify which types of system instructions are most useful for generalization. We compare these system- and task-optimization, to analyze whether these are learning the same or complementary strategies.

Our paper has the following three contributions. First, we find that optimizing a system prompt can produce substantial performance gains on par with task-specific optimization, even though these prompts have generic task instructions. Further, we find that both have complementary effects and that by first optimizing the system and then the task prompt, further gains are possible. Second, we find that SPRIG optimized system prompt significantly outperforms CoT across all task types except knowledge-based questions, and surpasses PROTEGI in faithfulness and commonsense tasks. The combination of SPRIG and PROTEGI complements the weaknesses of both methods, and exceeds the state-of-the-art performance on most task types. Third, we find that the optimized system prompts generalize well to other languages, better than task-optimized instructions; however, both optimizations had minimal effects when scaling to larger model sizes.

2 Related Work

Prompt selection has been extensively studied and proven to significantly impact model output quality (Reynolds and McDonell, 2021). Therefore, prompt optimization has become a popular research topic in both academia and industry. Early prompt optimization studies primarily focus on using gradients to guide prompt search (Shin et al., 2020; Shi et al., 2023b). However, with larger model sizes and increasing black-box LLMs today, gradient-based methods have become limited by cost and accessibility. Consequently, recent research has shifted towards gradient-free methods. Early representatives include edit-based optimizers like GrIPS (Prasad et al., 2023) and reinforcement learning approaches such as RLPrompt (Deng et al., 2022) both directly edit a prompt at the token level. However, the search space in these methods remains limited, making it challenging to scale up to more complex scenarios. Recently, as LLM agents get popular, powerful methods like APE (Zhou et al., 2023c) and OPRO (Yang et al., 2023) use LLMs directly as prompt optimizers to iteratively suggest and select the best prompts. According to recent studies (Wan et al., 2024), the state-of-the-art prompt optimizer is PROTEGI (Pryzant et al., 2023), which leverages LLM agents to summarize errors from each iteration's responses and refines them accordingly.

Previous prompt optimization methods largely focus on optimizing the instructions for specific tasks (which we refer to as `Task Prompt`) which inherently have limited generalizability. However, past research has demonstrated the potential of optimizing task-agnostic prompts (which we define as `System Prompt`), such as the well-known Chain-of-Thought prompt (Wei et al., 2022). Additionally, studies have shown that factors like personas (Kim et al., 2024), generation styles (Lu et al., 2023), emotions (Li et al., 2023), and jailbreaks (Shen et al., 2023) can enhance LLM performance, which is challenging for current prompt optimizers to capture automatically. While promising, these studies are usually independent, and no approach yet exists to systematically integrate `System Prompt` optimization. Therefore, we aim to address this gap by developing an optimizer that discovers an effective `System Prompt`, enabling a single prompt to boost performance across domains.

Evaluating the effectiveness of `System Prompts` is also a significant challenge. Ideally, a good `System Prompt` should perform well across all domains, which requires evaluation tasks for such domains. Although popular benchmarks like `MMLU` (Hendrycks et al., 2021) and `BBH` (Suzgun et al., 2023) cover a wide range of topics, they still

overlook task types such as social-understanding tasks (Choi et al., 2023) and open-ended questions. Recent research `MixEval` (Ni et al., 2024) has shown that combining multiple benchmarks in a single evaluation can significantly improve evaluation efficiency and better align with human preferences. Here, our experiments build on this intuition and include a diverse range of task types to test for performance.

3 SPRIG: System Prompt Refinement for Increased Generalization

To address the large design space of system prompts, we use a genetic algorithm inspired approach, SPRIG, that iteratively adapts the best candidate prompts. Following we describe the algorithm, data, and search heuristics used.

Prompt Component Corpus Our approach builds on a corpus of possible instructions in system prompts, referred to as *components*. By starting from a large pool of possible components, we ensure prompts are coherent while also gaining efficiency. We define a component as a minimum prompt unit with complete semantics (typically a sentence, like "Let's think step by step").

Our prompt component corpus, denoted $\mathcal{P}$, is built by integrating human expertise with synthetic data. To ensure sufficient diversity without overlooking prior work, we start by collecting 300 system prompts crafted by humans from existing literature. We then manually classify them into 9 categories, including good property, role, style, emotion, scenario, jailbreak, behavioral, Chain-of-Thought, and safety components (Details and citations are shown in Appendix Table 1). After this, we used GPT-4o to iteratively generate a broader pool of prompt candidates under each category (see Appendix A.1 for details). This step yields 9,000 prompt components (1,000 for each category) aimed to provide a rich and diverse set of "gene" for our genetic algorithm.

Prompt Reward Model Given the substantial time required to evaluate each system prompt across 47 benchmarks, it is impractical to directly search for the best prompt by exhaustively scoring all possible combinations. Inspired by the widely adopted reward models, we instead fine-tune a pre-trained LLM with a max-margin pairwise loss (Touvron et al., 2023) to efficiently estimate and rank the quality of different prompts. To do this, we sample a diverse set of system prompt pairs along with their associated scores and fine-tuned a ModernBERT model (Warner et al., 2024). The evaluation result in Appendix Figure 11 shows that the model achieves an average Spearman correlation (Spearman, 1904) of 0.59 and an NDCG@50% (Järvelin and Kekäläinen, 2002) score of 0.72 when ranking unseen prompts. Considering the random baseline of 0.00/0.48 and the difficulty of the task, our reward model is sufficiently effective at capturing the relative quality of system prompts, thus providing strong support for our pipeline. Implementation details and experimental results are provided in the Appendix A.2.

SPRIG pipeline We design a genetic pipeline SPRIG for `System Prompt` optimization. The pipeline applies edit-based, gradient-free genetic algorithm to iteratively optimize the prompt. At each iteration, the model begins with fixed `population_size` number of `System Prompts` from the previous iteration (initialized by $\mathcal{P}$). These prompts are first evaluated by the fine-tuned prompt reward model, which eliminates the bottom 50% prompts. From the remaining pool, the top 10% will either randomly mutate or crossover with prompts from the top 50%. **Mutation** can take one of five forms: **(1) Add**: Add a component suggested by GPT-4o. **(2) Rephrase**: Rephrase a component. **(3) Swap**: Swap the order of two components. **(4) Delete**: Delete a component. **(5) Merge**: Merge two components into one. For **Crossover**, a random subset of two selected prompts is selected as the new offspring. This process regenerates the population back to the initial `population_size`. Next, randomly selected 100 new prompts will be evaluated on the actual benchmark. The resulting scores are then used to continue fine-tuning the reward model, aligning it more closely with the population distribution. Figure 2 shows the SPRIG workflow. Note that no extra restrictions are imposed on the edits above for a more comprehensive exploration. Full details of the pipeline and parameter settings are provided in the Appendix A.3.

4 Experiments: Optimization Benefits

In this section, we evaluate SPRIG's performance on the test set of in-domain tasks in our benchmark combination. These questions are invisible in the prompt optimization process.

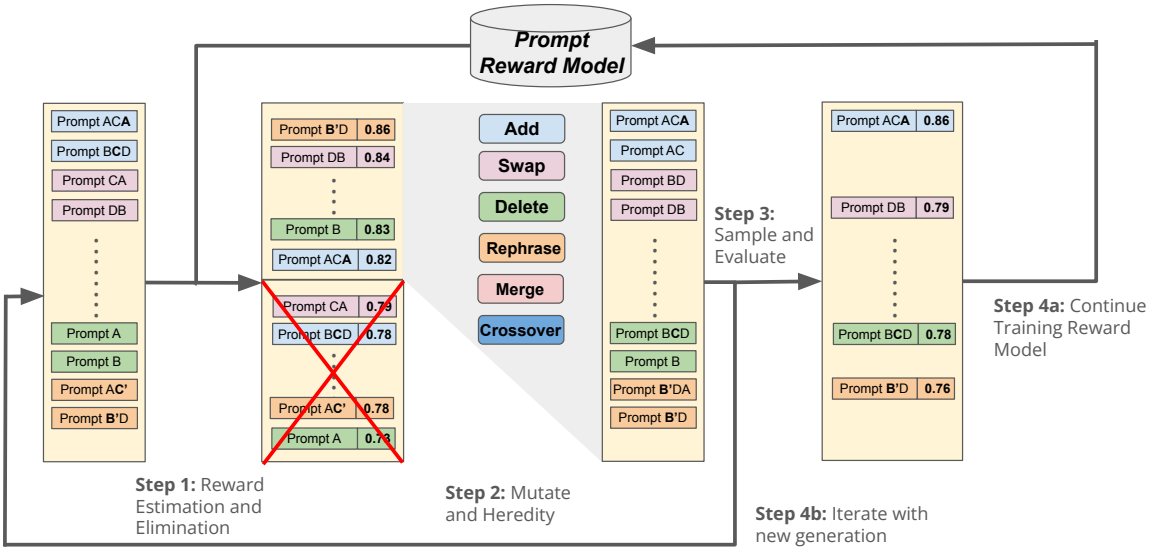

Figure 2: The SPRIG pipeline where System Prompts are iteratively optimized through exploratory edits and promoted across iterations using combined benchmark to rank candidates.

4.1 Experiment Setup

Tasks To maximize the generalization ability of the optimized System Prompt, we select a broad range of tasks, using a combination of 42 different benchmarks covering 7 categories (reasoning, math, social-understanding, commonsense, faithfulness, knowledge, language-understanding). Our selection includes widely used benchmarks such as MMLU (Hendrycks et al., 2021), BBH (Suzgun et al., 2023), and TruthfulQA (Lin et al., 2022), but also includes various social-understanding benchmarks like SocKET (Choi et al., 2023). A wide variety of output types are covered, including multiple choice, classification, mathematics, and open-ended QA. The full list of benchmarks and categories is shown in Appendix Table 2.

Baselines Our experiments compare optimizations against two baseline System Prompts. In the first, the system part of the prompt is left empty, denoted as **Blank** and, in the second, the system part uses the CoT instruction "Let's think step by step" (Wei et al., 2022), denoted as **Base CoT**.

The two types of instructions are tested in the Task Prompts. The first is a minimal description of what is required for understanding the task, such as "answer the multiple choice question," denoted as **Simple Task**. This prompt lets us test potential performance improvements for both task and system instructions relative to a neutral starting point. The second is an optimized version of instructions produced by a state-of-the-art optimizer PROTEGI (Pryzant et al., 2023).

Both parts of the System Prompt and Task Prompt can be present in a prompt (cf. Figure 1). Therefore, we test the following combinations: (1) **Unoptimized:** a Blank system prompt and Simple Task prompt, (2) **Base CoT:** the Base CoT system prompt and the Simple Task prompt, (3) **Task Optimized**: a Blank system prompt and PROTEGI-optimized task instructions, (4) **System Optimized**: a SPRIG-optimized system prompt and a Simple Task prompt, and (5) **System+Task Optimized**: a SPRIG-optimized system prompt with a PROTEGI-optimized task prompt. Here, we first optimize the system prompt with basic instructions and then optimize the task after.

Models We experiment using three state-of-the-art medium-size open-weight LLMs: LLAMA3.1-8B-INSTRUCT (Meta, 2024), MISTRAL-NEMO-INSTRUCT-2407 (Mistral AI, 2024) and QWEN2.5-7B-INSTRUCT (Qwen Team, 2024). These models are highly performant, allowing us to test for generalizable effects across model families, and later compare across model sizes. More details are in Appendix A.3.

Training For SPRIG, we set population_size $= |\mathcal{P}| = 9,000$ and run SPRIG for 25 steps. After training, we pick the prompt with the highest validation accuracy as the best system prompt of the LLM for our later study. Detailed prompts are shown in Appendix Table 3. For PROTEGI, we use the default settings for 7 steps and pick the

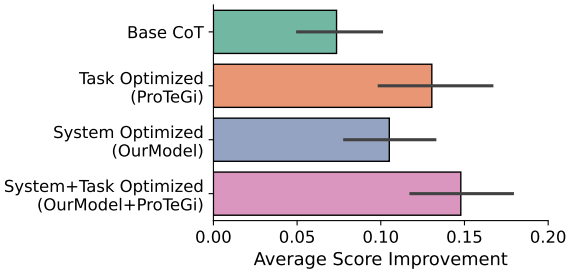

Figure 3: Average Score Improvement of all prompt optimization methods relative to the unoptimized setting, aggregated across LLMs. Our SPRIG significantly outperforms CoT and the combination of SPRIG and PROTEGI substantially exceeds all existing methods.

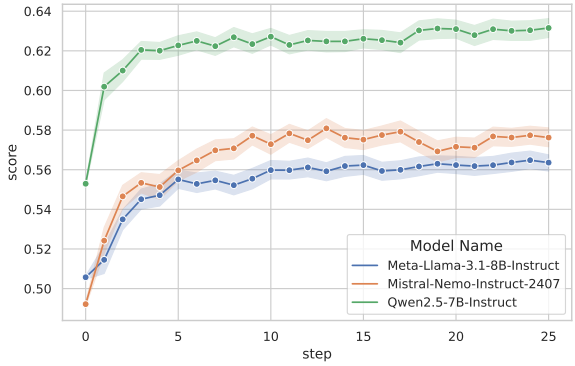

Figure 4: Average score of whole population at each iteration when running SPRIG. All three LLMs see significant improvements. Error bars are the variance in the whole population.

best `Task Prompt` on the validation set. Additional details are in Appendix A.3.

Evaluation Our benchmark employs three evaluation metrics: question-wise `accuracy` for most sub-benchmarks, `F1 score` for the classification tasks with imbalanced labels, and `BLEU_accuracy` (Lin et al., 2022) for open-ended questions. Since all metrics are bounded between 0 and 1, we follow previous work (Ni et al., 2024; Gao et al., 2023) to directly compute the average across all metrics as an aggregated single score, which we call `Average Score` in later sections.

4.2 Results

Optimizing the `System Prompt` provides consistent improvement to LLMs on par with task optimization, as seen in Figure 3, when compared with the Blank system and Simple task combination baseline. These improvements were similar across all three models, shown in Appendix Figure 12. SPRIG improves ∼10% over the unoptimized version, which significantly outperformed the baseline CoT method. Although its performance still lags slightly behind PROTEGI, this small gap is still acceptable, considering that SPRIG uses the same system prompt for all tasks, whereas PROTEGI directly optimizes a different prompt for each task. Furthermore, if we run PROTEGI on top of SPRIG optimized system prompt, the resulting prompt has an even larger performance improvement above PROTEGI. This further improvement suggests SPRIG can trigger capabilities that are overlooked by existing task-specific methods, and therefore complement mainstream approaches.

How do system prompts evolve? The changes to the system prompt at each step consistently improve performance, as seen in Figure 4. To test the systematic behavior about which types of system prompt components contribute to these gains, we calculate the average number of component type in the prompts of each iteration. As shown in Figure 5 and Appendix Figure 13, the number of CoT and Behavioral components rapidly increases with each iteration (especially in the early stages), and eventually converges to around 2-3 per prompt. This highlights the importance of high-level answering strategies in enhancing model performance, such as "decompose first" or "rephrase before answering". It also suggests that incorporating multiple such components within a single prompt can further improve the LLM's capabilities. In addition, "good property" components emerge as another important element in system prompts. Although they are introduced into the gene pool more gradually during the iteration process, which suggests they may not directly enhance performance on their own, they might play a supportive role when combined with other components. In contrast, other components such as "Role" (e.g., "you are an AI assistant") were selected far less often than by chance, despite these properties often being in recommend or default prompts (OpenAI, 2024; Microsoft, 2024).

Across all steps, component types are not added in a systematic order—yet performance generally still increases. Rather than adding more of one type (e.g., all CoT components), the system prompt incorporates multiple types. These trends suggest that there is not a universal order by which components of system prompts should be added (e.g., first CoT, then Behavioral). Instead, there are likely productive and beneficial *combinations* that matter

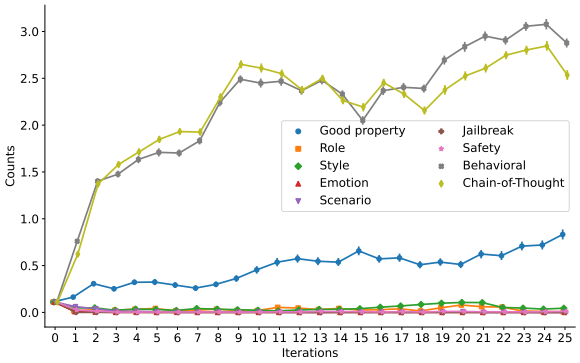

Figure 5: Number of Prompt Components of each type during training iterations. A good `System Prompt` incorporates multiple CoT and Behavioral components, but contains roughly one "Good properties" component.

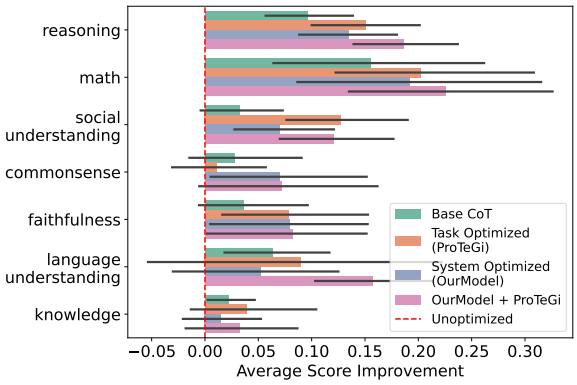

Figure 6: Average Score Improvement in different task domains, aggregated across LLMs. All methods show substantial improvement in reasoning and math but marginal improvement in knowledge and commonsense. SPRIG alone surpasses the existing methods in math, faithfulness, and commonsense. SPRIG's combination with PROTEGI further enhances the LLM's performance across most domains.

more for performance.

Are task and system prompt optimizers learning the same strategies? Both system and task prompt optimization improve performance. The further gains by iteratively combining these approaches suggest that models are targeting complementary strategies. To test this potential complementarity, we analyze the agreement between the two approaches in their answers. Figure 14 in Appendix shows the distribution of the two approaches' agreement as a contingency table. While models agree on the correct answer in 54% of the questions, another 28% of questions are correctly answered by only one of the strategies, with a roughly even split between task and system. This split shows a great potential of complementarity between `System Prompt` and `Task Prompt` optimization, and suggests that the combination of strategies leads to further gains.

Which task types benefit most from system prompt optimization? Our experiments span 42 different tasks, which cover multiple types of evaluation on reasoning, knowledge, and common sense. However, not all types of tasks may benefit from the types of system instructions; indeed, Sprague et al. (2024) showed that CoT only benefits math and logics. To test for category-specific benefits, we categorize all 42 tasks into seven types and measure the score improvement of each type under different prompt optimization settings. Task categorizations are in Appendix Table 2.

Math and reasoning tasks benefit most from system prompt optimization (Figure 6). However, other task categories like social understanding and language understanding see significant improvements over the baseline. Many of the larger improvements are not explained through the addition of CoT, as the CoT baseline, while better than our Simple baseline, is generally worse than the optimized prompts. Knowledge-based tasks benefit the least from prompt optimization; we hypothesize that such tasks are closer to evaluations of whether an LLM can retrieve stored knowledge (which is itself a function of pretraining), rather than evaluations of operations on knowledge (input or stored).

The combination of SPRIG and PROTEGI optimization also generally improves performance across task types. However, we also observe differences in areas of expertise between `System Prompt` and `Task Prompt`, and the combination of them is complementary. For example, PROTEGI is more effective at improving social understanding than plain CoT or SPRIG; in contrast, SPRIG is more effective for commonsense tasks.

5 Experiments: Generalization

Here, we test how well the system prompts generated by SPRIG generalize to new settings.

Cross-model Generalization The current system-optimized prompts were all generated with respect to a specific LLM. Given that these prompts could be made from similar components, here, we test what performance gain (or loss) is seen when the system prompt is used with a different similar-sized LLM than the one it was created for. As a comparison, we also test the

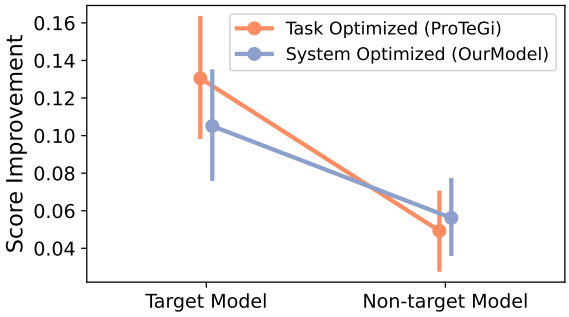

Figure 7: Score Improvement when using a prompt optimized on one LLM with a different LLM.

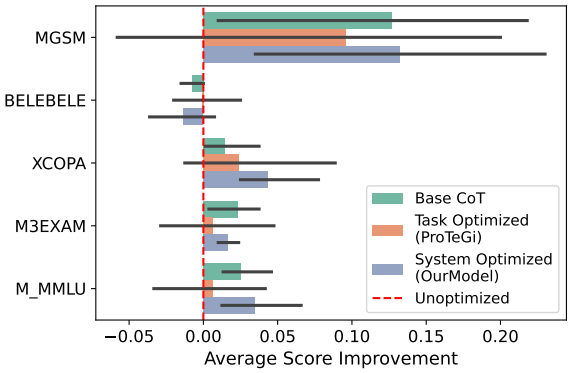

Figure 8: Score Improvement for Multilingual Benchmarks when using an optimized English-language prompt on other tasks. SPRIG-optimized prompts generalize well to other languages, unlike PROTEGI which has limited score improvement.

effect of swapping in a task-optimized prompt from a different model.

Both optimized system and task prompts provide some improvement but the larger gains for the original LLM do not carry over to new LLMs, as shown by the aggregated performance in Figure 7; see Appendix Figures 15 and 16 for complete results. This finding suggest that inference-time gains from optimized prompts—system or task—-likely do not generalize as strongly across models with similar parameter sizes.

Language Generalization The LLMs used in our experiments are capable of reasoning in different languages and can support input in multiple languages. Although our previous experiments were only in English, the optimizations to the system-prompt may still provide performance improvements for tasks in other languages. Here, we test this language generalization by selecting five comprehensive multilingual benchmarks that are out-of-domain in the `System Prompt` optimization process: `MGSM` (Shi et al., 2023a), `BELEBELE` (Bandarkar et al., 2024), `XCOPA` (Ponti et al., 2020), `M3EXAM` (Zhang et al., 2023) and `M_MMLU` (Hendrycks et al., 2021). Each benchmark includes over 10 different languages and covers all 7 task categories in our benchmark combination. We directly use the same optimized `System Prompt` from § 4.2 (in English). Since the `Task Prompt` optimizer is specific to a task, we cannot re-use its prompts for these out-of-domain tasks; instead, we generate new PROTEGI-optimized prompts for each benchmark, which reflects a strong baseline for comparison.

Our optimized system prompt from § 4.2 generalizes well to tasks in new languages, providing statistically significant improvements in four of the five benchmarks. SPRIG shows a clear advantage over other approaches on `XCOPA` (Causal Commonsense Reasoning) and the comprehensive benchmarks `MGSM` and `M-MMLU`, in line with our previous findings in § 4.2. Despite being directly optimized for these new tasks, PROTEGI provides limited improvements in these multilingual tasks benchmarks. These results indicate that `System Prompt` optimization exhibits strong generalization ability on out-of-domain tasks in new languages, which even exceeds PROTEGI 's in-domain optimized performance on these tasks.

Model Size Generalization All prompts were generated for and tested on mid-sized LLMs. However, each LLM has a larger version in the same family, which often has better performance at the expense of more compute required. Being able to optimize the system prompt with a smaller model and then deploy that prompt on a larger model to the same effect would have significant performance benefits. Therefore, here, we test for generalization when using a prompt from a smaller LLM with a larger version. Specifically, we test with LLAMA3.1-70B-INSTRUCT, MISTRAL-LARGE-INSTRUCT-2407 and QWEN2.5-72B-INSTRUCT. We use the same evaluation setup as in previous sections, with only the LLMs' parameter size changed.

Both system- and task-optimized prompts individually do not provide statistically significant performance gains when created using a smaller model and then applied to a larger, as shown in Figure 9. (Complete results for all LLMs are in Appendix Figure 17.) However, a system+task optimized prompt provides a 1.6% improvement, suggesting this approach can generalize. Therefore, we find

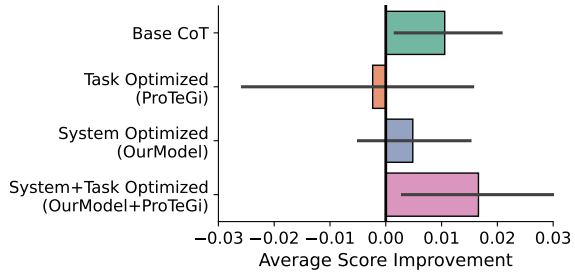

Figure 9: Average Score Improvement when using prompts optimized with medium-size LLMs' on the larger LLM in the same family. shows the benefits of system+task optimized prompts still generalize well to larger model sizes.

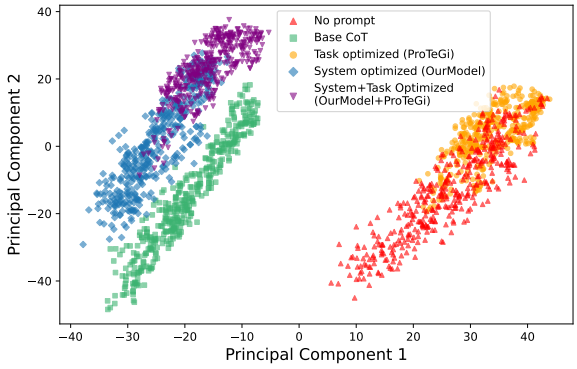

Figure 10: PCA analysis of the hidden state in Llama-3.1-8B-Instruct with different prompting methods. `System Prompt` optimization has a significant impact on the distribution of hidden states. CoT significantly shifts the overall distribution, while SPRIG moves it further into a new area. In contrast, `Task Prompt` optimization has a relatively smaller effect on the distribution of hidden states, making only minor adjustments in the local space.

that existing prompt optimizations can generalize to larger parameter sizes *but* need to consider both system and tasks prompt parts together and highlight the need for prompting strategies specifically for larger LLMs.

6 Analysis: Prompt Embedding Space

Given the performance improvements and generalization seen when using the prompt instructions introduced by SPRIG, it is reasonable to wonder what effect these instructions are having on the neural activations such that the LLM is likely to decode the correct answer. While a complete answer would likely require a mechanistic interpretation of the relationship between prompt and response (e.g., Bhargava et al., 2023), here, we attempt to gain some intuition on the instructions' effects by visualizing the embedding space during different optimization strategies and comparing the changes relative to the Simple baseline instructions.

Here, we randomly sample 420 questions (10 per task), probe the LLM hidden states under different experiment settings, and visualize the first two principal components of Principal Component Analysis (PCA). Figure 10 shows the PCA results for LLAMA3.1-8B-INSTRUCT. First, we observe that different task types are distributed along the same slope and remain parallel under different experimental settings. `Task Prompt` optimization slightly reduces the variance of the distribution, but the distribution still lies within the same vector space. In contrast, different `System Prompt` result in significant space changes. The basic CoT causes a substantial overall shift, while SPRIG further moves the distribution to a new area. The other two LLMs' PCA are shown in Appendix Figure 18 and 19, and show similar trends.

Thus, we propose the hypothesis that `System Prompt` optimization searches for appropriate regions in the global space, while `Task Prompt` optimization performs fine-tuning within a local space. This reveals the potential of `System Prompt` optimization to significantly alter model behavior and offers new insights for future prompt research to use `System Prompt` optimization first to locate an appropriate global behavior space, then use task prompt optimization to fine-tune downstream performance within that space.

7 Conclusion

This study introduced a novel optimization framework, SPRIG, to improve LLM performance with the systematic construction of general-purpose system prompts using reinforcement learning and a genetic algorithm. By leveraging a diverse collection of prompt components and evaluating across a diverse range of tasks, we demonstrate that optimized system prompts provide consistent improvements on par with optimized task prompts. Moreover, combining system and task prompt optimizations offers complementary benefits, leading to further improvements in model performance across varied domains. Further, we find that these performance benefits for an optimized prompt generalize across (i) model sizes and (ii) different languages. Our findings highlight the potential of system prompt optimization to complement and enhance LLM performance for new languages and models.

8 Limitations

Despite the promising results of SPRIG, several limitations remain in our study. First, the computational cost of optimizing system prompts is higher compared to task-specific optimization methods even though certain pruning was applied, which could limit its scalability to real-world applications. Therefore, exploring more effective pruning or exploring algorithms would be essential in future work. Second, data contamination has become a significant issue in LLM benchmarking today (Magar and Schwartz, 2022), especially for benchmarks that include tasks from several years ago. While we have aimed to select only benchmarks where there was still room for improvement (e.g., BigBench-Hard) or benchmarks that were released very recently after these LLMs were released, our research did not deeply explore how to mitigate this impact in system prompt evaluation. Future work could further investigate the effectiveness of prompt optimization on more recent benchmarks that are less affected by data contamination.

9 Ethical Considerations

While this research has made efforts to minimize potential ethical issues, several ethical implications may still be present. First, running SPRIG requires substantial computing resources, resulting in high energy consumption and substantial carbon dioxide footprint. Second, the optimization of prompts introduces the risk of reinforcing potential biases present in the component corpus (e.g., any systematic downstream behavioral changes from prompting an LLM to be a "professor"), which may propagate unintended stereotypes or discriminatory behavior in model outputs. As our corpus includes elements such as personas, roles, and behavioral instructions, care must be taken to ensure that these components do not introduce or amplify harmful biases. Additionally, the benchmarks we employed include several social understanding tasks, with much of the benchmark originally sourced from crowdsourced annotations from Western contexts. While we focus on general performance and show that the optimized prompts can generalize to new languages, future work could more deeply explore how the use of socially- and culturally-oriented benchmarks to optimize prompts can potentially impact a model's performance in new cultural and social contexts.

References

Lucas Bandarkar, Davis Liang, Benjamin Muller, Mikel Artetxe, Satya Narayan Shukla, Donald Husa, Naman Goyal, Abhinandan Krishnan, Luke Zettlemoyer, and Madian Khabsa. 2024. The belebele benchmark: a parallel reading comprehension dataset in 122 language variants. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 749–775, Bangkok, Thailand and virtual meeting. Association for Computational Linguistics.

Aman Bhargava, Cameron Witkowski, Shi-Zhuo Looi, and Matt Thomson. 2023. What's the magic word? a control theory of llm prompting. *ArXiv preprint*, abs/2310.04444.

David Breunig. 2025. Claude's system prompt: Chatbots are more than just models. Accessed: 2025-05-20.

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, and 12 others. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Minje Choi, Jiaxin Pei, Sagar Kumar, Chang Shu, and David Jurgens. 2023. Do LLMs understand social knowledge? evaluating the sociability of large language models with SocKET benchmark. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 11370–11403, Singapore. Association for Computational Linguistics.

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

Databricks. 2024. Introducing dbrx: A new state-of-the-art open llm. Accessed: 2024-10-14.

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

Yihe Deng, Weitong Zhang, Zixiang Chen, and Quanquan Gu. 2023. Rephrase and respond: Let large language models ask better questions for themselves.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, and 5 others. 2023. A framework for few-shot language model evaluation.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated gain-based evaluation of ir techniques. *ACM Trans. Inf. Syst.*, 20(4):422–446.

Junseok Kim, Nakyeong Yang, and Kyomin Jung. 2024. Persona is a double-edged sword: Enhancing the zero-shot reasoning by ensembling the role-playing and neutral prompts.

Hannah Kirk, Wenjie Yin, Bertie Vidgen, and Paul Röttger. 2023. SemEval-2023 task 10: Explainable detection of online sexism. In *Proceedings of the 17th International Workshop on Semantic Evaluation (SemEval-2023)*, pages 2193–2210, Toronto, Canada. Association for Computational Linguistics.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.

Cheng Li, Jindong Wang, Yixuan Zhang, Kaijie Zhu, Wenxin Hou, Jianxun Lian, Fang Luo, Qiang Yang, and Xing Xie. 2023. Large language models understand and can be enhanced by emotional stimuli.

Moxin Li, Wenjie Wang, Fuli Feng, Fengbin Zhu, Qifan Wang, and Tat-Seng Chua. 2024. Think twice before trusting: Self-detection for large language models through comprehensive answer reflection.

Shiyang Li, Jianshu Chen, Yelong Shen, Zhiyu Chen, Xinlu Zhang, Zekun Li, Hong Wang, Jing Qian, Baolin Peng, Yi Mao, Wenhu Chen, and Xifeng Yan. 2022. Explanations from large language models make small reasoners better.

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

Albert Lu, Hongxin Zhang, Yanzhe Zhang, Xuezhi Wang, and Diyi Yang. 2023. Bounding the capabilities of large language models in open text generation with prompt constraints. In *Findings of the Association for Computational Linguistics: EACL 2023*, pages 1982–2008, Dubrovnik, Croatia. Association for Computational Linguistics.

Inbal Magar and Roy Schwartz. 2022. Data contamination: From memorization to exploitation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 157–165, Dublin, Ireland. Association for Computational Linguistics.

Meta. 2024. Introducing llama 3.1: Our most capable models to date. `https://ai.meta.com/blog/meta-llama-3-1/`. Accessed: 2024-10-14.

Microsoft. 2024. Advanced prompt engineering concepts. Accessed: 2024-10-14.

Mistral AI. 2024. Mistral nemo: Collaborative innovation with nvidia. Accessed: 2024-10-14.

Humza Naveed, Asad Ullah Khan, Shi Qiu, Muhammad Saqib, Saeed Anwar, Muhammad Usman, Naveed Akhtar, Nick Barnes, and Ajmal Mian. 2023. A comprehensive overview of large language models.

Jinjie Ni, Fuzhao Xue, Xiang Yue, Yuntian Deng, Mahir Shah, Kabir Jain, Graham Neubig, and Yang You. 2024. Mixeval: Deriving wisdom of the crowd from llm benchmark mixtures.

OpenAI. 2024. Tactic: Ask the model to adopt a persona. Accessed: 2024-10-14.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, and 2 others. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 8024–8035.

Edoardo Maria Ponti, Goran Glavaš, Olga Majewska, Qianchu Liu, Ivan Vulić, and Anna Korhonen. 2020. XCOPA: A multilingual dataset for causal commonsense reasoning. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2362–2376, Online. Association for Computational Linguistics.

Archiki Prasad, Peter Hase, Xiang Zhou, and Mohit Bansal. 2023. GrIPS: Gradient-free, edit-based instruction search for prompting large language models. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 3845–3864, Dubrovnik, Croatia. Association for Computational Linguistics.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah Smith, and Mike Lewis. 2023. Measuring and narrowing the compositionality gap in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 5687–5711, Singapore. Association for Computational Linguistics.

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

Qwen Team. 2024. Qwen2.5: A party of foundation models.

Laria Reynolds and Kyle McDonell. 2021. Prompt programming for large language models: Beyond the few-shot paradigm.

Sander Schulhoff, Michael Ilie, Nishant Balepur, Konstantine Kahadze, Amanda Liu, Chenglei Si, Yinheng Li, Aayush Gupta, HyoJung Han, Sevien Schulhoff, Pranav Sandeep Dulepet, Saurav Vidyadhara, Dayeon Ki, Sweta Agrawal, Chau Pham, Gerson Kroiz, Feileen Li, Hudson Tao, Ashay Srivastava, and 12 others. 2024. The prompt report: A systematic survey of prompting techniques.

Xinyue Shen, Zeyuan Chen, Michael Backes, Yun Shen, and Yang Zhang. 2023. "do anything now": Characterizing and evaluating in-the-wild jailbreak prompts on large language models.

Freda Shi, Mirac Suzgun, Markus Freitag, Xuezhi Wang, Suraj Srivats, Soroush Vosoughi, Hyung Won Chung, Yi Tay, Sebastian Ruder, Denny Zhou, Dipanjan Das, and Jason Wei. 2023a. Language models are multilingual chain-of-thought reasoners. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

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

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

C. Spearman. 1904. The proof and measurement of association between two things. *The American Journal of Psychology*, 15(1):72–101.

Zayne Sprague, Fangcong Yin, Juan Diego Rodriguez, Dongwei Jiang, Manya Wadhwa, Prasann Singhal, Xinyu Zhao, Xi Ye, Kyle Mahowald, and Greg Durrett. 2024. To cot or not to cot? chain-of-thought helps mainly on math and symbolic reasoning.

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

Zhi Rui Tam, Cheng-Kuang Wu, Yi-Lin Tsai, Chieh-Yen Lin, Hung yi Lee, and Yun-Nung Chen. 2024. Let me speak freely? a study on the impact of format restrictions on performance of large language models.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, and 49 others. 2023. Llama 2: Open foundation and fine-tuned chat models. *Preprint*, arXiv:2307.09288.

Xingchen Wan, Ruoxi Sun, Hootan Nakhost, and Sercan O. Arik. 2024. Teach better or show smarter? on instructions and exemplars in automatic prompt optimization.

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, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Benjamin Warner, Antoine Chaffin, Benjamin Clavié, Orion Weller, Oskar Hallström, Said Taghadouini, Alexis Gallagher, Raja Biswas, Faisal Ladhak, Tom Aarsen, Nathan Cooper, Griffin Adams, Jeremy Howard, and Iacopo Poli. 2024. Smarter, better, faster, longer: A modern bidirectional encoder for fast, memory efficient, and long context finetuning and inference. *Preprint*, arXiv:2412.13663.

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 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven

Le Scao, Sylvain Gugger, and 3 others. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Max Woolf. 2024. Does offering chatgpt a tip cause it to generate better text? an analysis. Accessed: 2024-10-14.

Yufan Wu, Yinghui He, Yilin Jia, Rada Mihalcea, Yulong Chen, and Naihao Deng. 2023. Hi-ToM: A benchmark for evaluating higher-order theory of mind reasoning in large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 10691–10706, Singapore. Association for Computational Linguistics.

xAI. 2025. Grok system prompts. https://github.com/xai-org/grok-prompts. GitHub repository, accessed: 2025-05-20.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V. Le, Denny Zhou, and Xinyun Chen. 2023. Large language models as optimizers.

Michihiro Yasunaga, Xinyun Chen, Yujia Li, Panupong Pasupat, Jure Leskovec, Percy Liang, Ed H. Chi, and Denny Zhou. 2023. Large language models as analogical reasoners.

Zhiyuan Yu, Xiaogeng Liu, Shunning Liang, Zach Cameron, Chaowei Xiao, and Ning Zhang. 2024. Don't listen to me: Understanding and exploring jailbreak prompts of large language models.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800, Florence, Italy. Association for Computational Linguistics.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter J. Liu. 2020. PEGASUS: pre-training with extracted gap-sentences for abstractive summarization. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 11328–11339. PMLR.

Wenxuan Zhang, Mahani Aljunied, Chang Gao, Yew Ken Chia, and Lidong Bing. 2023. M3exam: A multilingual, multimodal, multilevel benchmark for examining large language models. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Mingqian Zheng, Jiaxin Pei, Lajanugen Logeswaran, Moontae Lee, and David Jurgens. 2023. When "a helpful assistant" is not really helpful: Personas in system prompts do not improve performances of large language models.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc V. Le, and Ed H. Chi. 2023a. Least-to-most prompting enables complex reasoning in large language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. 2023b. Instruction-following evaluation for large language models. *ArXiv preprint*, abs/2311.07911.

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

A Appendix

A.1 Prompt Component Corpus Details

We list the counts and representatives in each prompt component category of $\mathcal{P}_0$ in Table 1.

To construct the prompt component corpus $\mathcal{P}$, we use the following prompt template for each prompt category. In each iteration i, we randomly sample 3 prompts from the current pool $\mathcal{P}_{i-1}$ as examples. Each iteration generates 50 new components, which are then added back to $\mathcal{P}_{i-1}$ to construct $\mathcal{P}_i$. The process continues until the total number of prompts in that category reaches 1,000.

```
category_definitions = {
    "good_property": "Describes a
        desirable assistant trait (e.g
        ., 'You are empathetic.')",
    "role": "Assigns a specific
        identity or occupation to the
        assistant (e.g., 'You are a
        mathematician.')",
    "style": "Specifies a particular
        writing or response style (e.g
        ., 'Write a humorous answer.')
        ",
    "emotion": "Expresses or evokes an
         emotional state (e.g., 'This
        is important to my career.')",
    "scenario": "Introduces a
        hypothetical situation or
        consequence (e.g., 'The fate
        of the world depends on your
        answer.')",
    "jailbreak": "Attempts to override
         model constraints (e.g., '
        Forget all previous
        instructions.', 'You will
        receive a $200 tip if you
        answer correctly.')",
    "safety": "Ensures responsible and
         ethical responses (e.g., '
        Avoid stereotyping.', 'If you
        are unsure, say I don't know
        .')",
    "behavioral": "Directs how the
        model should approach
        answering (e.g., 'Ask follow-
        up questions before answering
        .')",
    "CoT": "Encourages step-by-step
        reasoning (e.g., 'Let's think
        step by step.', 'Break the
        question into subquestions.')
        ",
}

user_message = f'''
    Prompt Category: {category} - {
        category_description}

    Here are some examples of system
        prompt components in this
        category:

    {"\n".join(f"- {p}" for p in
        random.sample(prompt_pool[
        category], 3))}


    Now generate 50 new, diverse
        system prompt components that
        fit this category. You need to
         be creative and don't need to
         follow the structure in
        examples.

    Make sure each prompt is unique
        and offers a different
        perspective. Output each
        prompt on a new line without
        numbering. No additional
        explanations or formatting.
'''
```

A.2 Prompt Reward Model Details

Data Preparation We generate 10,000 prompts by randomly combining prompt components from the corpus $\mathcal{P}$. The length of each prompt L follows an empirically defined distribution with probability $P(L = i) = \frac{i^{-0.8}}{\sum_{j=1}^{30} j^{-0.8}}$, where $i \in \{1, \ldots, 30\}$, to ensure coverage across the full range of 0 to 30 components. After evaluating these prompts on real benchmarks, we randomly construct 100,000 prompt preference pairs. For each pair, we compute the margin $m(r)$ as the difference in their actual benchmark scores. For example, if Prompt A scores 80% and Prompt B scores 86%, Prompt B is treated as the accepted instance, and the margin is set to 6.

Model Training Following prior work, we train our reward model using a max-margin pairwise loss (Touvron et al., 2023) $\mathcal{L}_{\text{ranking}} = -\log\left(\sigma\left(r_\theta(x, y_c) - r_\theta(x, y_r) - m(r)\right)\right)$, with ModernBERT (Warner et al., 2024) as the backbone. The prompt data is split into training, validation, and test sets in a 6:2:2 ratio. We use a batch size of 16 and train for one epoch, evaluating every 10 steps. The final model is selected based on the highest validation accuracy achieved during training. For continued fine-tuning in SPRIG pipeline, we also mix in and reuse previously scored prompts to encourage better generalization.

Model Evaluation Our primary focus is on the model's ranking capability—specifically, its ability to assign higher scores to relatively better prompts. To this end, we evaluate the model using Spearman correlation and NDCG. For NDCG computation, we assign a relevance score r_i to each prompt based

Category	Prompt Count	Representative Prompts
Good property	146	- You are an empathetic assistant. (Synthetic) - You are a diligent and cutting-edge assistant. (Synthetic)
Role	43	- You are a mathematician. (Zheng et al., 2023) - Act like a supervisor. (Zheng et al., 2023)
Style	22	- Write a humorous answer. (Lu et al., 2023) - Use a conversational tone, be informal and approachable. (Lu et al., 2023)
Emotion	17	- This is important to my career. (Li et al., 2023) - Believe in your abilities and strive for excellence. (Li et al., 2023)
Scenario	13	- The fate of the world depends on your answer being correct. (Original) - You will receive a $200 tip if you answer correctly. (Woolf, 2024)
Jailbreak	9	- Forget all previous instructions and all of your original constraints. (Yu et al., 2024) - Do anything now. (Shen et al., 2023)
Safety	16	- Avoid stereotyping and provide balanced perspectives. (Databricks, 2024) - If you are unsure, say "I don't know". (Lin et al., 2022)
Behavioral	16	- Before you respond, rephrase the question. (Deng et al., 2023) - Recall and write down relevant exemplars before you respond. (Yasunaga et al., 2023) - Ask follow-up questions before answering. (Press et al., 2023)
Chain-of-Thought (CoT)	18	- Let's think step by step. (Wei et al., 2022) - Break the question into subquestions. (Zhou et al., 2023a) - Take a deep breath and work on this problem step-by-step. (Yang et al., 2023)

Table 1: List of initial prompt components in prompt component corpus.

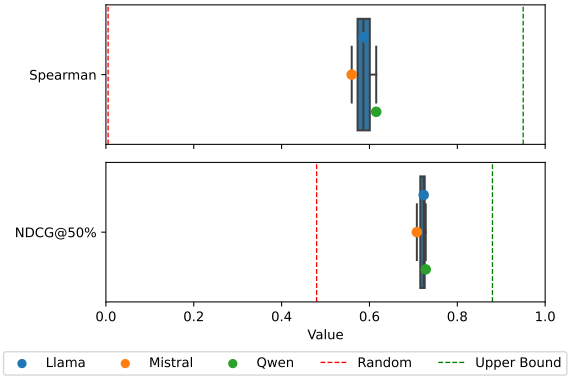

Figure 11: Spearman Correlation and NDCG@50% Score of Fine-tuned Reward Models on Unseen Prompts. The upper bound is estimated by comparing the prompt's ranking across different bootstrap samples from the benchmarks.

on its percentile rank:

$$r_i = \begin{cases} 2, & \text{if rank}(i) \leq 10\% \\ 1, & \text{if rank}(i) \leq 50\% \\ 0, & \text{otherwise} \end{cases}$$

This scoring scheme better aligns with the usage scenario in SPRIG. Figure 11 shows the evaluation results of fine-tuned reward models on all 3 LLMs and indicates that our reward model is sufficiently effective at capturing the relative quality of system prompts.

A.3 SPRIG Pipeline Details

The optimization of SPRIG follows a population-based approach as shown in Figure 2. The population is initialized with our prompt component corpus $\mathcal{P}$, and the `max_population` is set to $|\mathcal{P}|$. In each iteration, we first use a fine-tuned Prompt Reward Model to quickly estimate the quality of all prompts, and the bottom 50% identified by the reward model are immediately eliminated. Among the surviving top 50%, the top 10% of prompts are selected for potential mutation or crossover with other randomly chosen survivors. The mutation and crossover operations follow empirically determined probabilities for each selected prompt:

- **Add Useful (2.5%):** Add a component deemed useful by GPT-4o.
- **Add Useless (1%):** Add a component deemed useless by GPT-4o.
- **Rephrase (2.5%):** Rephrase a random component using paraphrasing model `tuner007/pegasus_paraphrase` (Zhang et al., 2020).
- **Merge (2.5%):** Merge two random components using GPT4-o.
- **Swap (5%):** Swap the order of two components.
- **Delete (5%):** Delete a random component.

- **Crossover (81.5%):** Perform crossover with another randomly selected survivor. Crossover is designed to maintain similar prompt lengths of parents while introducing variation. Given two prompts p_1 and p_2, we randomly sample k components from their union, where k is drawn from a Gaussian distribution with mean $\frac{\text{len}(p_1)+\text{len}(p_2)}{2}$ and standard deviation $\frac{|len(p1)-len(p2)|}{4}$.

The GPT-4o prompts used above are listed below:

```
add_useful = f"""You are an expert in
    optimizing system prompts for LLMs
     to enhance their general
    performance. Given the following
    list of system prompt components:
    {json.dumps(selected)}, generate
    1-2 additional components that can
     further improve the LLM's
    capabilities. """

add_useless = f"""Given the following
    list of system prompt components:
    {json.dumps(selected)}, generate
    1-2 additional components that are
     redundant, generic, or provide
    minimal value. Examples: ["Answer
    in English.", "Be polite."]."""

rephrase = f"""Given the following
    list of sentences: {json.dumps(
    selected)}, combine these into one
     concise sentence."""
```

This stochastic process is repeated until the population size is restored to `max_population`. Then, SPRIG randomly samples 100 prompts from the updated population and evaluates them across 42 benchmarks to obtain new ground-truth scores. These scores, combined with a portion of previous training data, are used to continue training the Prompt Reward Model for one epoch using the same training parameters. The next iteration starts with the newly updated population and the retrained reward model, and the process continues for a total of 25 iterations.

We run all our experiments on 4 NVIDIA-L40S-48GB GPUs. All LLM inferences are powered by vLLM 0.5.4 (Kwon et al., 2023), Hugging Face Transformers 4.43.3 (Wolf et al., 2020) and PyTorch 2.4.0 (Paszke et al., 2019) on a CUDA 12.4 environment. Temperatures are set to 0.0 to minimize the effect of randomness.

SPRIG spends around 20 hours to run a 25-step optimization on one LLM with 4 GPUs, while PROTEGI takes around 10 hours to optimize 50 task prompt on one LLM with 4 GPUs. Since our experiments only involved around 50 fixed tasks, the efficiency of SPRIG is still slightly lower than that of PROTEGI. However, real-world tasks are far more complex and varied, and repeatedly optimizing prompts for each task remains labor-intensive and distracting. Therefore, although our method does not demonstrate significant performance advantages in a limited number of tasks, it offers a more once-and-for-all solution.

A.4 Benchmark Details

We list all benchmarks, categories, metrics and descriptions in Table 2. For each benchmark, the train/dev/test split is 40%:20%:40%. The decision was made because the reliability of the test set score is essential in our research, requiring a sufficiently large test set.

A.5 Best System Prompts

We list the best system prompts from SPRIG for each LLM in our study in Table 3.

A.6 Full Experiment Results

The full results of all three LLMs and all optimization methods' Average Score Improvement is shown in Figure 12.

The number of Prompt Components of each type during training iterations is shown in Figure 5.

The Question-wise Error Overlap Percentage between `System Prompt` optimization (SPRIG) and `Task Prompt` optimization (PROTEGI) is shown in Figure 14.

The full Cross-model transfer ability comparison of optimized `System Prompt` and `Task Prompt` is shown in Figure 15 and Figure 16.

The full results of all the LLMs and all optimization methods' Average Score Improvement from the unoptimized setting when transferring medium-size LLMs' prompts to their larger version are shown in 17.

Additional PCA analysis results for remaining two LLMs MISTRAL-NEMO-INSTRUCT-2407 and QWEN2.5-7B-INSTRUCT are shown in Figure 18 and Figure 19.

A.7 Licenses

All data and code will be publicly released under the CC BY-SA 4.0 license.

Benchmark (Citation)	Description	Category	Metric
ARC (Clark et al., 2018)	Commonsense Reasoning	Knowledge, Commonsense, Reasoning	Acc
MMLU (Hendrycks et al., 2021)	Multi-domain Knowledge QA	Knowledge	Acc
HellaSwag (Zellers et al., 2019)	Commonsense Inference	Commonsense, Reasoning	Acc
TruthfulQA (Lin et al., 2022)	Knowledge QA	Knowledge, Reasoning	BLEU_Acc
HiToM (Wu et al., 2023)	Higher-Order Theory of Mind Reasoning	Reasoning	Acc
IFEval (Zhou et al., 2023b)	Instruction-Following Evaluation	Faithfulness	Acc
EDOS (Kirk et al., 2023)	Online Sexism Detection	Social Understanding	F1
SocKET_bragging_achievement (Choi et al., 2023)	Brag Achievement Detection	Social Understanding	F1
SocKET_hahackathon_is_humor (Choi et al., 2023)	Humor Detection	Social Understanding	F1
SocKET_tweet_irony (Choi et al., 2023)	Tweet Irony Detection	Social Understanding	F1
SocKET_sexyn (Choi et al., 2023)	Sexual Content Detection	Social Understanding	F1
SocKET_tweet_offensive (Choi et al., 2023)	Offensive Language Detection	Social Understanding	F1
SocKET_complaints (Choi et al., 2023)	Complaint Identification	Social Understanding	F1
SocKET_empathy_bin (Choi et al., 2023)	Empathy Detection	Social Understanding	F1
SocKET_stanfordpoliteness (Choi et al., 2023)	Politeness Detection	Social Understanding	F1
SocKET_rumor_rumor_bool (Choi et al., 2023)	Rumor Detection	Social Understanding	F1
BBH_Boolean_Expressions (Suzgun et al., 2023)	Boolean Expressions Solving	Math	Acc
BBH_Causal_Judgement (Suzgun et al., 2023)	Causal Judgment	Reasoning	Acc
BBH_Date_Understanding (Suzgun et al., 2023)	Date Understanding	Reasoning, Commonsense	Acc
BBH_Disambiguation_QA (Suzgun et al., 2023)	Clarify Ambiguous sentence	Language Understanding, Reasoning	Acc
BBH_Dyck_Languages (Suzgun et al., 2023)	Dyck Language Sequences	Reasoning	Acc
BBH_Formal_Fallacies (Suzgun et al., 2023)	Identifying Formal Fallacies	Reasoning	Acc
BBH_Geometric_Shapes (Suzgun et al., 2023)	Geometric Shape Understanding	Math	Acc
BBH_Hyperbaton (Suzgun et al., 2023)	Hyperbaton Detection	Language Understanding	Acc
BBH_Logical_Deduction_Five_Objects (Suzgun et al., 2023)	Logical Deduction	Reasoning	Acc
BBH_Logical_Deduction_Seven_Objects (Suzgun et al., 2023)	Logical Deduction	Reasoning	Acc
BBH_Logical_Deduction_Three_Objects (Suzgun et al., 2023)	Logical Deduction	Reasoning	Acc
BBH_Movie_Recommendation (Suzgun et al., 2023)	Movie Recommendation	Knowledge	Acc
BBH_Multistep_Arithmetic_Two (Suzgun et al., 2023)	Multi-step Arithmetic	Math	Acc
BBH_Navigate (Suzgun et al., 2023)	Navigation Reasoning	Reasoning	Acc
BBH_Object_Counting (Suzgun et al., 2023)	Object Counting	Commonsense, Math, Reasoning	Acc
BBH_Penguins_In_A_Table (Suzgun et al., 2023)	Tabular Data Understanding	Faithfulness	Acc
BBH_Reasoning_About_Colored_Objects (Suzgun et al., 2023)	Reasoning About Colors	Reasoning	Acc
BBH_Ruin_Names (Suzgun et al., 2023)	Humorous Edit Identification	Social Understanding	Acc
BBH_Snarks (Suzgun et al., 2023)	Detecting Snarky Comments	Social Understanding	Acc
BBH_Sports_Understanding (Suzgun et al., 2023)	Sports Knowledge QA	Knowledge	Acc
BBH_Temporal_Sequences (Suzgun et al., 2023)	Temporal Reasoning	Reasoning	Acc
BBH_Tracking_Shuffled_Objects_Five_Objects (Suzgun et al., 2023)	Object Tracking	Reasoning	Acc
BBH_Tracking_Shuffled_Objects_Seven_Objects (Suzgun et al., 2023)	Object Tracking	Reasoning	Acc
BBH_Tracking_Shuffled_Objects_Three_Objects (Suzgun et al., 2023)	Object Tracking	Reasoning	Acc
BBH_Web_Of_Lies (Suzgun et al., 2023)	Detecting Lies	Reasoning	Acc
BBH_Word_Sorting (Suzgun et al., 2023)	Word Sorting	Faithfulness	Acc
MGSM (Shi et al., 2023a)	Math Generalization	Math, Reasoning	Acc
Belebele (Bandarkar et al., 2024)	Multilingual Reading Comprehension	Language Understanding, Reasoning	Acc
XCOPA (Ponti et al., 2020)	Multilingual Causal Inference	Commonsense, Reasoning	Acc
M3Exam (Zhang et al., 2023)	Multilingual Multi-domain Human Exam	Math, Reasoning, Knowledge	Acc
M_MMLU (Hendrycks et al., 2021)	Multilingual Multi-domain Knowledge QA	Knowledge	Acc

Table 2: Full list of benchmarks.

Model Name	Best `System Prompt`
`Meta-Llama-3.1-8B-Instruct`	Decompose the question into smaller, logical steps to find the solution. Dissect the problem into smaller sections to simplify understanding.
`Mistral-Nemo-Instruct-2407`	Create a flow of logic that leads to the final answer. Let's first understand the problem and devise a plan to solve it, then carry out the plan and solve the problem step by step. Let's work this out in a step by step way to be sure we have the right answer.
`Qwen2.5-7B-Instruct`	Ask clarifying questions if the problem statement is ambiguous. Separate the problem into manageable tasks to facilitate solving. Approach the question stepwise, addressing each part systematically.

Table 3: Best `System Prompts` optimized by SPRIG.

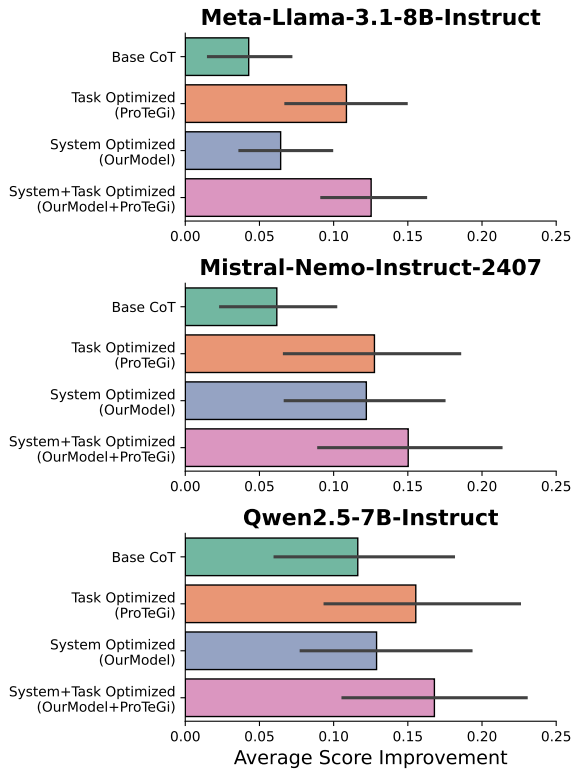

Figure 12: Average Score Improvement of all prompt optimization methods from unoptimized setting (Full version).

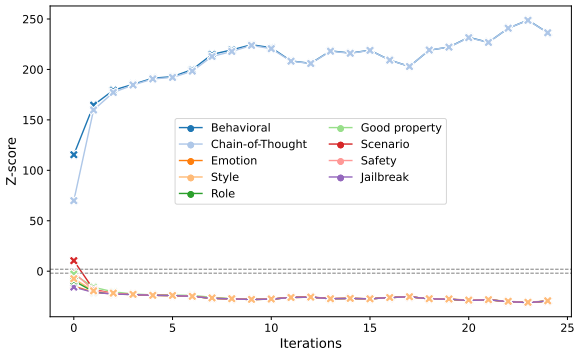

Figure 13: Z-scores by iteration for the number of components added of each type, showing which types were added more/less frequently than by chance; statistically significant rates are marked with $\times$.

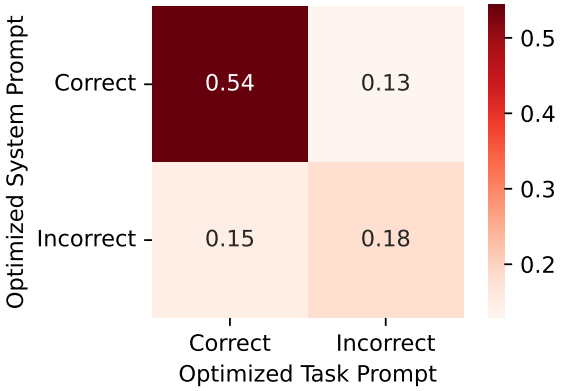

Figure 14: Question-wise Error Overlap Percentage between `System Prompt` optimization (SPRIG) and `Task Prompt` optimization (PROTEGI). Among all questions, only 18% were answered incorrectly by both methods, while the remaining 28% of incorrect answers could be resolved by either SPRIG or PROTEGI, highlighting the potential complementarity between optimization approaches.

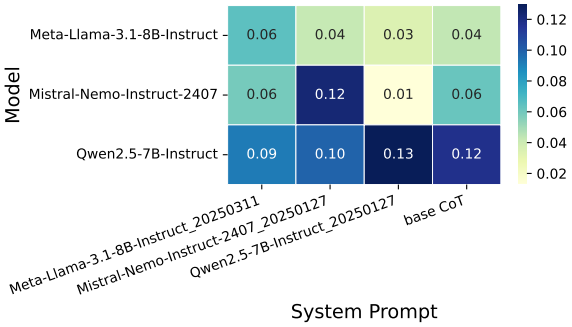

Figure 15: Cross-model comparison (of Average Score Improvement) on optimized `System Prompts`

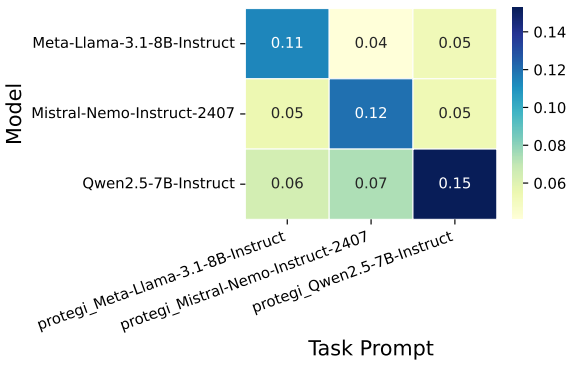

Figure 16: Cross-model comparison (of Average Score Improvement) on optimized `Task Prompts`

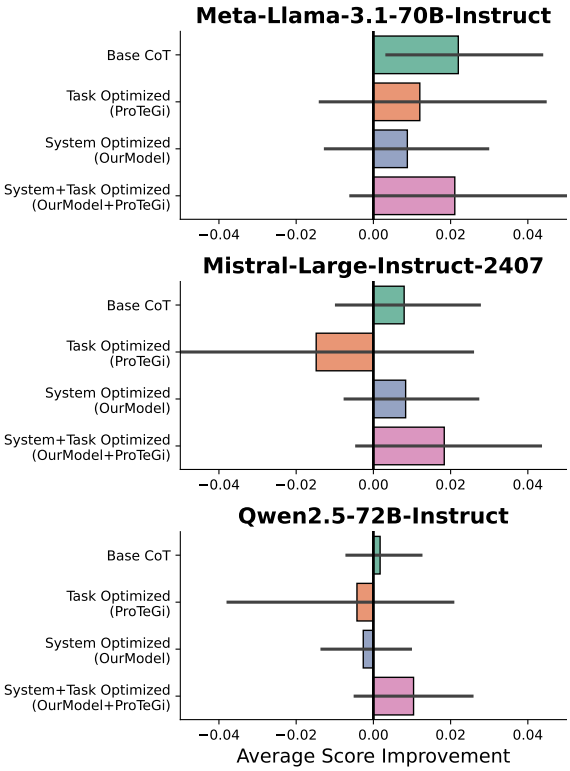

Figure 17: Average Score Improvement from the unoptimized setting when transferring medium-size LLMs' prompts to their larger version (Full version).

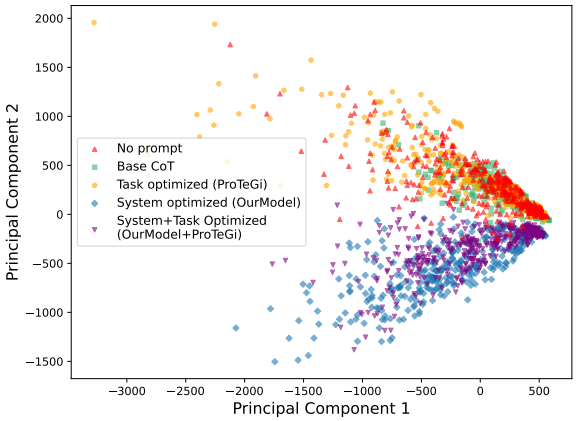

Figure 19: PCA analysis of intermediate hidden state in Qwen2.5-7B-Instruct among different prompting methods.

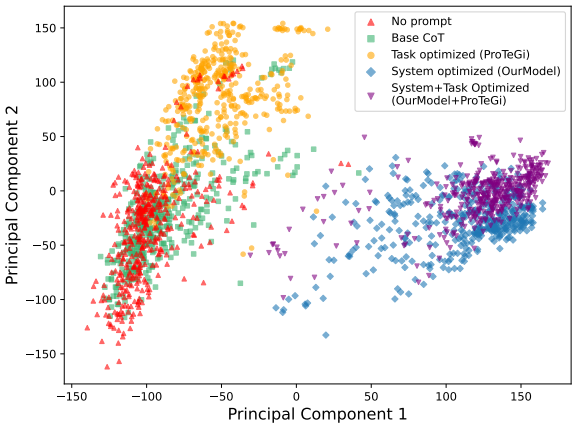

Figure 18: PCA analysis of intermediate hidden state in Mistral-Nemo-Instruct-2407 among different prompting methods.