# Learning from Contrastive Prompts: An Automated Prompt Optimization Framework

Anonymous ACL submission

## Abstract

As LLMs evolve, significant effort is spent on manually crafting prompts to unlock their full potential. While existing prompt optimization methods automate this process, they often underperform due to their reliance on learning exclusively from incorrect samples. We propose the Learning from Contrastive Prompts (LCP) framework, a novel approach that leverages contrastive learning to generate more effective prompts. Unlike previous methods, LCP analyzes the distinctive patterns between high-performing and low-performing prompts, extracting crucial insights about what makes a prompt successful. This contrastive mechanism enables the framework to identify subtle prompt characteristics that significantly impact model performance. Our evaluation on the Big-Bench Hard dataset shows that LCP has a win rate of over 87% over existing methods in prompt optimization. LCP offers a systematic approach to prompt engineering, reducing manual effort in deploying LLMs across varied contexts.

## 1 Introduction

Prompt engineering—the process of crafting effective instructions for Large Language Models (LLMs)—remains a critical but largely empirical practice. Despite its importance in optimizing model performance, the process relies heavily on trial-and-error experimentation. This challenge is amplified by LLMs' high sensitivity to prompt phrasing; for example, the simple addition of "Let's think step-by-step" in zero-shot chain-of-thought prompting can dramatically improve reasoning capabilities (Kojima et al., 2022). However, even minor variations in semantically similar prompts can lead to significant performance differences (Zhou et al., 2023; Salinas and Morstatter, 2024), making the search for optimal prompts labor-intensive and time-consuming.

Recent research ((Yang et al., 2024; Guo et al., 2024; Wang et al., 2024; Zhou et al., 2023; Sun et al., 2023)) has attempted to automate prompt optimization with varying success. Approaches such as AutoHint (Sun et al., 2023) use LLMs to generate hints from incorrect samples, though this risks over-specializing prompts to specific error cases. Similarly, OPRO (Yang et al., 2024) employs LLMs as optimizers to iteratively generate prompts based on ranking previous attempts, but fails to adequately incorporate feedback from incorrect responses. These limitations in existing methods highlight the need for more robust approaches that can effectively balance generalization with task-specific optimization through more comprehensive analysis of what makes prompts succeed or fail.

In this paper, we introduce Learning from Contrastive Prompts (LCP), a novel framework for automated prompt optimization. Operating in two stages, LCP first generates diverse prompt candidates to thoroughly explore the solution space, then creates new prompts by analyzing structural and semantic differences between high and low-performing examples. Unlike previous approaches, LCP leverages contrastive learning principles that have proven successful across various domains but remain unexplored for prompt engineering. This allows LCP to systematically compare effective and ineffective prompts, identifying the distinctive characteristics that drive performance. This contrastive approach enables a more nuanced understanding of prompt quality beyond what iterative refinement or error correction methods can achieve alone.

We demonstrate the effectiveness of our approach on the Big-Bench Hard benchmark (Suzgun et al., 2022).Our framework achieves a win rate of over 87% versus OPRO (Yang et al., 2024), AutoHint (Sun et al., 2023), DSPy (Opsahl-Ong et al., 2024), and ProTeGi (Pryzant et al., 2023). LCP especially excels at algorithmic and multi-step

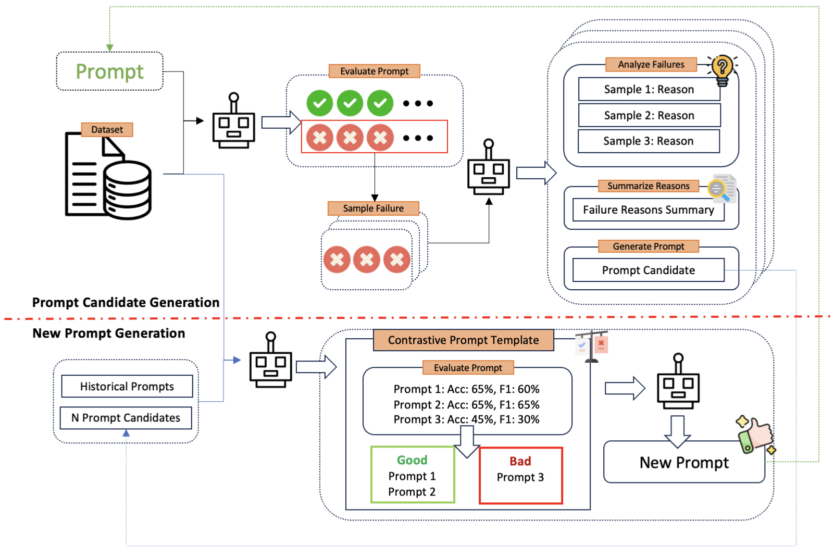


Figure 1: Learning from Contrastive Prompts (LCP) framework. LCP analyzes failures from an initial prompt and training data to generate new prompt candidates based on common error patterns. It leverages the inherent capabilities of LLMs to understand the underlying patterns through contrastive prompts to generate a new prompt.

arithmetic reasoning tasks.

## 2 Methodology

Our LCP framework (shown in Figure 1) consists of two key components that form the core of prompt optimization processes.

### 2.1 Prompt Candidate Generation

Starting from a train dataset and a prompt, we evaluate the prompt using backbone LLM and identify the failure examples. Our approach analyzes the failure reasons (*gradients*) for incorrectly predicted samples and summarize the reason into prompt feedback (*gradient reduction*). The reasons and feedback serve as ingredients for new prompt generation (*backward propagation*). The prompt template in this step is shown in Appendix A.6.

**Self-consistency for diversity injection.** Crafting prompt candidates solely based on the reason for each wrong sample (Pryzant et al., 2023) can lead to biased, overly specific prompts. While some existing methods attempt to address this by summarizing error feedback, they still risk over-fitting to selected samples. Our approach addresses these challenges in two ways: (1) using a higher temperature setting during generation to encourage creativity, and (2) generating multiple prompt candidates (N=10 based on our experiments) to better explore the prompt space. This diversity-aware approach helps avoid overfitting to specific error patterns while maintaining the benefits of learning from failures.

### 2.2 New Prompt Generation

Now that we have $N$ prompt candidates from the previous step, our goal is to generate a new prompt using them. First, we assign a score to each candidate based on its inference performance on the training set. We then rank all the candidates according to their scores. Inspired by contrastive learning, we instruct the LLM to identify the underlying patterns that distinguish good prompts from bad prompts. Specifically, we define the top-$K$ prompts as the good prompts and the bottom-$K$ prompts as the bad prompts, and we use the meta-prompt shown in Appendix A.6 to instruct the LLM to generate a new prompt that follows the underlying pattern of good prompts while improving the performance. The generated prompts from previous iterations can also influence the optimization process, leading to a better performance. Therefore, we integrate these prompts into the prompt candidates to ensure that the accumulated knowledge from past iterations contributes to the ongoing optimization process. We add the prompts generated in past along with the newly generated prompt.

The main contribution of our work lies in the use of contrastive prompts to understand the underlying patterns between effective and ineffective prompts. By integrating prompts from previous iterations, the distinction between good and bad prompts becomes more pronounced over time. Our method is motivated by the human learning process, which relies on understanding both successful and unsuccessful approaches to develop reasoning skills.

| TASK | LCP | AutoHint | OPRO | ProTeGi | DSPy |
|---|---|---|---|---|---|
| | Last / Best | Last / Best | Last / Best | Last / Best | Last / Best |
| *Algorithmic and Multi-Step Arithmetic Reasoning* | | | | | |
| geometric_shapes | **51.25 / 61.00** | 45.00 / 45.00 | 33.50 / 33.50 | 35.70 / 41.80 | 42.0 |
| logical_deduction_three_objects | **92.50 / 90.25** | 76.00 / 76.00 | 70.50 / 76.00 | 67.90 / 70.20 | 81.0 |
| logical_deduction_five_objects | **71.50 / 74.50** | 52.00 / 52.00 | 54.00 / 65.00 | 54.20 / 57.70 | 48.5 |
| logical_deduction_seven_objects | **62.00 / 62.50** | 2.00 / 2.00 | 48.00 / 48.50 | 51.30 / 57.00 | 56.0 |
| penguins_in_a_table | **97.86 / 96.15** | 86.30 / 86.30 | 87.20 / 94.00 | 59.66 / 59.66 | 60.68 |
| reasoning_about_colored_objects | 85.30 / 84.40 | 85.50 / 85.50 | **90.50 / 90.50** | 69.90 / 75.80 | 73.5 |
| temporal_sequences | **98.25 / 97.00** | - / - | 77.50 / 80.00 | 86.80 / 91.90 | 94.5 |
| tracking_shuffled_objects_three_objects | 92.00 / 99.00 | - / - | **99.50 / 99.50** | 69.60 / 84.00 | 40.0 |
| tracking_shuffled_objects_five_objects | **90.50 / 95.50** | - / - | 82.50 / 89.50 | 62.00 / 87.40 | 24.0 |
| tracking_shuffled_objects_seven_objects | **92.10 / 98.30** | - / - | 72.50 / 92.00 | 33.10 / 64.90 | 28.5 |
| Win Rate (%) | **93.00 / 93.00** | 28.00 / 25.00 | 55.00 / 57.00 | 35.00 / 38.00 | 40.00 / 35.00 |
| *Natural Language Understanding* | | | | | |
| disambiguation_qa | **66.83 / 66.33** | 55.00 / 57.00 | 50.00 / 50.00 | 63.10 / 63.80 | 55.5 |
| hyperbaton | **78.25 / 84.00** | 63.00 / 63.00 | 29.00 / 42.50 | 33.50 / 63.60 | 57.0 |
| salient_translation_error_detection[1] | 57.25 / **69.50** | **65.00** / 67.00 | 63.00 / 66.50 | 52.30 / 63.90 | 59.5 |
| snarks | 65.73 / 70.98 | **84.40 / 84.40** | 67.80 / 67.80 | 43.64 / 65.17 | 64.34 |
| Win Rate (%) | 69.00 / **94.00** | **75.00** / 69.00 | 38.00 / 25.00 | 25.00 / 50.00 | 44.00 / 12.00 |
| *Use of World Knowledge* | | | | | |
| date_understanding | 75.50 / 74.50 | 75.50 / 75.50 | **80.50 / 80.50** | 62.80 / 70.40 | 55.0 |
| movie_recommendation | **87.75 / 85.50** | 72.00 / 72.00 | 36.00 / 36.00 | 59.50 / 78.00 | 72.5 |
| ruin_names | 76.50 / 75.25 | **76.50 / 79.50** | 68.00 / 68.00 | 67.60 / 76.40 | 68.5 |
| Win Rate (%) | **75.00 / 67.00** | 58.00 / **67.00** | 42.00 / 33.00 | 17.00 / 58.00 | 42.00 / 25.00 |
| *Multilingual Knowledge and Reasoning* | | | | | |
| salient_translation_error_detection | 57.25 / **69.50** | **65.00** / 67.00 | 63.00 / 66.50 | 52.30 / 63.90 | 59.5 |
| Win Rate (%) | 25.00 / **100.0** | **100.0** / 75.00 | 75.00 / 50.00 | 0.00 / 25.00 | 50.00 / 0.00 |
| **Overall Win Rate (%)** | **82.81 / 87.50** | 57.69 / 55.77 | 45.31 / 42.19 | 25.00 / 40.63 | 37.50 / 23.44 |

Table 1: Test accuracy of prompt optimization approaches on four types of 17 BBH tasks for last iteration prompt (Last) versus the prompt with best training accuracy (Best). Average across 5 runs are reported. - indicates cases where AutoHint could not produce meaningful results. Blue indicates overall best results for Last or Best. **Bold** indicates the highest row value. Win rates are calculated with pair-wise comparisons following Liang et al., 2022.

## 3 Experiments

### 3.1 Setup

**Benchmarks.** Our evaluation benchmark is a subset of the Big-Bench Hard dataset (Suzgun et al., 2022), consisting of 17 challenging multi-choice tasks. The tasks are diverse, spanning across various categories like natural language understanding, the use of world knowledge, multilingual knowledge and reasoning, and algorithmic and multi-step arithmetic reasoning, making it a comprehensive test for our framework. We report results for each task category based on the keyword taxonomy provided by Big-Bench Hard dataset[2].

**Baselines.** We compare our approach with four existing methods. AutoHint optimizes prompts based on wrong samples in two iterations, using hint generation and summarization (Sun et al., 2023). OPRO optimizes prompts by maintaining a ranking list of historical prompts and relying solely on that (Yang et al., 2024). ProTeGi improves prompts through gradient descent step guided by a beam search and bandit selection procedure ProTeGi (Pryzant et al., 2023). MIPRO focus on optimization of multi-stage LM programs (Opsahl-Ong et al., 2024) which is integrated into DSPy (Khattab et al., 2023). Since these works used LLMs such as the GPT family (Brown et al., 2020) and the PaLM family (Chowdhery et al., 2023), which we don't have access to, we reimplemented their techniques on Claude-3-sonnet (Anthropic, 2024), which we also used in our framework, to ensure a fair comparison.

**Implementation details.** We use the same data split as OPRO on the Big-Bench Hard dataset, with 50 samples for training and 200 samples for testing. The temperature is set to 1.0. The maximum number of iterations is set to 50, followed by a selection step. In each random sampling step, we select 3 incorrectly predicted samples and repeat this step 10 times. For contrastive prompts, we select 3 good prompts and 3 bad prompts from the ranking list.

### 3.2 Results

Our prompt optimization begins with the initial prompt "Let's solve the problem." in the same fashion as OPRO and AutoHint. We report the results on last iteration from our method and baselines as well as from the prompt with best performance on the training set. Either choice is in line with previous works ( (Yang et al., 2024; Sun et al., 2023)) as strategies like a separate validation set,

[2]https://github.com/google/BIG-bench/blob/main/keywords.md

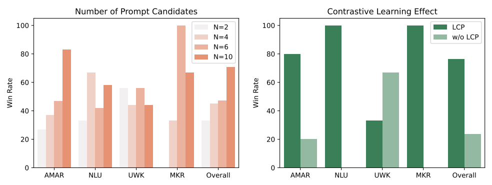


Figure 2: Ablation study with number of prompt candidates ($N$) on the left and effect of contrastive learning (w/ and w/o constrastive learning) on the right. AMAR refers to Algorithmic and Arithmetic, NLU to Natural Language Understanding, UWK to Understanding of World Knowledge, and MKR to Multilingual Knowledge and Reasoning. Reported are win rates on the prompt selected with best training set performance.

does not provide any benefit owing to being highly correlated with training performance. Also, we did not observe over-fitting with LCP. For further discussion, please refer to Appendix A.1. While we report results from both, given a relatively high variation and a slightly lower performance using the last prompt (47% win rate versus 53% for best training set prompt), we recommend using best prompt on the training set as the selected prompt.

As shown in Table 1, our LCP framework achieves the best performance with a win rate of 82% compared to baselines when using the last iteration prompt, and 87% when using the best prompt on training set. We believe the reason for LCP's superior performance over AutoHint is that LCP overcomes AutoHint's limitation in summarizing diverse hints. In contrast to OPRO, we take advantage of LLMs' inherent capability to contrast good prompts and bad prompts, making the process easier and more detailed than relying on a ranked list. Using contrastive prompts directly aligns with the way LLMs are fine-tuned with preference modeling, by learning to rank and distinguish between better and worse options (Rafailov et al., 2024). Additionally, we pay more attention to failures than OPRO, which solely relies on the generated prompts and their corresponding scores. The results highlight our framework's strong performance particularly on algorithmic and multi-step arithmetic reasoning tasks. This is understandable as algorithmic and arithmetic tasks involve more detailed instructions versus the other three categories which LCP excels through its contrastive and diversity injection mechanisms.

## 4 Discussion

Contrastive learning serves as the cornerstone of our approach, enabling LCP to extract meaningful patterns from both high and low-performing prompts. As shown in Figure 2 and Table 2, when we compare our full contrastive framework against a variant that simply receives ranked prompt candidates, contrastive learning achieves a 76% win rate. To better understand how contrastive learning influences prompt quality, we examine prompts generated by LCP compared to those from OPRO and AutoHint for representative tasks (Examples shown in Table 6). LCP prompts frequently incorporate explicitly numbered steps, systematic problem-solving frameworks, and task-specific methodologies rather than general principles. This structured approach is particularly evident in algorithmic and arithmetic tasks. For example, in logical reasoning tasks, LCP prompts methodically outline approaches for deconstructing complex problems. The contrastive nature of our framework appears to extract and amplify the organizational elements that distinguish effective instructions.

The diversity of the prompt space exploration directly impacts the quality of this contrastive learning. Our ablation experiments with varying numbers of prompt candidates ($N = 2, 4, 6, 10$) demonstrate that increasing $N$ consistently improves performance (Figure 2, Table 3), with each additional candidate providing more contrastive signal. While larger values of $N$ showed limited additional benefit relative to the increased computational cost, making $N = 10$ a practical choice. This highlights how diversity injection and contrastive learning work synergistically: greater diversity provides richer material for contrastive analysis, which in turn yields more effective prompt optimization. Further ablation studies can be found in A.3- A.5.

## 5 Conclusion

In this paper, we proposed Learning from Contrastive Prompts (LCP), a comprehensive framework for prompt optimization. LCP outperformed existing methods with an 87% win rate on Big-Bench Hard benchmark. Our analysis demonstrates that the synergy between diversity injection and contrastive learning enables a more principled exploration of the prompt space, particularly benefiting complex reasoning tasks. The contrastive learning paradigm represents a fundamental advancement in automated prompt engineering by providing a principled way to extract insights from the complex relationship between prompt structure and model performance, moving beyond traditional iterative refinement or error correction methods.

## Limitations

**Computation Cost.** The multiple prompt candidate generation and contrastive learning components of LCP require significant computational resources. Generating ten prompt candidates per iteration and evaluating them needs substantial LLM calls. This computational overhead may limit practical applications in resource-constrained settings.

**Convergence.** We presented convergence behavior in Appendix A.1 to share and highlight with the community an area that warrants further investigation. While our method is stochastic in nature, this challenge is not unique to LCP. For instance, the OPRO paper (e.g., Figure 23) shows similar non-converging behavior across multiple tasks, yet this was not explicitly discussed in the paper or other papers in the area. We believe that acknowledging and understanding this characteristic is important, and we view it as a valuable direction for future work to develop more stable prompt adaptation methods.

## References

AI Anthropic. 2024. The claude 3 model family: Opus, sonnet, haiku. *Claude-3 Model Card*, 1.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, and 1 others. 2023. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113.

Qingyan Guo, Rui Wang, Junliang Guo, Bei Li, Kaitao Song, Xu Tan, Guoqing Liu, Jiang Bian, and Yujiu Yang. 2024. Connecting large language models with evolutionary algorithms yields powerful prompt optimizers. In *The Twelfth International Conference on Learning Representations*.

Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan, Saiful Haq, Ashutosh Sharma, Thomas T. Joshi, Hanna Moazam, Heather Miller, Matei Zaharia, and Christopher Potts. 2023. Dspy: Compiling declarative language model calls into self-improving pipelines. *CoRR*, abs/2310.03714.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213.

Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, and 1 others. 2022. Holistic evaluation of language models. *arXiv preprint arXiv:2211.09110*.

Krista Opsahl-Ong, Michael J Ryan, Josh Purtell, David Broman, Christopher Potts, Matei Zaharia, and Omar Khattab. 2024. Optimizing instructions and demonstrations for multi-stage language model programs. *arXiv preprint arXiv:2406.11695*.

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.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2024. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36.

Abel Salinas and Fred Morstatter. 2024. The butterfly effect of altering prompts: How small changes and jailbreaks affect large language model performance. *arXiv preprint arXiv:2401.03729*.

Hong Sun, Xue Li, Yinchuan Xu, Youkow Homma, Qi Cao, Min Wu, Jian Jiao, and Denis Charles. 2023. Autohint: Automatic prompt optimization with hint generation. *arXiv preprint arXiv:2307.07415*.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V Le, Ed H Chi, Denny Zhou, and 1 others. 2022. Challenging big-bench tasks and whether chain-of-thought can solve them. *arXiv preprint arXiv:2210.09261*.

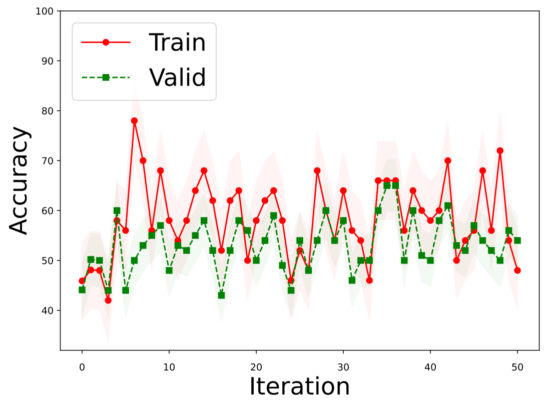
Xinyuan Wang, Chenxi Li, Zhen Wang, Fan Bai, Haotian Luo, Jiayou Zhang, Nebojsa Jojic, Eric Xing, and Zhiting Hu. 2024. Promptagent: Strategic planning with language models enables expert-level prompt optimization. In *The Twelfth International Conference on Learning Representations*.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V Le, Denny Zhou, and Xinyun Chen. 2024. Large language models as optimizers. In *The Twelfth International Conference on Learning Representations*.

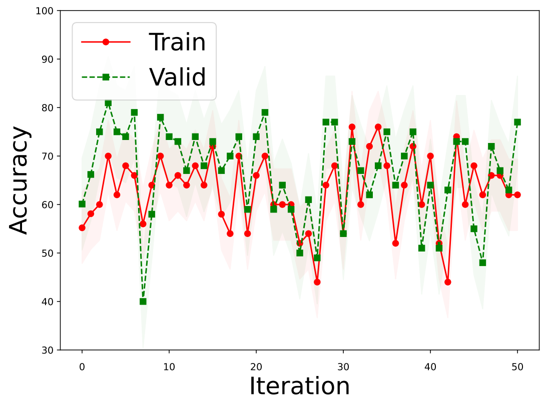
Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. 2023. Large language models are human-level prompt engineers. In *The Eleventh International Conference on Learning Representations*.

## A Appendix

### A.1 Discussion about Validation Set



(a) geometric_shapes task



(b) disambiguation_qa task

Figure 3: Accuracy as a function of number of training examples for LCP, AutoHint, and OPRO.

We did not use a validation set for our experiments following OPRO (Yang et al., 2024) and AutoHint (Sun et al., 2023), and based on our experiments. We show the training and validation accuracy curves when we do setup a validation set aside in Figure 3 on two tasks. We use a split of 33.33% training, 33.33% validation, and 33.33% testing sets to show these results.

We observe that there is no inherent bias-variance trade-off between the training/validation accuracies; typically validation accuracy follows training accuracy. We observe a moderate Spearman's correlation between 0.45-0.55 (p-values$<$0.001), showing that they are quite correlated. Further, we see that our method does not really overfit; training accuracy is lower or similar to validation accuracy unlike overfitting exhibited by OPRO as noted by (Yang et al., 2024). Unlike traditional fine-tuning machine learning regime where the training data gets embedded into the model weights, it is quite clear on how to define over-fitting in prompt optimization except prompts becoming too specific to the training samples. Since prompt accuracies change significantly iteration-over-iteration, further exploration is needed in this space to devise a way of final prompt selection. To keep it consistent with prior works (Yang et al., 2024; Sun et al., 2023) and to keep things simple in absence of an evidence of over-fitting, we chose to use last iteration prompt and prompt with best accuracy on training set.

### A.2 Detailed Results on LCP Ablations

Table 2 and Table 3 show detailed results on accuracies and win rates over the 17 tasks for the BBH data for the two ablation studies: w/ and w/o contrastive learning, and number of generated prompt candidates, respectively.

### A.3 Number of Contrastive Prompts

Table 4 shows the ablation of number of selected prompts for contrastive learning's feedback. We observe there is some variation in the performance across the number of prompts but no clear trend. Hence, no clear choice of which number of prompts to select. We chose 3 as higher number of prompts incur much more computation costs.

### A.4 Number of selected wrong data samples

AutoHint (Sun et al., 2023) observed that using no more than 3 samples per iteration achieves the best performance, as more samples could confuse the LLM when generating the summary. We also investigate how the performance varies with different numbers of selected wrong samples in Table 5. We do not observe a clear benefit of increasing the number from three. Hence, we use three wrong samples during our experiments in accordance with AutoHint.

### A.5 Number of Training Examples

To provide insights into the number of examples required for our method to maintain effectiveness, we report the performance when using 5, 10, 25, and 50 examples for training, in Figure 4 for three tasks. We notice a trend when we analyzed the training plots. For tasks like `reasoning about colored objects` whose training accuracy was relatively

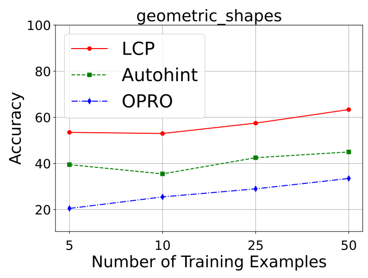
[2]Note, salient_translation_error_detection task comes under both Natural Language Understanding and Multilingual Knowledge and Reasoning but is only counted once in the overall win rates.

| **TASK** | **LCP** | **LCP (w/o contrastive learning)** |
|---|---|---|
| | Last / Best | Last / Best |
| *Algorithmic and Multi-Step Arithmetic Reasoning* | | |
| geometric_shapes | 51.25 / **61.00** | **53.50** / 56.00 |
| logical_deduction_three_objects | 92.50 / 90.25 | **93.00 / 93.00** |
| logical_deduction_five_objects | **71.50 / 74.50** | 71.00 / 70.00 |
| logical_deduction_seven_objects | **62.00 / 62.50** | 58.00 / 53.00 |
| penguins_in_a_table | **97.86 / 96.15** | 94.23 / 93.87 |
| reasoning_about_colored_objects | **85.30 / 84.40** | 80.50 / 83.50 |
| temporal_sequences | **98.25** / 97.00 | 94.50 / **98.50** |
| tracking_shuffled_objects_three_objects | **92.00 / 99.00** | 89.50 / 90.00 |
| tracking_shuffled_objects_five_objects | 90.50 / **95.50** | **94.50** / 92.50 |
| tracking_shuffled_objects_seven_objects | **92.10 / 98.30** | 89.50 / 92.50 |
| Win Rate (%) | **70.00 / 80.00** | 30.00 / 20.00 |
| *Natural Language Understanding* | | |
| disambiguation_qa | 66.83 / **66.33** | **70.50** / 63.50 |
| hyperbaton | 78.25 / **84.00** | **79.00** / 79.50 |
| salient_translation_error_detection | 57.25 / **69.50** | **66.50** / 66.00 |
| snarks | 65.73 / **70.98** | **67.73** / 68.23 |
| Win Rate (%) | 0.00 / **100.00** | **100.00** / 0.00 |
| *Use of World Knowledge* | | |
| date_understanding | 75.50 / 74.50 | **77.00 / 75.00** |
| movie_recommendation | **87.75 / 85.50** | 85.00 / 75.00 |
| ruin_names | 76.50 / 75.25 | **77.50 / 78.00** |
| Win Rate (%) | 33.00 / 33.00 | **67.00 / 67.00** |
| *Multilingual Knowledge and Reasoning* | | |
| salient_translation_error_detection | 57.25 / **69.50** | **66.50** / 66.00 |
| Win Rate (%) | 0.00 / **100.00** | **100.00** / 0.00 |
| **Overall Win Rate (%)** | 47.06 / **76.47** | **52.94** / 23.53 |

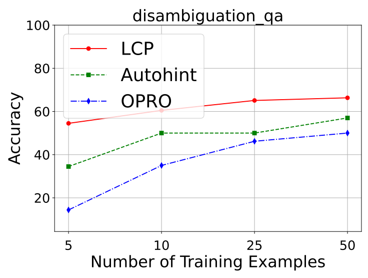
Table 2: Test accuracy of LCP with and w/o contrastive learning on four types of 17 BBH tasks for last iteration prompt (Last) versus the prompt with best training accuracy (Best). Blue indicates overall best win rates for Last or Best.

| **TASK** | $N = 10$ | $N = 6$ | $N = 4$ | $N = 2$ |
|---|---|---|---|---|
| | Last / Best | Last / Best | Last / Best | Last / Best |
| *Algorithmic and Multi-Step Arithmetic Reasoning* | | | | |
| geometric_shapes | 51.25 / **61.00** | **54.00** / 58.50 | 50.50 / 60.00 | 52.00 / 56.50 |
| logical_deduction_three_objects | **92.50** / 90.25 | 84.00 / **90.50** | 87.00 / 87.00 | 82.00 / 80.50 |
| logical_deduction_five_objects | **71.50 / 74.50** | 62.00 / 62.00 | 59.00 / 59.00 | 60.00 / 60.50 |
| logical_deduction_seven_objects | 62.00 / **62.50** | **63.00** / 58.50 | 59.00 / 55.00 | 54.50 / 55.00 |
| penguins_in_a_table | **97.86** / 96.15 | 94.87 / 94.87 | 96.58 / **96.58** | 95.73 / 95.73 |
| reasoning_about_colored_objects | 85.30 / 84.40 | 83.00 / **87.00** | **86.50** / 83.50 | 82.50 / 82.50 |
| temporal_sequences | 98.25 / 97.00 | **99.50** / 96.50 | 96.00 / 96.00 | 95.50 / **99.50** |
| tracking_shuffled_objects_three_objects | 92.00 / **99.00** | 94.50 / 94.50 | 94.00 / **99.00** | **95.50** / 95.50 |
| tracking_shuffled_objects_five_objects | 90.50 / **95.50** | 78.50 / 78.50 | 92.50 / 92.50 | **98.50** / 95.00 |
| tracking_shuffled_objects_seven_objects | 92.10 / **98.30** | **93.50** / 96.50 | 85.00 / 88.50 | 79.00 / 79.00 |
| Win Rate (%) | **63.00 / 83.00** | 60.00 / 47.00 | 43.00 / 37.00 | 33.00 / 27.00 |
| *Natural Language Understanding* | | | | |
| disambiguation_qa | 66.83 / 66.33 | 66.00 / 68.00 | **68.00 / 71.00** | 66.00 / 63.00 |
| hyperbaton | 78.25 / **84.00** | 82.00 / 81.50 | 82.00 / 82.50 | 81.50 / **83.50** |
| salient_translation_error_detection | 57.25 / 69.50 | **70.00 / 70.00** | 65.50 / 68.50 | 67.50 / 65.50 |
| snarks | 65.73 / 70.98 | 60.14 / 60.14 | **74.13 / 76.22** | 71.33 / 71.33 |
| Win Rate (%) | 25.00 / 58.00 | 42.00 / 42.00 | **75.00 / 67.00** | 42.00 / 33.00 |
| *Use of World Knowledge* | | | | |
| date_understanding | **75.50** / 74.50 | 72.00 / 72.00 | 73.50 / 74.00 | 74.00 / **76.00** |
| movie_recommendation | **87.75** / 85.50 | 86.50 / **87.50** | 79.00 / 77.00 | 82.00 / 83.00 |
| ruin_names | 76.50 / 75.25 | 76.00 / 79.50 | 79.00 / **80.00** | **80.00** / 77.50 |
| Win Rate (%) | **78.00** / 44.00 | 22.00 / **56.00** | 33.00 / 44.00 | 67.00 / **56.00** |
| *Multilingual Knowledge and Reasoning* | | | | |
| salient_translation_error_detection | 57.25 / 69.50 | **70.00 / 70.00** | 65.50 / 68.50 | 67.50 / 65.50 |
| Win Rate (%) | 0.00 / 67.00 | **100.0 / 100.0** | 33.00 / 33.00 | 67.00 / 0.00 |
| **Overall Win Rate (%)** | **56.86** / **70.59** | 49.02 / 47.06 | 49.02 / 45.10 | 41.18 / 33.33 |

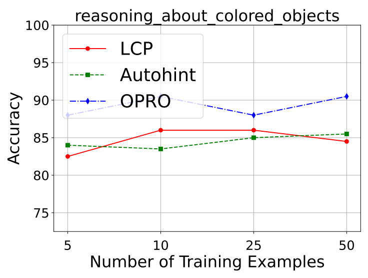
Table 3: Test accuracy of LCP with different values of number of prompt candidates ($N$) on four types of 17 BBH tasks for last iteration prompt (Last) versus the prompt with best training accuracy (Best). Blue indicates overall best win rates for Last or Best.



(a) geometric_shapes task



(b) disambiguation_qa task



(c) reasoning colored objects

Figure 4: Accuracy as a function of number of training examples for LCP, AutoHint, and OPRO.

| **Task** | **2** | **3** | **4** | **5** |
|---|---|---|---|---|
| date_understanding | **79.0** / 76.5 | 75.5 / 74.5 | 77.5 / 73.5 | 78.5 / **78.0** |
| reasoning_about_colored_objects | **85.5** / 83.5 | 85.3 / **84.4** | 84.5 / 81.0 | 85.5 / 84.0 |

Table 4: Performance results with ablation on number of prompts used for Contrastive learning. We present the last / best performance for each task.

| **Task** | **3** | **4** | **5** | **6** |
|---|---|---|---|---|
| date_understanding | 75.5 / 74.5 | 70.5 / **75.0** | **77.0** / 72.5 | **77.0** / 74.0 |
| reasoning_about_colored_objects | **85.3** / 84.4 | 81.0 / **85.5** | 79.5 / 82.5 | 82.5 / 83.0 |

Table 5: Performance results with ablation on number of wrong samples performance. We present the last / best performance for each task.

flat during the iterations, number of examples had little effect, while for tasks like `geometric shapes` with training curves showing considerable improvement across training iterations, we see a consistent improvement in the performance as number of examples increased. Further, we observe that LCP is relatively more sample efficient, giving a relatively higher performance at lower number of samples versus AutoHint or OPRO. This can be attributed to our multiple candidate generation for contrastive learning that helps model explore diverse prompts to derive insights.

## A.6 Meta prompt

**Reason Generation Prompt**

Given input: [INPUT]
And its expected output: [OUTPUT]

Explain the reason why the input corresponds to the given expected output. The reason should be placed within tag <reason></reason>.

**Summarization Prompt**

Given input and expected output pairs, along with the reason for generated outputs, provide a summarized common reason applicable to all cases within tags <summary> and </summary>.
The summary should explain the underlying principles, logic or methodology governing the relationship between the inputs and corresponding outputs. Avoid mentioning any specific details, numbers, or entities from the individual examples, and aim for a generalized explanation.

**Contrastive Prompt**

Given $m$ examples of good prompts and their corresponding scores and $m$ examples of bad prompts and their corresponding scores, explore the unerlying pattern of good prompts, generate a new prompt based on this pattern. Put the new prompt within tag <prompt> and </prompt>.

Good prompts and scores:
Prompt 1: [PROMPT 1]
Score: [SCORE 1]
...
Prompt $m$: [PROMPT $m$]
Score: [SCORE $m$]

Summarize these explanation and generate a new prompt accordingly. Put the new prompt within tag <prompt> and </prompt>.

## A.7 Examples of Generated Prompts

Table 6: Comparison of prompts generated by our proposed LCP, AutoHint and OPRO.

| TASK | Method | Generated Prompts |
|---|---|---|
| geometric_shapes | LCP | To determine the expected output shape from an SVG path element, follow these steps:<br>1. Parse the path data string and identify the distinct line segments formed by the path commands and coordinates. 2. Count the number of line segments, as each segment represents a side of the resulting polygon. 3. Map the number of sides to the corresponding polygon name (e.g., 3 sides = triangle, 4 sides = quadrilateral, 5 sides = pentagon, 6 sides = hexagon, 7 sides = heptagon, and so on). 4. Note that the path data typically starts with a moveTo (M) command to set the initial point, followed by lineTo (L) commands that define the sides of the polygon by connecting the vertices.<br>By following this process, you can accurately identify the expected output shape based on the number of sides determined from the SVG path data. |
| | AutoHint | The summarized reason for the expected outputs is that the SVG path element with the given "d" attribute value defines a series of line segments that connect specific points, forming a closed geometric shape. The coordinates specified in the "d" attribute determine the vertices or corners of the shape, and the sequence of line segments connecting these points outlines the overall geometry. By analyzing the number of sides or angles formed by the connected line segments, one can identify the type of polygon or shape being drawn, such as a kite, trapezoid, or pentagon. |
| | OPRO | The SVG path element specifies a sequence of line segments and coordinates that connect to form a closed polygonal shape. By analyzing the path data, which defines the vertices and line segments, the number of sides in the resulting polygon can be determined. For example, if the path data forms a closed shape with 3 vertices and 3 line segments, the expected output would be a triangle. Similarly, shapes with 4, 5, 6, or more sides would correspond to quadrilaterals, pentagons, hexagons, and so on, respectively. The task is to examine the path data and identify the type of polygon created based on the number of line segments and vertices in the closed shape. |

| | | |
|---|---|---|
| logical_deduction_three_objects | LCP | 1) Carefully analyzing the given comparative statements that describe relationships or ordering among three objects.<br>2) Using logical reasoning and deductive skills to establish the complete order or ranking of the objects based on the provided information.<br>3) Identifying the object that satisfies the specific condition or criteria mentioned in the question (e.g., second-oldest, second-most expensive) based on the deduced order or ranking.<br>4) The reasons should explain how the correct option is determined by logically interpreting the statements and accurately reflecting the deduced order through careful analysis and reasoning. |
| | AutoHint | The summarized reasons are: In these types of questions, the key is to carefully analyze the given statements and logically deduce the order or ranking of the objects/entities based on the provided information. The statements establish relationships between the objects, such as "newer than," "less expensive than," or "finished before/after." By piecing together these relationships, one can infer the correct order or ranking, eliminating the incorrect options and identifying the valid answer choice. |
| | OPRO | To solve these problems, the crucial step is meticulously analyzing the provided statements that describe the comparisons or relative qualities among the three objects. By logically interpreting these clues and deducing their implications, we can establish the complete ordering or ranking of the objects. Once this order is determined, the task is to identify the object that satisfies the specified condition, such as being the second-most expensive or second-newest, to arrive at the correct solution. |

| | | |
|---|---|---|
| logical_deduction_five_objects | LCP | To reliably solve ordering and sequence problems that require deducing the correct arrangement based on logical constraints, follow these steps:<br>1. Carefully read and analyze each given statement or condition to identify constraints on the ordering, such as explicit mentions of absolute positions (e.g. "X is the first"), relative comparisons (e.g. "Y is more expensive than Z"), or ranges (e.g. "A is one of the three cheapest").<br>2. Translate each statement into a logical constraint on the ordering and use reasoning to deduce partial orderings or relationships between objects or entities based on these constraints.<br>3. Systematically combine these partial orderings and relative relationships by considering all possible arrangements and eliminating any that contradict the given information.<br>4. Construct the complete sequence or ranking that satisfies all constraints simultaneously.<br>5. Map this deduced ordering to the specific condition asked in the question (e.g. third from the left, second most expensive) to determine the expected output or correct option.<br>By methodically identifying constraints, applying logical reasoning to deduce orderings, and mapping the solution to the requested condition, you can reliably solve these types of ordering problems. |
| | AutoHint | The summarized reasons are: The expected output is based on the explicit statements or logical implications provided in the given information, which establish the correct ordering or relationship among the entities involved. |

| | | |
|---|---|---|
| | OPRO | To solve ordering problems based on logical statements, follow this systematic approach: 1. Carefully read through all the given statements that describe relationships between objects or their positions relative to each other or the ends. 2. Identify any direct statements about the exact position of an object (e.g., "X is the third from the left"). 3. Look for comparative statements that indicate the order between two objects (e.g., "Y is to the right of Z"). 4. Use the identified information to reconstruct the overall order by positioning objects based on their relationships and constraints. 5. While constructing the order, make valid logical inferences from the given information to deduce the positions of objects not directly specified. 6. Check if the deduced order satisfies all the given statements consistently without violating any constraints. 7. The option corresponding to this fully reconstructed order that meets all the conditions is the correct answer. By meticulously following this step-by-step approach and making careful logical deductions, you can reliably solve ordering problems based on statements describing relative positions. |
| logical_deduction_seven_objects | LCP | To solve logical reasoning problems involving the order or relative positions of objects, follow this systematic approach: 1) Thoroughly read and comprehend all the given statements or constraints describing the relationships, positions, or orders of the objects. 2) Identify any definitive placements or orderings of objects that are explicitly stated in the constraints. 3) Use logical deduction to derive additional constraints, implications, or relationships based on the given information and the definitive placements/orderings identified in step 2. 4) Methodically evaluate each option, eliminating those that violate any of the stated constraints or logically deduced implications from step 3. 5) The remaining option(s) that adheres to all the given constraints and deduced implications is the expected correct output. The key is to diligently analyze the information, recognize logical relationships between objects, employ deductive reasoning based on the constraints, and arrive at a solution that is consistent with all the provided information. |

| | | |
|---|---|---|
| | AutoHint | Understood, I will provide a general summary of the reasons for the expected outputs without referring to any specific examples or entities mentioned in the data. |
| | OPRO | To solve logic problems involving the ordering of a set of objects, the key steps are: 1. Carefully read and analyze all the given statements describing the relative positions or characteristics of the objects. 2. Identify the logical constraints and relationships imposed by each statement, such as "X is to the left of Y" or "X is taller than Y". 3. Use logical reasoning to deduce the implications of these constraints on the positions of the objects relative to each other. 4. Systematically combine the deduced information to reconstruct the complete order while ensuring it satisfies all the provided conditions. 5. Eliminate any options that violate the inferred order or the given constraints. 6. The option that correctly reflects the deduced order based on the given information is the solution. |
| penguins_in_a_table | LCP | To solve a problem involving tabular data, one must carefully analyze the information presented in the given table(s). Identify the specific column(s) or data points that are relevant to answering the question. Then, perform any necessary operations on that data, such as sorting, filtering, counting, or calculations, as per the requirements stated in the question. After logically processing the relevant data, you can determine the correct answer choice or expected output. |
| | AutoHint | The summarized general reason for the expected outputs is that they are based on carefully analyzing the given information or data and applying logical reasoning to arrive at the correct answer. The expected outputs are determined by thoroughly understanding the context, identifying the relevant details, and making deductions or inferences based on the provided facts or conditions. |
| | OPRO | To effectively solve questions involving tabular data, carefully analyze the structure and contents of the given table(s). Identify the column(s) containing information pertinent to the question asked. Based on the requirements stated in the question, you may need to perform operations such as sorting the relevant column(s) in ascending or descending order, filtering the data based on certain criteria, counting specific occurrences, or calculating derived values using the data. Logically process the tabular data by applying the necessary operations, and use the resulting information to arrive at the correct answer choice. |

| | | |
|---|---|---|
| reasoning_about_colored_objects | LCP | To determine the expected output, study the provided set of items and their descriptions (color, shape, size, etc.). Take note of the particular attribute or condition specified in the question, such as "items of a certain color" or "items remaining after removing a specific type." Systematically go through each item, checking if it fulfills the stated condition. Count the total number of items that meet the criteria. The option that matches this final count represents the expected output. |
| | AutoHint | The summarized reasons for the expected outputs are: The questions provide information about a set of items arranged in a specific order or position relative to each other. The expected output is determined by carefully analyzing the given details, such as the colors of the items, their arrangement, and the specific item or position referenced in the question. By logically interpreting the spatial relationships and attributes described in the input, one can deduce the correct answer choice that satisfies the conditions stated in the question. |
| | OPRO | To solve the problem accurately, carefully read the given information to identify the set of items or objects described, along with their relevant attributes (such as color, type, etc.). Understand the specific condition or operation mentioned in the question (e.g., removing certain items, counting items with a particular attribute). Apply this condition or operation to the identified set of items, modifying or filtering the set as instructed. Then, logically analyze the resulting set of items to determine the option that correctly matches the final composition or count after applying the specified condition. |
| temporal_sequences | LCP | The expected output represents the sole remaining time window that is not accounted for in the person's daily schedule and activities as described. It is determined by meticulously considering all the provided information about the person's whereabouts and commitments throughout the day, as well as any relevant constraints like opening/closing hours of the location. By systematically eliminating all the other time slots occupied by the person's observed activities or locations, the correct answer emerges as the only unoccupied period when the person could have potentially visited the specified destination. |
| | AutoHint | I will provide a general summary of the reasons for the expected outputs, without referring to any specific examples or entities mentioned in the data. |

| | | |
|---|---|---|
| | OPRO | The solution involves carefully examining the timeline of events and activities provided in the problem. First, identify all the time slots where the person's whereabouts and activities are explicitly stated. Then, determine the remaining time window that is not covered by any of these known activities or constraints, such as the operating hours of the location mentioned. This unoccupied time period represents the only available opportunity for the person to have visited the specified destination (e.g., bakery, library, movie theater) before it closed for the day. By process of elimination, this remaining time slot becomes the most logical answer for when the person could have gone to the location in question. |
| tracking_shuffled_objects_three_objects | CP | To solve problems involving a sequence of swaps or exchanges between multiple people, it is crucial to carefully track the movement of each item or position through the entire series of swaps. Begin by noting the initial state, mapping which person or entity holds which item or position. Then, systematically follow each swap or exchange step-by-step, updating the holdings or positions after each swap according to the provided sequence. By meticulously tracking these changes through the entire set of swaps, you can determine the final state and identify the correct answer corresponding to the item or position held by the person or entity in question after the last swap involving them has occurred. |
| | AutoHint | Understood, I will provide a general summary of the reasons for the expected outputs without referring to any specific examples or entities mentioned in the data. |
| | OPRO | In these types of questions, there are typically several individuals (say, Alex, Maya, and Sameer) who are initially assigned certain roles or possessions (e.g., playing a sport position, holding a particular object). The problem then describes a sequence of swaps or trades between pairs of these individuals, where they exchange their roles or possessions. To determine the final role or possession of a specific individual after all the swaps, it is crucial to carefully follow the entire sequence, meticulously updating each individual's state after every swap. By accurately tracing the swaps from the initial assignments to the end, you can arrive at the correct final state or possession for the given individual. |

| | | |
|---|---|---|
| tracking_shuffled_objects_five_objects | LCP | In these types of problems involving swaps or exchanges, the key to arriving at the correct solution is to diligently track the sequence of changes that occur. By methodically following each swap or exchange step-by-step and updating the current state of assignments or positions, one can trace the path of how entities (people, objects, etc.) move from their initial state to the final outcome. Maintaining an organized record of the swaps and their effects allows you to meticulously trace the progression until the end, enabling you to determine the final configuration accurately. |
| | AutoHint | Understood, I will provide a general summary of the reasons for the expected outputs without referring to any specific examples or entities mentioned in the data. |
| | OPRO | In these types of problems involving swaps or trades, it is crucial to first understand the initial distribution of objects among a group of individuals. Then, methodically trace each swap or trade that occurs between pairs of individuals, carefully updating the ownership of objects after each step. By systematically following the given sequence of swaps from start to end, updating who holds which object after each transaction, the final state or ownership of objects can be accurately determined. |
| tracking_shuffled_objects_seven_objects | LCP | These types of problems involve tracking the movement of objects or assignments between a set of individuals through a given sequence of swaps or trades. The key to finding the correct solution is to start from the initial state where objects or roles are assigned to individuals, and then meticulously follow each swap step-by-step, updating the associations between individuals and objects/roles as dictated by the swap instructions. By carefully applying the entire sequence of swaps to the initial state, keeping an accurate record of how objects or assignments are redistributed among the individuals, the final state can be determined, revealing the correct output option for the object or role associated with a particular individual after all swaps have occurred. |
| | AutoHint | I will provide a general summary of the reasons for the expected outputs, without referring to any specific examples or entities mentioned in the data. |

| | | |
|---|---|---|
| | OPRO | In these types of problems involving swaps or exchanges of items (e.g., gifts, positions, partners) between individuals, the key to finding the solution is to diligently track the changes that occur after each swap. Start by understanding the initial state or arrangement of items among the individuals. Then, systematically follow the given sequence of swaps, updating the state by moving the items between individuals as specified. After applying all the swaps in the correct order, you will arrive at the final state, allowing you to determine the ultimate position or ownership of the particular item or individual in question. |
| disambiguation_qa | LCP | To correctly identify the antecedent (the noun or noun phrase that a pronoun refers to) in a sentence, follow these steps:<br>1. Examine the context and meaning of the sentence to understand the logical relationships and actions described.<br>2. Identify the nouns or noun phrases that the pronoun could potentially refer to based on grammatical agreement (number, gender, etc.) and proximity to the pronoun.<br>3. Evaluate each potential antecedent by substituting it for the pronoun. The antecedent should maintain the logical flow and coherence of the sentence.<br>4. If only one potential antecedent satisfies the criteria in step 3, that noun or noun phrase is the unambiguous antecedent.<br>5. If multiple potential antecedents satisfy the criteria, and the context does not provide enough information to disambiguate, then the pronoun usage is considered ambiguous. |
| | AutoHint | The summarized reasons for determining the antecedent of a pronoun in a sentence are as follows: The pronoun typically refers to the subject or object of the main clause that logically connects to the clause containing the pronoun. The context and logical flow of the sentence provide clues to identify the antecedent. If the pronoun can reasonably refer to multiple entities mentioned in the sentence, then the antecedent is considered ambiguous due to a lack of sufficient contextual information. |

| | | |
|---|---|---|
| | OPRO | To correctly identify the antecedent (the entity that a pronoun refers to) within a sentence, it is crucial to analyze the context and relationships described. The pronoun should logically reference the most plausible noun or noun phrase based on the meaning conveyed by the sentence. Pay close attention to the surrounding information and flow of ideas to determine which entity performs or is associated with the actions mentioned. If there are multiple potential antecedents and the context lacks sufficient details to disambiguate, then the pronoun usage is considered ambiguous, as the referent cannot be definitively pinpointed. |
| hyperbaton | LCP | In the English language, when multiple adjectives are used to describe a noun, they must follow a specific order to construct grammatically correct sentences. This conventional order is: Opinion, Size, Age, Shape, Color, Origin, Material, Qualifier/Purpose, Noun. Deviating from this standardized sequence results in unnatural and potentially incorrect phrasing. |
| | AutoHint | The summarized reason is: There are established conventions or rules for the correct order of adjectives when multiple adjectives are used to modify a noun. The expected output follows these conventions, ensuring that the adjectives are arranged in the proper sequence based on their specific categories or types. |
| | OPRO | In the English language, when multiple adjectives are used to describe a noun, they are expected to follow a specific order for clear and natural sentence construction. This established order places opinion adjectives first, followed by size, age, shape, color, origin, material, and purpose adjectives modifying the noun. Adhering to this conventional adjective order is crucial for coherence and proper comprehension of the description. |

| | | |
|---|---|---|
| salient_translation_error_detection | LCP | The expected output category should capture the type of error or change introduced in the English translation compared to the original German text. Consider the following error categories:<br>- Named Entities: Incorrect translation of proper names, locations, or other entities.<br>- Numerical Values: Missing, added, or altered numbers, dates, measurements, or numerical expressions.<br>- Modifiers/Adjectives: Changes to descriptive words, adjectives, or modifiers that alter the attributes or qualities of a noun.<br>- Negation/Antonyms: Introduction of negation, or swapping comparatives with their opposites/antonyms, altering the intended meaning.<br>- Trivial Factual Errors: Inaccuracies or mistakes in factual information unrelated to the other categories.<br>- Dropped Content: Significant omission of phrases, clauses, or parts of the original text in the translation.<br>Identify which of these error categories best describes the change or discrepancy observed in the given translation compared to the source German text. |
| | AutoHint | The summarized reasons for the expected outputs in the given examples are: The errors in the translations can be categorized into different types, such as Named Entities, Numerical Values, Modifiers or Adjectives, Negation or Antonyms, Facts, and Dropped Content. The expected outputs identify the specific type of error present in each translation. The reasons provided explain how the translation deviates from the original meaning or content, leading to the identified error type. This could involve missing or altering crucial information like names, numerical values, modifiers, introducing negations or antonyms, factual inaccuracies, or omitting significant clauses or content from the original text. |
| | OPRO | 1) Clearly stating that the expected output focuses on identifying the type of error introduced in the translation compared to the original text.<br>2) Listing and explaining the different categories of error types, such as changes to named entities, numerical values, modifiers/adjectives, negations/antonyms, factual errors, or dropped content.<br>3) Emphasizing that the expected output should correctly categorize the specific type of error present in the translation. |

| | | |
|---|---|---|
| snarks | LCP | Sarcasm relies on creating an intentional contradiction between the literal words used and the underlying sentiment being conveyed. It leverages techniques like hyperbole, irony, and rhetorical questioning to juxtapose opposing elements that clearly contradict common sense or reality. By expressing an exaggerated or mocking version of the opposite perspective, sarcastic statements unmask their true critical or derisive meaning beneath the facade of the contradictory words themselves. This discrepancy between the stated words and intended meaning is the hallmark of sarcastic communication. |
| | AutoHint | The summarized general reason for the expected sarcastic outputs in the given examples is that sarcasm is expressed through statements that contradict or exaggerate the intended meaning in an ironic or critical way. Sarcastic statements often convey the opposite of their literal meaning, using exaggeration, irony, or contradiction to imply criticism, mockery, or a different intended meaning than the literal words suggest. |
| | OPRO | Sarcastic statements rely on creating a deliberate contradiction or contrast between the literal meaning and the intended meaning conveyed through irony or mockery. They often employ techniques like exaggeration, rhetorical questions, and juxtaposing positive/negative sentiments to highlight this incongruity. The sarcasm arises from this clash between the stated words and the true critical intent behind them, suggesting the opposite of what is expressed literally. |
| date_understanding | LCP | 1) Emphasize carefully analyzing the provided information, such as the current or starting date, time intervals (days, months, years), and any context about leap years.<br>2) Outline the key steps of establishing the reference date, calculating the target date by properly applying the specified time periods forward or backward, and handling factors like the number of days in each month and year boundaries.<br>3) Highlight the importance of paying close attention to details and performing accurate calculations to arrive at the correct date in the specified format (MM/DD/YYYY). |
| | AutoHint | The summarized general reason for the expected outputs is that the questions provide specific details about a date or event, and the correct answer corresponds to the date or day that logically follows from those details, taking into account the calendar system and conventions for representing dates. |

| | | |
|---|---|---|
| | OPRO | To accurately determine a date based on given information, it is crucial to methodically follow these steps: 1. Identify the provided reference date or starting point from the details given. This could be a birth date, anniversary, or specific calendar date. 2. Determine the time period or duration to calculate from the reference date. This may be a number of days, weeks, months, or years to be added or subtracted. 3. Consider if the time period should be added to the reference date to get a future date, or subtracted to get a past date. Carefully account for this direction. 4. Perform the date calculation, properly applying the time period while taking into account factors like number of days in each month and adjusting for leap years when necessary. 5. Ensure the final calculated date is presented in the exact format requested (e.g. MM/DD/YYYY). By diligently analyzing all provided information and implementing precise step-by-step calculations while adhering to calendar conventions, the correct date can be determined reliably. |
| movie_recommendation | LCP | - Highly popular and critically acclaimed - Culturally impactful and became a phenomenon - Achieved mainstream success and global recognition - From a comparable time period or era as the reference movies - Represents a significant work in the context of popular cinema with broad appeal |
| | AutoHint | The summarized reasons for the expected outputs are: The expected output is chosen because it shares similar genres, tones, themes, and overall cinematic styles with the given examples. The selected movie aligns with the general mood, narrative elements, and target audience of the reference films, making it the most appropriate choice among the provided options. Factors like genre (drama, action, thriller, etc.), tone (serious, lighthearted, suspenseful, etc.), and thematic elements (overcoming adversity, romance, historical events, etc.) are considered to determine the most suitable option that resonates with the given examples in terms of overall cinematic experience. |

| | | |
|---|---|---|
| | OPRO | The expected output is a movie that aligns closely with the examples provided in terms of genre (e.g. action, drama, comedy), tone/mood (e.g. lighthearted, gritty, emotional), level of critical praise and cultural significance, as well as overall production values and widespread appeal. The reasoning involves identifying the commonalities between the listed movies in terms of factors like storytelling approach, themes explored, filmmaking techniques, and target audience, then selecting the option that best matches that collective profile in a way that would be considered a comparable cinematic experience for viewers familiar with the given examples. |
| ruin_names | LCP | The expected output involves humorous edits that playfully modify the original names or phrases through clever linguistic techniques. These may include substituting a word with one that contrasts humorously, splitting words and recombining the parts to create new meanings, or introducing elements from wildly different contexts to generate an amusing, incongruous juxtaposition with the original. The key is to introduce an element of wordplay, unexpected meaning, or absurdity that creates a comedic effect, while still maintaining enough familiarity with the source material for the reader to recognize and appreciate the creative twist. |
| | AutoHint | The summarized reasons are: The expected outputs are considered humorous edits because they involve wordplay or puns created by slightly modifying the original word, phrase, or name in a clever or unexpected way. This can include replacing letters with similar-sounding ones, altering the spelling, or making slight changes to the wording. These types of edits are often used for comedic effect, as they play with the audience's familiarity with the original text while introducing a new, humorous interpretation or meaning. |
| | OPRO | The expected outputs demonstrate clever and humorous modifications of familiar names, titles, or phrases. These edits playfully replace or alter certain words or letters to create an amusing contrast or incongruity with the original source material. Through techniques like wordplay, puns, and subtle linguistic substitutions, the humorous outputs inject an element of witty absurdity while still retaining a recognizable connection to the original. This form of intelligent and creative linguistic manipulation is an effective way to subvert expectations and elicit laughter by twisting the familiar into something comically unexpected. |