Analysis of Prompting Strategies for Math and Coding

COMP7607: Natural Language Processing - The University of Hong Kong

Fall 2025

Due: November 29, 23:59 PM

In our previous assignment, we explored the capabilities of LLMs in two domains: **math reasoning** (Lu et al., 2023) and **code generation** (Sun et al., 2024). In this assignment, we will continue to delve into how prompting affects the reasoning abilities of LLMs. Similarly, you can choose one task—either mathematics or coding—based on your interests, or you can do both.

You are highly encouraged to reuse the implementation from A1 to complete this assignment:)

Submit: You should submit your assignment to the COMP7607 Moodle page. You will need to submit (1) a PDF file **UniversityNumber.pdf** of your report, with **detailed experimental details**, your **analysis** and your **thinking** (2) a zip file **UniversityNumber.zip**, which includes:

- `.py` files, if any.
- `.ipynb` files, if any.
- Other files (e.g., data, prompts) you consider necessary.

Please note that the `UniversityNumber` is the number printed on your student card.

1 Introduction

Recap. Prompt engineering refers to methods for how to instruct LLMs for desired outcomes without updating model weights. In Assignment 1, we designed methods for prompting LLMs to improve accuracy in math problem-solving or code generation. In this assignment, we will conduct an in-depth exploration of prompt learning, focusing on how (1) prompt quality (2) the number of demonstrations (3) prompt diversity (4) prompt complexity affect task performance.

Note. As an **analytical assignment**, you can approach your analysis from any of the above angles. You can cover a wide range or focus deeply on one aspect. You can also propose new perspectives. Most importantly, we value your thinking and insights on how these factors affect math reasoning or code generation. Considering the API response rate, you can take a task subset for all experiments (but please specify this in your report).

2 In-Depth Analysis of Prompting Strategies for Math and Coding

We will analyze the impact of prompting strategies on math and coding tasks. You are encouraged to think creatively and freely design your analytical methods to complete the assignment. Feel free to integrate your analysis with the implementations from A1, such as self-refine (Madaan et al., 2023).

2.1 Prompt Quality

In most cases, we consider the given problem statement and demonstration to be correct, with the right format, rationale, and answers aligned with the problem to be solved. But what if they are incorrect? For example, if the problem statement is correct but the demonstration is wrong, or if the demonstration is correct but not relevant to our problem, how would they affect the performance of math reasoning or code generation? Please try to analyze this based on previous A1 implementations.
If you have no ideas, you can refer to the following papers:

- Towards Understanding Chain-of-Thought Prompting: An Empirical Study of What Matters (Wang et al., 2023)
- What In-Context Learning "Learns" In-Context: Disentangling Task Recognition and Task Learning (Pan et al., 2023)

Hint: You can try selecting some prompts used in A1 for GSM8K or HumanEval, "disturb" them, and then conduct your experiments.

2.2 Prompt Complexity

How does the complexity of prompts affect task performance? For the task to be solved, is it better if the problem statement is more detailed and the demonstration more complex? Or could simpler prompts sometimes yield better performance by reducing cognitive load on the model?

- Complexity-Based Prompting for Multi-Step Reasoning (Fu et al., 2023)
- DQ-LoRe: Dual Queries with Low Rank Approximation Re-ranking for In-Context Learning

Hint: You can try curating more complex/simpler prompts for your task and then conduct comparative experiments. For convenience, you may find some from prompt libraries like Chain-of-Thought Hub.

2.3 Number of Demonstrations

Given a fixed task statement, does the number of demonstrations affect task performance? Obviously, it does, but how exactly does it influence the performance? Will continuously increasing the number of demonstrations linearly enhance the LLM's math reasoning and coding capabilities? What happens if the number of demonstrations is reduced? Under which settings is performance most sensitive to changes in the number of demonstrations? Try to analyze prompting strategies from the perspective of the number of demonstrations.

- Language Models are Few-Shot Learners (Brown et al., 2020)
- Rethinking the Role of Demonstrations: What Makes In-Context Learning Work? (Min et al., 2022)

Hint: Researchers noticed this issue as early as the release of GPT-3 in 2020. If you are interested, you can review these classic works above before starting your experimental design.

2.4 Prompt Diversity

Is it better for prompts to be more diverse or more standardized? How would these choices impact the LLM's math and coding capabilities? Try to analyze them from the perspectives like: (1) Using different phrasing and sentence/code structures to guide LLMs, avoiding over-reliance on fixed formats.
(2) Providing varied task instructions or background information to help the model better understand the task requirements. (3) Using prompts with diverse styles and tones to improve the model's adaptability in different contexts. You are also encouraged to identify more aspects that reflect diversity. We are looking forward to your insights!

- Diversity of thought improves reasoning abilities of large language models (Naik et al., 2023)
- PAL: Program-aided Language Models (Gao et al., 2023)

Hint: Consider how different levels of diversity in prompts might affect the LLM's reasoning and coding ability. You may want to explore how varying the prompts can lead to more robust and generalized performance.

2.5 Generalization (Optional)

Congratulations on completing your analysis of LLM reasoning and coding capabilities! Until now, your experiments have likely focused on GSM8K and HumanEval, as in A1. Would your methods and analysis change when applied to other datasets?

If you find the previous tasks not challenging enough, you can choose 1-2 additional datasets from the lists below, repeat your experiments, and report your observations. See if your methods or conclusions generalize well to these new datasets.

- Math: e.g., MultiArith (Roy and Roth, 2015), AQuA (Ling et al., 2017), GSM-Hard (Gao et al., 2023)[1], GSM-Plus (Li et al., 2024), a list available at: here for reference.
- Coding: e.g., MBPP (Austin et al., 2021), APPS (Hendrycks et al., 2021), HumanEval-X (Zheng et al., 2023), a list available at: here for reference.

3 Model and API

Similarly, you may use the Qwen series of models, which is a powerful open-source model that natively supports multilingual capabilities, coding, reasoning, and tool usage.

In this assignment, you can use any LLM model to complete the tasks. You can get free token credits (Note: Your free tokens are limited. If youencounter API calling errors, check whether you've exceeded your quota.) on the Alibaba Cloud Bailian platform:https://www.alibabacloud.com/en?_p_lc=1/(You can find the guide of Bailian platformhere, (CN) andhereis an example API tutorial to help youget started). Alternatively, you're welcome to purchase your own tokens from providers like OpenAI, DeepSeek, Grok, Gemini, or Claude tocomplete the assignment. You can find API usage guides for these LLM providers on their official websites.

4 Report

You will write a report including the following parts:

- The description of your implemented analytical methods, including the experimental settings, the hyperparameters, etc.
- The **outcomes** and **discussion** of your analysis, such as the prompts you used, the carefully designed demonstrations, and some appropriate statistics and studies.

5 Gadgets

The following resources might help you with this assignment:

- A repository containing Chain of Thought and related papers: Chain-of-ThoughtsPapers.
- A repository with a wealth of code generation work: Awesome-Code-Intelligence.

[1]Code-based solution is preferred for GSM-Hard (Gao et al., 2023)

6 Note

There are some key points you should pay attention to:

- Your assignment will not be evaluated solely based on your experimental results (e.g., task accuracy). As an analytical assignment, **we are more interested in seeing your thought process and creativity in experimental design and your report**. We highly recommend visualizing your experimental results.
- Considering the complexity of task design and the richness of existing research, coding will be more challenging to analyze than math reasoning. Don't worry; we will take task difficulty into account during grading.
- We have observed that some students in A1 used program-aided language models (Gao et al., 2023) to tackle math reasoning. This is excellent! You can try cross-analyzing LLM reasoning and coding. Some relevant literature is available here for reference.
- The papers listed in this document are for reference purposes only. You are not required to follow them for expansion or replication of results.
- (Optional) Beyond `Llama-3.1-8B-Instruct`, you can explore other available models for this assignment. Feel free to modify decoding parameters like temperature.

References

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. 2021. Program synthesis with large language models. https://arxiv.org/abs/2108.07732.

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

Yao Fu, Hao Peng, Ashish Sabharwal, Peter Clark, and Tushar Khot. 2023. Complexity-based prompting for multi-step reasoning. In *The Eleventh International Conference on Learning Representations*. https://openreview.net/forum?id=yf1icZHC-l9.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. Pal: program-aided language models. In *Proceedings of the 40th International Conference on Machine Learning*. JMLR.org, ICML'23.

Dan Hendrycks, Steven Basart, Saurav Kadavath, Mantas Mazeika, Akul Arora, Ethan Guo, Collin Burns, Samir Puranik, Horace He, Dawn Song, and Jacob Steinhardt. 2021. Measuring coding challenge competence with apps. In J. Vanschoren and S. Yeung, editors, *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*. volume 1.

Qintong Li, Leyang Cui, Xueliang Zhao, Lingpeng Kong, and Wei Bi. 2024. GSM-plus: A comprehensive benchmark for evaluating the robustness of LLMs as mathematical problem solvers. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Bangkok, Thailand, pages 2961–2984. https://doi.org/10.18653/v1/2024.acl-long.163.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In Regina Barzilay and Min-Yen Kan, editors, *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*. Association for Computational Linguistics, pages 158–167. https://doi.org/10.18653/v1/P17-1015.

Pan Lu, Liang Qiu, Wenhao Yu, Sean Welleck, and Kai-Wei Chang. 2023. A survey of deep learning for mathematical reasoning. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Toronto, Canada, pages 14605–14631. https://doi.org/10.18653/v1/2023.acl-long.817.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. 2023. Self-refine: Iterative refinement with self-feedback. In *Thirty-seventh Conference on Neural Information Processing Systems*. https://openreview.net/forum?id=S37hOerQLB.

Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2022. Rethinking the role of demonstrations: What makes in-context learning work? In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang, editors, *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Abu Dhabi, United Arab Emirates, pages 11048–11064. https://doi.org/10.18653/v1/2022.emnlp-main.759.

Ranjita Naik, Varun Chandrasekaran, Mert Yuksekgonul, Hamid Palangi, and Besmira Nushi. 2023. Diversity of thought improves reasoning abilities of large language models. *arXiv preprint arXiv:2310.07088* .

Jane Pan, Tianyu Gao, Howard Chen, and Danqi Chen. 2023. What in-context learning "learns" in-context: Disentangling task recognition and task learning. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, *Findings of the Association for Computational Linguistics: ACL 2023*. Association for Computational Linguistics, Toronto, Canada, pages 8298–8319. https://doi.org/10.18653/v1/2023.findings-acl.527.

Subhro Roy and Dan Roth. 2015. Solving general arithmetic word problems. In Lluís Màrquez, Chris Callison-Burch, Jian Su, Daniele Pighin, and Yuval Marton, editors, *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*. The Association for Computational Linguistics, pages 1743–1752. https://doi.org/10.18653/v1/d15-1202.

Qiushi Sun, Zhirui Chen, Fangzhi Xu, Kanzhi Cheng, Chang Ma, Zhangyue Yin, Jianing Wang, Chengcheng Han, Renyu Zhu, Shuai Yuan, Qipeng Guo, Xipeng Qiu, Pengcheng Yin, Xiaoli Li, Fei Yuan, Lingpeng Kong, Xiang Li, and Zhiyong Wu. 2024. A survey of neural code intelligence: Paradigms, advances and beyond. *arXiv preprint arXiv:2403.14734* https://doi.org/10.48550/arXiv.2403.14734.

Boshi Wang, Sewon Min, Xiang Deng, Jiaming Shen, You Wu, Luke Zettlemoyer, and Huan Sun. 2023. Towards understanding chain-of-thought prompting: An empirical study of what matters. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Toronto, Canada, pages 2717–2739. https://doi.org/10.18653/v1/2023.acl-long.153.

Qinkai Zheng, Xiao Xia, Xu Zou, Yuxiao Dong, Shan Wang, Yufei Xue, Zihan Wang, Lei Shen, Andi Wang, Yang Li, Teng Su, Zhilin Yang, and Jie Tang. 2023. Codegeex: A pre-trained model for code generation with multilingual benchmarking on humaneval-x. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. pages 5673–5684.