Project Proposal: The Practicality of Prompt Engineering

Anonymous ACL submission

Abstract

This paper examines the practicality of prompt engineering in improving the performance of Large Language Models (LLMs). Through empirical analysis, we evaluate the trade-offs between costs and benefits of prompting using novel metrics. Different prompting methods are assessed using standardized tasks and both modern and older models.

Introduction

Prompt engineering, the practice of developing specialized prompts and queries to improve the accuracy of Large Language Models after training, is a prominent topic of interest in the NLP community, and among the general public. The practice is believed to allow for improvements in LLM performance a variety of domains without investment in underlying training (Martineau, 2021). It is not, however, without its critics. Some commentators believe that the practice will become irrelevant as models grow larger and more powerful, becoming more capable of directly interpreting a user's intent. (Ethan Mollick [@emollick], 2023). Others question the need for specialized professionals or traiing to attain minimal improvements which are often not repeatable across domain types and contexts (Shackell, 2023; Acar, 2023).

Despite such controversy, it is difficult to find empirical and quantitative analyses of the trade-off between costs and accuracy benefits associated with advanced prompting. Papers introducing new prompting techniques often only include performance benchmarks concerning the techniques' efficacy, typically within a limited question domain. Some authors briefly mention problems associated with human-tailored problems, such as the increased complexity induced by prompt-chaining and limitations on creativity and randomness (Wu et al., 2022), and others suggest the automation of prompting (Diao et al., 2023). Online marketplaces such as Promptbase (noa) provide input token costs for prompt texts sold on the platform, but do not provide any other information.

This paper uses several metrics to evaluate the costs of prompt engineering methods systematically, and analyzes the tradeoffs inherent in their application to standardized data. Such an assessment is valuable on several dimensions. Beyond quantifiably testing the practicality of prompt engineering as a whole, it can be used to compare the performance of different approaches, useful in a world where so many competing techniques are available. I also offer a new look at these methods in a period long after ideas were introduced and in an enviroment with greater capabilities in underlying models. Finally, I survey and introduce some useful measures of costs and complexity such as the ratio of interaction length with prompting to the length of an accepted human-generated answer.

Data

To evaluate the LLMs, I attempt to use tasks that are both general-purpose and close to practical, real-world applications. In this spirit, I use GRE General Test questions (likely from a purchased prep book/or practice exams recently published online to minimize contamination), as well as HumanEval coding problems. For multiple choice choice questions, I add formatting to the prompt to indicate what should be done to represent a final answer "When you are ready, please put your final, letter answer in the form of a single capital letter, enclosed in parentheses. For example, if you think the answer is A, please write (A)." HumanEval uses the pass at k metric to automatically assess the probability a solution is correct given a set of unit tests. (Chen et al., 2021)

I perform the analysis on one cutting-edge model, to get the current state of things, and one older model, closer to the time that these techniques were introduced. This will allow one picture of the changing costs and benefits of advanced prompting, a trend that may even be extrapolated into the future if current LLM scaling laws continue to hold. As the most widely used models and the ones behind much original work in the field, I select two models from the OpenAI series: GPT-4 and text-davinci-003 on OpenAI playground. Should text-davinci-003 (a legacy model) become unavailable during the course of the project, or should I encounter other difficulties, I will use the simplest/smallest model available, likely GPT-3.5 (something to note is that models older than GPT-3.5 have, in the past, scored 0% for accuracy on coding problems - I will need to test all of my evaluations quickly to get a sense of their feasibility before scaling up). All of these models are available both via the OpenAI API and in web interfaces - where possible I will use web interfaces to limit resources required.

To the extent possible, I will report accuracy scores on the domain dataset as they are in the original paper. It may also be possible to use any prompts, LLM responses, and correct responses provided along with original papers to calculate other simple metrics such as response lengths and complexity. However, some metrics (time taken, human assessments of complexity) will require my own evaluations.

Metrics

Accuracy

Improved accuracy can be a benefit of prompt engineering. I report:

- Correct/Incorrect accuracy at the point a technique has been fully implemented (the end of the chain of thought, etc., modelling the real world)

Length

The length of responses and interactions can effect the practicality of prompting. It could indicate that a model is carefully and correctly solving through the steps of a problem. It can also impose time and financial costs to users, or become an indicator of degraded performance as models sometimes tend to go off on tangents or repeat themselves (to the extent some platforms have imposed length limitations) (Mann). I report:

- Length of the entire interaction in tokens
- Financial cost of the entire interaction in tokens
- Length of the entire interaction in tokens relative to the length of the baseline task + a human/solved out/generally accepted as correct answer. How much is the engineering stretching the interaction out? Is this stretching adding value/improving accuracy?
- Length of the entire interaction in time (seconds). This can include time writing a response, waiting for a response, or reviewing a response - this level of granular data may be hard to collect, but it might be possible to look at human assessments of time spent on these activities. Attempts will be made to have queries to models made during off-peak hours to minimize confounding due to server load, connectivity issues, etc.

Complexity

Similarly to length, complexity could be an indicator of high accuracy. However, it has substantial costs in potentially making review more difficult. I report:

- Vocabulary - share of words on the Academic Vocabulary List (AWL). (Gardner and Davies, 2014)
- Sentence length and Flesch reading ease (implemented via the textstat Python package) for natural language and non-code components of responses. (Flesch, 2016; Aggarwal)
- Cyclomatic complexity for code responses (implemented via the radon Python package). (Lacchia)
- Ratio or difference of these scores in prompts vs. responses, responses vs. accepted/outside correct answer
- Human assessment of need for specialized knowledge/difficulty of implementation of the technique. This could be task specific (done for some novel real world example questions/prompting scenarios), or it could be done overall based on a pre-existing description of the technique. Perhaps a balance of both is best. The actual metric will be a numerical score and a qualitative description.
- Human assessment of output complexity (ease of evaluating results). This could be task specific (done for some novel real world example

questions/prompting scenarios), or it could be done overall based on a pre-existing examples of the technique. Perhaps a balance of both is best. The actual metric will be a numerical score and a qualitative description.

Prompting Methods Assessed

This list is subject to change, pending further assessment of implementation difficulty and potential accuracy gains for each method. Higher accuracy methods are likely to be more interesting.

Below items are listed in order of increasing complexity/human intervention:

- Zero-Shot Control Baseline: Providing the question/task directly. (Sometimes called "Direct Prompting")

- Zero-Shot Chain of Thought Prompting: Existing literature mentions several examples of this: a simple one to append to every initial question/task, found to be optimal through automated testing is "Let's work this out in a step by step way to be sure we have the right answer." (Hebenstreit et al., 2023; Zhou et al., 2022)

- Tree of Thought Prompting: This prepends the following to the task/question: "Imagine three different experts are answering this question. All experts will write down 1 step of their thinking, then share it with the group. Then all experts will go on to the next step, etc. If any expert realises they're wrong at any point then they leave. The question is..." (Hulbert, 2023)

- Chain of Verification Prompting: The LLM is prompted a series of times to produce an initial baseline response, write its own verification questions, answer those verification questions, and write a final, verified response. (Dhuliawala et al., 2023)

- Few-Shot Prompting: The prompter provides a few examples of successfully/answered questions or tasks before the main question/task. Despite the work involved in implementing this method, I believe it has potential to be effective for two reasons. First, prior evidence indicates that larger and more modern language models benefit more from few-shot learning, potentially making this a consistently useful technique. (Brown et al., 2020) Second, earlier research has found that the formatting and input/label space distribution is more important than example correctness, meaning this method is somewhat robust to human error. (Min et al., 2022)

Analyses

I provide summary statistics of the metrics for each prompting method by model. In cases where human/textual assessment and comments have been provided, it might be interesting to use NLP methods to evaluate responses (ex: for sentiment).

Responsiblities

I am the sole author of this proposal, but I am open to working with others with similar interests. Expanding the group would be helpful in order to improve the project.

Limitations

It was difficult to select prompt engineering methods to try for this paper, and there is potential for my choice of methods to be somewhat biased. I mostly picked methods based my perception of their popularity and on their ease of implementation. If anything, this may lead to an underestimation of costs.

Just as my evaluation comes at a time with significantly more capable LLMs relative to that where much work began on prompting, I expect the underlying calculus concerning prompting to continue to change in the future. However, I again only expect relative costs of complex engineering to increase as models get better.

Another potential problem is the extent that prompting techniques have been absorbed into default LLM behavior, likely through reinforcement learning. GPT-4 in particular does seem to implement chain-of-thought methods when prompted with a sufficiently complex problem. In this environment, this paper become less of an evaluation of prompting techniques themselves, but more of an evaluation of their manual implementation.

Finally, though I have taken steps to limit it, data contamination remains a real concern. The evaluations I use may have been used to train the LLMs, or the questions/tasks from them may have been introduced through reinforcement learning based on other evaluations. On the other hand, this seems unlikely to bias the results for any one particular prompting method relative to the others - comparisons between them are still likely to be useful.

Acknowledgements

The template for this document was adapted by Jordan Boyd-Graber, Naoaki Okazaki, and Anna Rogers.

References

PromptBase.

Oguz A. Acar. 2023. AI Prompt Engineering Isn't the Future. *Harvard Business Review*. Section: Technology and analytics.

Shivam Bansal Aggarwal, Chaitanya. textstat: Calculate statistical features from text.

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. ArXiv:2005.14165 [cs].

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. 2021. Evaluating Large Language Models Trained on Code. ArXiv:2107.03374 [cs] version: 2.

Shehzaad Dhuliawala, Mojtaba Komeili, Jing Xu, Roberta Raileanu, Xian Li, Asli Celikyilmaz, and Jason Weston. 2023. Chain-of-Verification Reduces Hallucination in Large Language Models. ArXiv:2309.11495 [cs].

Shizhe Diao, Pengcheng Wang, Yong Lin, and Tong Zhang. 2023. Active Prompting with Chain-of-Thought for Large Language Models. ArXiv:2302.12246 [cs].

Ethan Mollick [@emollick]. 2023. I have a strong suspicion that "prompt engineering" is not going to be a big deal in the long-term & prompt engineer is not the job of the future AI gets easier. You can already see in Midjourney how basic prompts went from complex in v3 to easy in v4. Same with ChatGPT to Bing. https://t.co/BTtSN4oVF4.

Rudolf Flesch. 2016. How to Write Plain English.

Dee Gardner and Mark Davies. 2014. A New Academic Vocabulary List. *Applied Linguistics*, 35(3):305–327.

Konstantin Hebenstreit, Robert Praas, Louis P. Kiesewetter, and Matthias Samwald. 2023. An automatically discovered chain-of-thought prompt generalizes to novel models and datasets. ArXiv:2305.02897 [cs].

Dave Hulbert. 2023. Using Tree-of-Thought Prompting to boost ChatGPT's reasoning. Original-date: 2023-05-22T19:03:27Z.

Michele Lacchia. radon: Code Metrics in Python.

Jyoti Mann. Microsoft limits Bing chat exchanges and conversation lengths after 'creepy' interactions with some users.

Kim Martineau. 2021. What is prompt tuning?

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? ArXiv:2202.12837 [cs].

Cameron Shackell. 2023. Prompt engineering: is being an AI 'whisperer' the job of the future or a short-lived fad?

Tongshuang Wu, Michael Terry, and Carrie Jun Cai. 2022. AI Chains: Transparent and Controllable Human-AI Interaction by Chaining Large Language Model Prompts. In *CHI Conference on Human Factors in Computing Systems*, pages 1–22, New Orleans LA USA. ACM.

Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. 2022. Large Language Models are Human-Level Prompt Engineers.