University *of Ljubljana*
Faculty *of Computer and Information Science*

LLM Prompt Strategies for Commonsense-Reasoning Tasks

Jaša Samec, Haris Kupinić, and Jovana Videnović

Abstract

Keywords
Keyword1, Keyword2, Keyword3 ...

Advisor: Assist. Aleš Žagar

Introduction

Due to their ability to generate human-like text and perform well on a variety of tasks, large language models, *(LLMs)*, have become increasingly popular topics in recent periods. Nowadays, they are almost a daily tool for not just researchers, but millions of people. However, there are still some tasks that LLMs struggle with, such as simple math and logic, and commonsense reasoning. This is because LLMs are trained on large corpora of text and are not explicitly taught to reason about the world. To improve their performance on these tasks, well-constructed prompts are effective. In this project, we explore different prompt strategies for improving the performance of LLMs on commonsense-reasoning tasks, including Chain of Thought (CoT) [1], in-context learning, and Plan-and-Solve (PS) [2]. One example of prompting is shown in Figure 1. The effectiveness of these techniques is evaluated on commonsense reasoning datasets; including parts of CommonsenseQA [3] and BIG-Bench [4] datasets. The model we employ is Llama 2 [5].

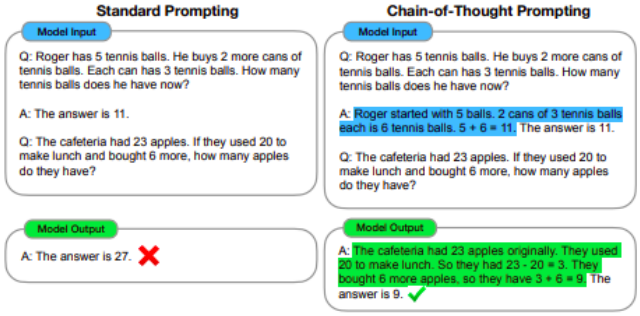

Figure 1. Example of Chain of Thought approach, taken from [1]. By splitting the problem into smaller subsets, model obtains the right result.

Related work

To address the challenges that LLMs face in dealing with mathematical and logical reasoning tasks [6], researchers have suggested the use of prompts. Well-designed prompts have demonstrated to notably enhance the performance of LLMs, sometimes achieving results comparable to explicit fine-tuning [7]. One fundamental prompting technique is *in-context learning*, wherein the model is provided with example inputs and outputs before the desired output (e.g., supplying the model with example translations).

Another early technique in prompt engineering is Chain-of-Thought (CoT) prompting [1]. This method adds a triple of the form *(input, chain of thought, output)*, with the hope that when the model gets a new input (a question), it will follow it up with a chain of thought and then the output. The example CoT that is passed to the LLM breaks down the problem into smaller steps, making it easier to solve. This approach offers two benefits: first, the model can allocate more time/steps to the more difficult aspects of the problem. Second, it gives the user some insight into the model's reasoning process.

Furthermore, from the Chain-of-Thought prompting, researchers began to develop zero-shot prompting strategies. The most known are zero-shot CoT [8] and Plan-and-Solve [9]. Both of these methods guide the models response by asking it to think in smaller steps. Zero-shot CoT just starts the response by adding "Let's think step by step." While Plan-and-solve first guides the model to generate a list of possible steps and then asks it to follow them.

Later approaches such as Tree-of-Thought [10] and Graph-of-Thought [11] also break down the problem into smaller steps in different ways. For example, Tree-of-Thought asks the model to generate possible steps and then evaluates them. If a step is evaluated to be correct it builds upon it, otherwise the method backtracks through the previously chosen

steps and generates new steps. Using this method the model generates a tree of logical steps that lead to the solution.

However, newer methods such as Promptbreeder [12] have already surpassed the performance of some hand-crafted prompting techniques. This method combines the original prompt with different *thinking styles* (e.g., "How could I measure progress on this problem?") and *mutator prompts* (e.g., "How would you help an LLM to follow the instruction?") to generate new prompts by passing them to an LLM. Then, it scores the fitness of the generated prompts and iteratively mutates the best ones over multiple generations to evolve prompts specifically tailored to the task.

Methods

To perform evaluation of both baselines methods, as well as Promptbreeder [12], we developed a prompting framework that iterates through the given dataset and for each question, generates a prompt based on the provided strategy that we want to test. As baselines strategies, we opted for *Zero-shot CoT*, *CoT*, and *Plan-and-Solve*. In the pipeline, we also include testing when no prompting technique was used.

Raw prompt

Raw prompt technique is the simplest possible approach of prompting as we do not provide the model with any additional instruction how to create the answer. The user's prompt is directly fed into the model in its raw form.

Zero-shot Chain-of-Thought

Zero-shot CoT strategy includes very basic instruction for the model and, as the name suggests, provides no examples of how should model answer. Besides the original prompt, we include following instruction *"Let's think step by step."* as the beginning of the model's answer. Compared to the naive approach when no instructions are given, we expect this method to allow model to break the problem into smaller subproblems, and helps it define the answer more easily.

Chain-of-Thought

This method builds upon Zero-shot CoT. Besides helping the model to break problem into smaller chunks, it also augments the prompt with some examples, which makes it easier for model to follow.

In this project, we include one example into the prompt. It is being added before the original prompt, so the model can use it as a "template" which it should follow. Our example prompt is: "*[INST]* You are an expert problem solver. In the input you have two questions. One has a correct response and the other has no response. Your job is to answer the question that has no response. Q1: The only baggage the woman checked was a drawstring bag, where was she heading with it? Answer Choices: (A) garbage can (B) military (C) jewelry store (D) safe (E) airport*[/INST]*. A1: Jewelry stores and safes are not typical destinations for someone with only a drawstring bag. Military personnel typically have significantly more luggage when deployed and garbage cans are not a destination. Airports are the most likely destination for someone with only a drawstring bag as checked luggage. Answer: E." After the example prompt, original prompt is added within new **[INST]** tags. These tags are used to help the model distinguish between different questions within the prompt.

Plan-and-Solve

Plan-and-Solve is the last strategy we will use as a baseline. It works in a similar fashion as zero-shot CoT, as it helps the model to break problem into smaller steps and follow them. As in the CoT example, on top of the original prompt, we also add an additional instruction for the model. For the Plan-and-Solve strategy, we include following instruction: *"Let's first understand the problem, extract relevant variables and their corresponding numerals, and devise a complete plan. Then, let's carry out the plan, calculate intermediate variables (pay attention to correct numerical calculation and commonsense), solve the problem step by step, and show the answer."*

LLM based prompt generation

Inspired by Promptbreeder, we propose two new methods that leverage large language models to generate new prompts. Our goal was to create a method that can use LLMs for generating prompts without significant overhead, unlike Promptbreeder which employs genetic algorithms to score the final prompts.

Our first method involves scoring mutator prompts (taken from Promptbreeder) during training. Then, at test time, it randomly selects one of the more successful mutations. This method aims to automatically discover the most effective approaches for a given problem domain.

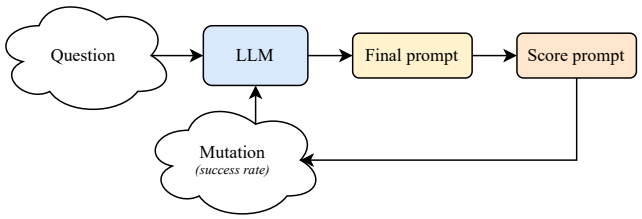

Figure 2. Pipeline for prompt generation with mutating prompts.

The second method utilizes a two-layer approach. The first layer generates N prompts, with parameters that can be set to encourage creative exploration (e.g., increasing temperature). This aims to produce a diverse set of approaches. The second layer then combines these generated outputs into a single prompt. Ideally, the second layer can effectively identify and integrate the most promising ideas to address the problem.

Evaluation

Result of our framework for every method is a dictionary, which among others, includes also the final model's prediction as well as the true prediction. As the CommonSenseQA [3] dataset is based on multiple choices in form of *A, B, C, D*, the final/true predictions are just strings that represent one

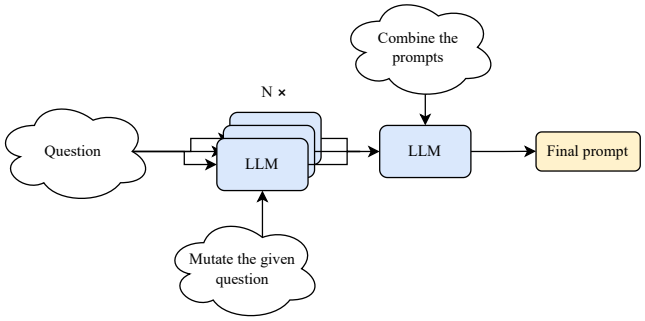

Figure 3. Pipeline for two-layer approach prompt generator.

Raw prompt: "After he got hired, he hoped for success in his vocation."

Zero-shot CoT: "Let's think step by step. After he got hired, he hoped for success at his...(A) vocation - This is the correct answer! After getting hired, he hoped for success at his vocation or profession. So, the answer is (A) vocation."

Chain-of-Thought: "The answer is (B) new job. The sentence suggests that the person hoped for success in their new job after getting hired."

Plan-and-Solve: "Let's first understand the problem, extract relevant variables and their corresponding numerals, and devise a complete plan. Then, let's carry out the plan, calculate intermediate variables (pay attention to correct numerical calculation and commonsense), solve the problem step by step, and show the answer. Finally, let's verify the answer and ensure it is correct. So, after he got hired, he hoped for success at his (A) vocation."

Table 1. Model's answers with different strategies to question: *After he got hired he hoped for success at his what? Answer Choices: (A) vocation (B) new job (C) michigan (D) working hard (E) manual"*. Correct answer is (B).

of the above mentioned choice. To evaluate the model, we use solve rate metric, which measures number of times the model's prediction coincides with the true value.

Results

Each method is tested on 1221 instances from the CommonSenseQA [3] dataset. Results for each method are shown in Figure 4.

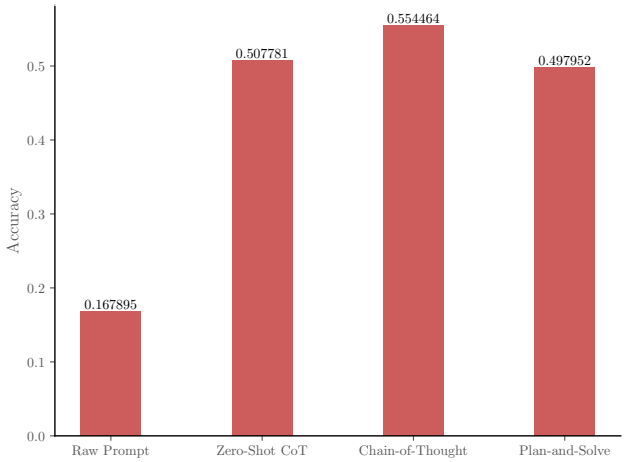

Figure 4. Classification accuracy for baseline methods.

There are not any significant differences between the the *Zero-shot CoT* and *Plan-and-Solve* techniques. This is somewhat expected, as both methods are zero-shot, meaning that the model does not know how to actually deduce the answer. However, including even basic instruction for the model, allows it to increase its performance almost by half. The best approach seems to be the *Chain-of-Thought*. Model is shown an example of expected way of thinking which allows it to create better response.

Discussion

Acknowledgments

References

[1] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 24824–24837. Curran Associates, Inc., 2022.

[2] Lei Wang, Wanyu Xu, Yihuai Lan, Zhiqiang Hu, Yunshi Lan, Roy Ka-Wei Lee, and Ee-Peng Lim. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2609–2634, Toronto, Canada, July 2023. Association for Computational Linguistics.

[3] Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North*

American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pages 4149–4158, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[4] BIG bench authors. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Transactions on Machine Learning Research*, 2023.

[5] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models, 2023.

[6] Jack W. Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, H. Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, Eliza Rutherford, Tom Hennigan, Jacob Menick, Albin Cassirer, Richard Powell, George van den Driessche, Lisa Anne Hendricks, Maribeth Rauh, Po-Sen Huang, Amelia Glaese, Johannes Welbl, Sumanth Dathathri, Saffron Huang, Jonathan Uesato, John Mellor, Irina Higgins, Antonia Creswell, Nat McAleese, Amy Wu, Erich Elsen, Siddhant M. Jayakumar, Elena Buchatskaya, David Budden, Esme Sutherland, Karen Simonyan, Michela Paganini, Laurent Sifre, Lena Martens, Xiang Lorraine Li, Adhiguna Kuncoro, Aida Nematzadeh, Elena Gribovskaya, Domenic Donato, Angeliki Lazaridou, Arthur Mensch, Jean-Baptiste Lespiau, Maria Tsimpoukelli, Nikolai Grigorev, Doug Fritz, Thibault Sottiaux, Mantas Pajarskas, Toby Pohlen, Zhitao Gong, Daniel Toyama, Cyprien de Masson d'Autume, Yujia Li, Tayfun Terzi, Vladimir Mikulik, Igor Babuschkin, Aidan Clark, Diego de Las Casas, Aurelia Guy, Chris Jones, James Bradbury, Matthew J. Johnson, Blake A. Hechtman, Laura Weidinger, Iason Gabriel, William Isaac, Edward Lockhart, Simon Osindero, Laura Rimell, Chris Dyer, Oriol Vinyals, Kareem Ayoub, Jeff Stanway, Lorrayne Bennett, Demis Hassabis, Koray Kavukcuoglu, and Geoffrey Irving. Scaling language models: Methods, analysis & insights from training gopher. *CoRR*, abs/2112.11446, 2021.

[7] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D 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 Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020.

[8] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners, 2023.

[9] Lei Wang, Wanyu Xu, Yihuai Lan, Zhiqiang Hu, Yunshi Lan, Roy Ka-Wei Lee, and Ee-Peng Lim. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models, 2023.

[10] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In A. Oh, T. Neumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 11809–11822. Curran Associates, Inc., 2023.

[11] Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and Torsten Hoefler. Graph of thoughts: Solving elaborate problems with large language models, 2024.

[12] Chrisantha Fernando, Dylan Banarse, Henryk Michalewski, Simon Osindero, and Tim Rocktäschel. Promptbreeder: Self-referential self-improvement via prompt evolution, 2023.