FRI Natural language processing course 2024
Project report

University of Ljubljana
Faculty of Computer and Information Science

LLM Prompt Strategies for Commonsense-Reasoning Tasks

Haris Kupinić, Jaša Samec, and Jovana Videnović

Abstract

This report explores various prompt strategies for commonsense reasoning tasks using large language models (LLMs). Of the seven methods evaluated, four were adapted from previous research, while we introduced three new LLM-based approaches. Among all of these, the Chain-of-Thought (CoT) method delivered the best performance. Within our proposed methods, only the one where the LLM generates argumentation for multiple-choice questions achieved results on par with CoT. Overall, our findings indicate that prompting techniques require further refinement to be effective with smaller LLMs on commonsense datasets. The conclusion provides recommendations for future research in this area.

Keywords

Prompt engineering, Large language models, Question answering

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 existing prompt strategies for improving the performance of LLMs on commonsense-reasoning tasks. One example of CO [1] prompting is shown in Figure 1. Furthermore, we propose new prompting techniques which are based on LLMs. This work is inspired by Promptbreeder [2] paper. The effectiveness of these techniques is evaluated on commonsense reasoning datasets; including CommonsenseQA [3] and ProtoQA [4]. The model we employ is Llama 2 [5].

To summarize, our contributions to this work are the following:

- Review of the current prompting techniques, including Chain of Thought (CoT),
- Design of three LLM-based prompting techniques. More specifically, we propose three new methods: Mutation-based prompting, Two-LLMs prompting, and Self-argumentation prompting.

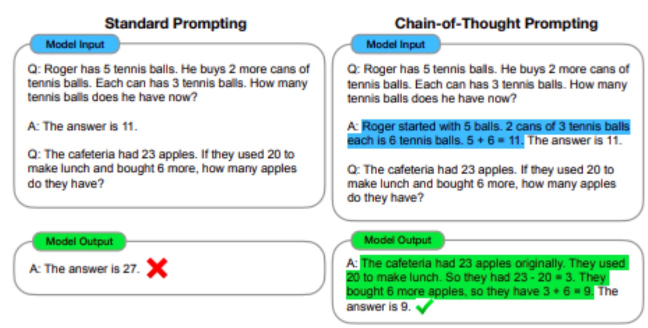

Figure 1. Example of Chain of Thought approach, taken from [1]. By splitting the problem into smaller subsets, the model obtains the right result.

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 been 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.

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 model's response by asking it to think in smaller steps. Zero-shot CoT just starts the response by adding *Let's think step by step*. Plan-and-solve first guides the model to generate a list of possible steps and then ask 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 [2] 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

In the following section, we present the baseline methods we experimented with, Zero-shot CoT, CoT, and Plan-and-Solve techniques. Subsequently, we describe our three proposed LLM-based methods. The section is divided into two subsections, corresponding to existing and proposed prompting methods.

Existing prompting techniques

In this part, we aim to explain the existing prompting engineering methods, we used in our work.

Raw prompt

The raw prompt technique is the simplest possible approach to prompting as we do not provide the model with any additional instruction on 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 instructions for the model. As the name suggests, it provides no examples of how the should model answer. Besides the original prompt, we include the following instruction, at the beginning of the model's answer:

"Let's think step by step"

Compared to the naive approach when no instructions are given, we expect this method to allow the model to break the problem into smaller subproblems, and help it define the answer more easily.

Chain-of-Thought

This method, besides helping the model to break the problem into smaller chunks, also augments the prompt with some examples, which makes it easier for the model to follow.

In this project, we include one example in 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 shown in the following:

"[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, the original prompt is added within new **[INST]** tags. These tags are used to help the model distinguish between different questions within the prompt. Similar formatting is done with other prompt strategies.

Plan-and-Solve

Plan-and-Solve is the last strategy we use as a baseline. It works similarly to zero-shot CoT, as it helps the model break problems into smaller steps and follow them. As in the CoT example, on top of the original prompt, we also add instructions for the model. For the Plan-and-Solve strategy, we include the following instructions:

"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 three 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.

Mutation-based prompting

Our first method involves taking mutator prompts or thinking styles (both taken from Promptbreeder, with some selection) This method aims to automatically discover the most effective approaches for a given problem domain. The pipeline for the approach is visualized in Figure 2.

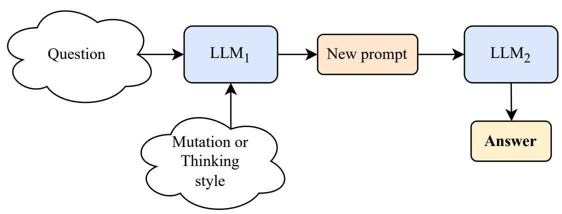

Figure 2. Prompt generation with mutating prompts. Together with the question, LLM receives the mutation.

Two-LLMs prompting

The second method is a two-LLM approach. The first LLM takes in the question and a prompt to construct arguments for the possible answers. The key here is that the first LLM has a higher temperature making the output more creative with the hopes it can find a possible solution a lower temperature LLM could not. The arguments are then passed to the second LLM (with a lower temperature) is asked to answer the question but it can use the arguments as help. The pipeline for the approach is visualized in Figure 3.

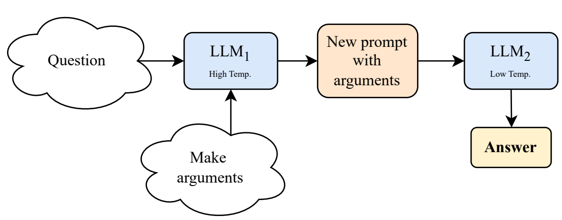

Figure 3. Two-layer prompt generator. One LLM generates the prompts, the second one identifies the best ideas for the question.

Self-argumentation prompting

The third method is quite similar to the second one, with the main difference in the way prompts are generated. For this method, a single LLM is responsible for creating arguments, as well as determining the final answer. We may expect that the reasoning about each answer may bring us closer to the right answer. This method aims to craft LLM prompts specifically designed for answering multiple-choice questions.

Results

In the first part of this section, we explain the foundations of our evaluation pipeline, including datasets, models, and metrics employed. In the continuation, we analyze the results obtained.

Evaluation details

This section is organized into different parts, which all explain important aspects of our evaluation methodology. For this work, we designed a prompting framework, which supports different models and commonsense datasets. Regarding the question-answering, we cover multi-choice and open-ended variants. For both QA cases, we explain how we evaluate LLM's performance.

Prompting framework

To evaluate the prompting methods, 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.

Regarding the models, the framework supports models Llama 2. In this work, we employed Llama models with 7 and 13 billion parameters. The framework supports CommonsenseQA, StrategyQA, and ProtoQA commonsense datasets.

Datasets

We test the prompting strategies on CommonsenseQA and *ProtoQA* datasets. A brief explanation of both is presented in the following.

CommonsenseQA: CommonsenseQA is a dataset designed for evaluating the understanding of common sense knowledge. It consists of multiple-choice questions that require an understanding of everyday concepts and their relationships. The correct answer must be chosen from several plausible options. The dataset tests the ability of models to leverage implicit human knowledge. Each method is tested on all questions from the CommonSenseQA dataset.

ProtoQA: ProtoQA is a dataset focused on evaluating a model's ability to generate answers to questions that require prototypical knowledge. The focus is on open-ended question answering. Questions in ProtoQA are designed to require answers that reflect general, prototypical concepts rather than specific facts. The datasets test the model's ability to produce diverse and contextually appropriate answers.

These datasets provide complementary challenges: CommonsenseQA focuses on selecting the best answer from given choices, while ProtoQA evaluates the generation of suitable responses to open-ended questions.

Metrics

To evaluate the model, we use the solve rate metric. It calculates the proportion of instances where the model's output matches the correct output.

CommonSenseQA: As this dataset is based on multiple choices in the form of *A, B, C, D*, final and correct outputs are just strings that represent one of the above-mentioned choices. To evaluate the model, we use the solve rate metric, which measures the number of times the model's prediction coincides with the true value.

ProtoQA: The answers for this dataset do not follow any form, so rather than evaluating the model automatically, we manually review the model's answers. We determine its quality, which is based on the correctness of the response, its creativity, as well as number of answers provided.

Mutliple-choice QA evaluation

In this section, we aim to present the performance evaluation, got with the previously explained evaluation framework. The results are obtained with the LLama2 model with 7 billion parameters, evaluated on the CommonsenseQA dataset.

Existing methods

In Figure 4, we present the performance evaluation, using solve rate metric, with the existing prompt engineering methods. We can observe that prompting greatly improves the results.

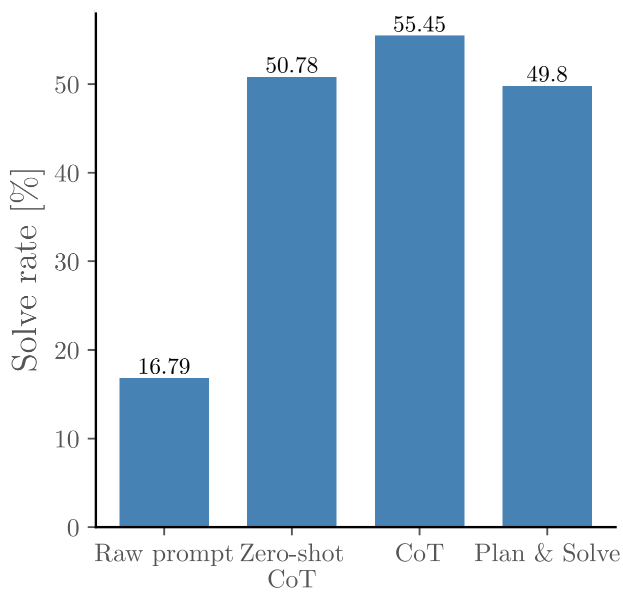

Figure 4. Solve rate for existing prompting methods. CoT outperforms every other method. There is a significant difference when prompting methods are used, compared to raw prompts.

There are no significant differences between 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 deduce the answer. However, even with these basic instructions for the model, the performance increased by over 30%.

The best performance approach is a chain of thought. As the model is shown as an example of an expected way of thinking, it allows it to create better responses. Compared to zero-shot methods, the performance is increased by around 5%.

Overall, we would say that even the best CoT, has performed worse than expected. With the solve rate around 55%, we would say that the technique probably is not effective enough on the smaller LLMs.

Some examples of the LLM's answers after prompting are shown in Table 2 in Appendix. We can see that CoT and LLM-based prompts provided the right answer.

Our methods

In Figure 5, we present the results for the proposed LLM-based methods. All LLM-based methods performed worse than the CoT method. Mutation-based and Two-LLMs based prompting perform worse than zero-shot CoT, and Plan and Solve methods.

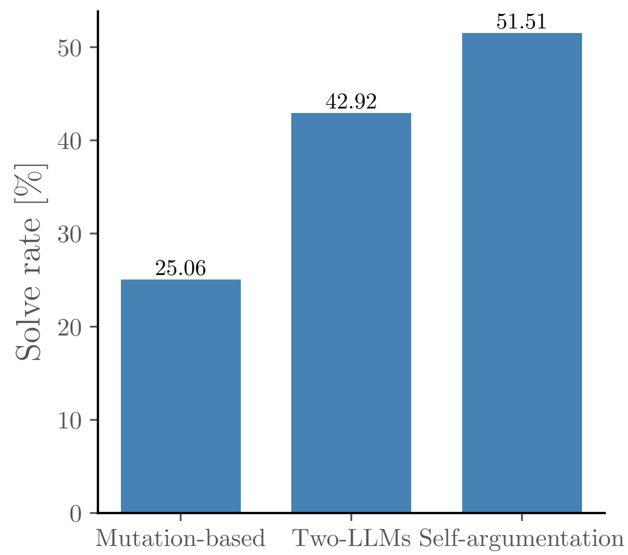

Figure 5. Solve rate for LLM-based methods. Mutation-based approach performs quite poorly. Self-argumentation prompting is the second best overall, with a 3% lower rate compared to CoT.

The worst-performing LLM-based method is mutation-based. Selecting from a pool of potential prompts seems not to bring the expected results. The potential problem is that these prompts are not generous enough, even though they are directly imposed from the Promptbreeder article. Even selecting only the best mutations (a way of overfitting) the highest score is 53% (with 15 occurrences) which is lower than CoT. The thinking styles performed even worse with an 18% solve rate.

The usage of LLM-created arguments seems to improve the performance. Using the same LLM for arguments and predicting proved to be a somewhat better option. One explanation could be, that if LLM has to create arguments, it already goes towards the right solution. On the other hand, using already provided arguments, gives the LLM some reasoning, without really coming up with that answer.

Open-ended QA evaluation

The results are obtained with the LLama2 model with 7 billion parameters, evaluated on the ProtoQA dataset.

Note that the evaluation of the creative tasks is somewhat subjective. As mentioned, for every approach, we evaluated the answer's correctness, creativity, and number of options it provided. Quality is represented by grades from 0 to 5, where 0 is graded when the answer is not applicable, and 5 represents the highest quality. For each method, we graded 15 examples and then took the mean grade of all answers as the final grade of the method.

Results are presented in Table 1.

Method	Quality [%]
Raw prompt	69.09
Zero-shot CoT	79.09
Chain-of-Thought	**80**
Plan-and-Solve	76.36
Self-argumentation	70
Two-LLMs	73.63
Mutation-based	29.09

Table 1. ProtoQA evaluation results. Results of different methods on open-ended questions.

We can notice that most methods perform well, with the Chain-of-Thought method being the best. The worst-performing method is mutation-based, which is relatively consistent with the results obtained on the CommonsenseQA dataset. Most methods provide similar answers and also fail on the same questions.

Additional experiments

After analyzing previously stated results, we wanted to test if the model's size affects the results for the self-argumentation prompting. We thought, that with more parameters, LLM is more likely to provide better argumentation.

The comparison of the solve rate between CoT and the self-argumentation method is presented in Figure 6. The results are obtained with the LLama2 model with 13 billion parameters, evaluated on the CommonsenseQA dataset.

As we can see, there are no significant differences between the two models. The solve rate for the self-argumentation method is slightly higher for the 13 billion parameters model, but this balances out with the CoT method, which performs somewhat better with the 7 billion parameters model. Taking into account the obtained results, and the complexity of the models, we may conclude that for this scenario, the more suitable model is the one with 7 billion parameters.

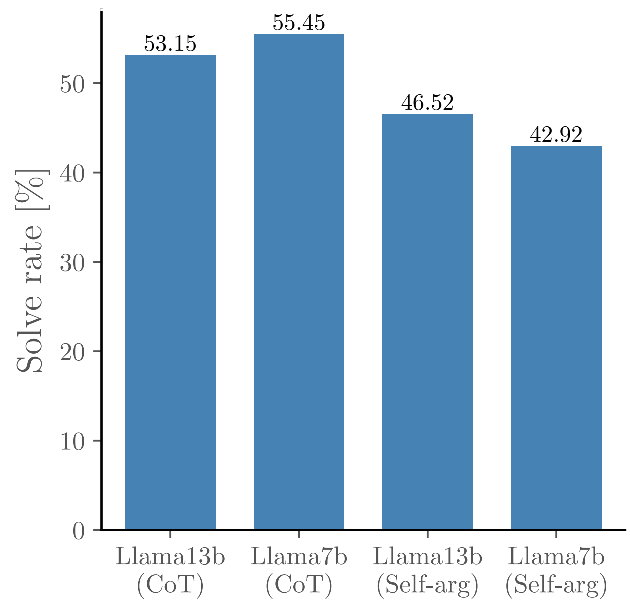

Figure 6. Comparison of CoT and self-argumentative methods between Llama7b and Llama13b.

Discussion

In this project, we proved that the way we prompt the LLM may have a large influence on its output. Providing any type of prompt seems to largely increase the quality of the model's answer, both for the multiple choice and creative tasks. It also seems that there are no significant differences between types of simple prompting, such as Zero-shot CoT and Plan and solve. Overall, CoT gave the best results in all evaluations.

Compared to simple methods, LLM-based methods are much more computationally complex. However, compared to CoT, they do not require additional user input.

We suppose that proposed LLM-based methods need more improvement to get the full potential out of them. Our aim in this work was to provide intuition for generating new prompting techniques and provide guidance for future works. In the work to follow, we would try LLM-based approaches with larger models, as we expect that this could enhance the performance. Also, we would evaluate prompting strategies on the more common dataset, as differences between CoT and others are relatively small.

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

[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] Protoqa: A question answering dataset for prototypical common-sense reasoning. GitHub, 2020.

[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.

1. Different prompt strategies

Table 2. LLM's answers with different prompt strategies. The question is: *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"*. The correct answer is (B).

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 in his vocation or profession. So, the answer is (A) vocation."

Chain-of-Thought: "The answer is (B) a 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."

Mutation-based: "Based on the scenario provided, the most likely outcome or goal that the person would hope for after getting hired is (B) a new job. The person is looking for a job in a different state, which suggests that they are hoping to start a new career or job in a different location. Therefore, option (B) is the most appropriate answer."
Mutation: "Help a system solve the multiple-choice questions by modifying/adding instructions based on the following rule: In one short sentence, what's a good prompt to get a language model to solve a problem like this?"

Two-LLMs: "Answer the following questions. You may use the arguments provided but only one answer is correct. Q: A revolving door is convenient for two-direction travel, but it also serves as a security measure at a what? Answer Choices: (A) bank (B) library (C) department store (D) mall (E) New York Arguments: Argument for Option (A): Bank - While it may seem counterintuitive that a revolving door would serve as a security measure at a bank, consider the logic behind it. Banks often have sensitive financial information and valuable assets that they need to protect from unauthorized access or theft. By using a revolving door, banks can ensure that only authorized personnel enter through the secure entrance while keeping potential thieves or intruders out. Additionally, a revolving door allows employees to easily exit the building without having to stop and open the main doors every time, which could be especially important during a robbery or other emergency. Argument for Option (B): Library - *(The rest of the arguments are omitted to preserve space)* Argument for Option (C): Department Store - *(omitted)* Argument for Option (D): Mall - *(omitted)* Argument for Option (E): New York City - *(omitted)*"

Self-argumentation: "Let's first consider arguments for and against each choice and then decide which one is the most convincing. After he got hired, he hoped for success in his: A) vocation - This answer choice is incorrect because the word v̈ocationr̈efers to a person's calling or profession, which is not related to the context of the passage. B) new job - This answer choice is incorrect because the passage does not mention anything about a new job. C) Michigan - This answer choice is incorrect because the passage does not mention anything about Michigan. D) working hard - This answer choice is incorrect because the passage does not mention anything about working hard. E) manual - This answer choice is incorrect because the passage does not mention anything about a manual. Based on the context of the passage, the most convincing answer choice is (B) job. The passage states that the man hoped for success in his job, which suggests that he had been hired for a position and was hoping to be successful in it."

---

2. Open-ended q&a examples

Table 3. Examples of questions and answers for open-ended questions.

Question 1: Name something people might turn their garage into. Answer:

- Home gym or workout space
- Home office or workspace
- Art studio or craft room
- Music room or recording studio
- Game room or entertainment area
- Storage unit or organization space
- Workshop or woodworking area
- Car repair or mechanicś shop
- Home brewery or winery
- Home theater or movie room

Question 2: Name something that keeps children safe.
Answer: Car seats!

Question 3: Name a vegetable that is about as big as your head.
Answer: A good candidate for a vegetable that is about as big as my head is a bell pepper. Bell peppers are typically around 2-3 inches in diameter, which is similar to the size of an adult human head.