Symbolic Math Reasoning with Language Models

Vedant Gaur
Aragon High School, Foster City, USA
vedantgaur101@gmail.com

Nikunj Saunshi
Princeton University, Princeton, USA
nsaunshi@cs.princeton.edu

2022 IEEE MIT Undergraduate Research Technology Conference (URTC) | 978-1-6654-7345-3/22/$31.00 ©2022 IEEE | DOI: 10.1109/URTC56832.2022.10002218

***Abstract*—The emergence of large language models (LLMs) such as OpenAI's GPT-3, Google's LaMDA, Meta's OPT [2, 3, 7, 10] etc. have revolutionized the field of natural language processing (NLP). These models with upwards of hundreds of billions of parameters are trained on large unlabeled text corpora and can subsequently solve downstream tasks with little to no labeled data. While these models are increasingly versatile in their abilities, e.g., solving math word problems, the larger question of their ability to reason remains. Using and modifying the SVAMP dataset, we find that GPT-3's davinci-002 model, in addition to having good performance on numerical math word problems, also performs well on the potentially harder symbolic version of the same problems. Furthermore, adopting a two-step approach (solve symbolically and then substitute numerical values) leads to better accuracy on the numerical test set in the zero-shot regime. Additionally, we find that the use of specific prompting techniques pushes the model, in many cases, to actively describe its thought process and aid in the final answer output when faced with a complex, multi-step problem, aligning with recent observations.**

Keywords—Natural Language Processing, Zero-shot, Large Language Models

I. INTRODUCTION

Autoregressive Language Models (LMs) aim to model the distribution of natural language by being trained on large text corpora. A popular idea to do so is through training an LM on the task of next word prediction, i.e. given a context (sentence or a partial sentence), an LM outputs a distribution over the set of words that could potentially follow that given context. This next word prediction ability of an LM can be leveraged to generate or sample longer sequences of text by sampling one word at a time. Large LMs trained with this simple strategy of next word prediction turn out to be very useful for a wide range of conventional Natural Language Processing (NLP) tasks, such as machine translation, question-answering, and sentiment analysis, taking us closer to the goal of constructed general-purpose learning agents.

Recently there has been an increased interest in exploring applications of LMs in more versatile scenarios. Rather surprisingly, LMs have been shown to have some proficiency in solving mathematical tasks, specifically math word problems [4, 6, 8]. Such tasks are non-trivial to solve, since they implicitly require understanding a text problem statement, parsing it appropriately, and then performing simple arithmetic calculations to arrive at a final answer; see Figure 1(a) for an example. Nevertheless, such a word problem can be solved by an LM, off-the-shelf, by simply asking it to *generate the answer* to a given problem. This surprising success of LMs has prompted further research, leading to the observation that the performance on such tasks can be significantly boosted in many cases through the use of appropriate *prompts* while querying the LM [4, 8]. This suggests that LMs possibly have the capability to do even better on mathematical tasks if used in the right way.

Inspired by the above observations, this work further explores the reasoning abilities of LMs, when used off-the-shelf. In particular, rather than evaluating LMs on numeric math word problems, we propose evaluating them on the potentially more challenging task of symbolic math word problems; Figure 1(c) provides an example of a symbolic word problem. Instead of dealing with numbers directly, the LM is asked to solve a math problem by returning a symbolic math expression that depends on the variables present in the problem. The numeric task can then be solved by asking the LM to substitute the variables with their appropriate numeric values from the original numerical problem. Using and evaluating an LM in this way confers multiple benefits:

- Provides a more "human-like" approach to solving the problem through an intermediate symbolic step
- Interpretability: Encourages the LM to output a more interpretable solution to the problem. This makes it easier to debug the model in case of mistakes. It also alleviates the fear that the LM solved the problem by "memorizing" rather than understanding math.
- Robust evaluation: Rather than evaluating the LM on all possible numbers, this provides a way to evaluate the LM on a more generic version of the problem that is independent of specific numeric values.

We find that the manual eval of symbolic accuracy (66%) of LMs is quite high, not much lower than the numeric accuracy (72%), despite potentially being a more challenging task. Secondly, we find that a two-step evaluation can lead to accuracy at the same level as solving the numeric task directly, with an added advantage of being able to debug the incorrect answers better.

The basis of this paper lies under the assumption that Large LMs act in one of two states: a compilation of their training data, enabled by their huge training set and number of examples; simply acting as a lookup table when presented with an unseen question. This means that the output of the model is heavily dependent on what it has seen in the training data, and how well the task aligns with prior examples the model has seen. On the other hand, the model may exhibit more "intelligent" behavior, i.e., exhibiting an ability to understand and reason. The problem with traditional approaches to this

Code available at https://github.com/vedantgaur/LM-Math-Paper-22

978-1-6654-7345-3/22/$31.00 ©2022 IEEE

problem is that by using numerical datasets, the assumption of the model simply pulling from the training data is overlooked.

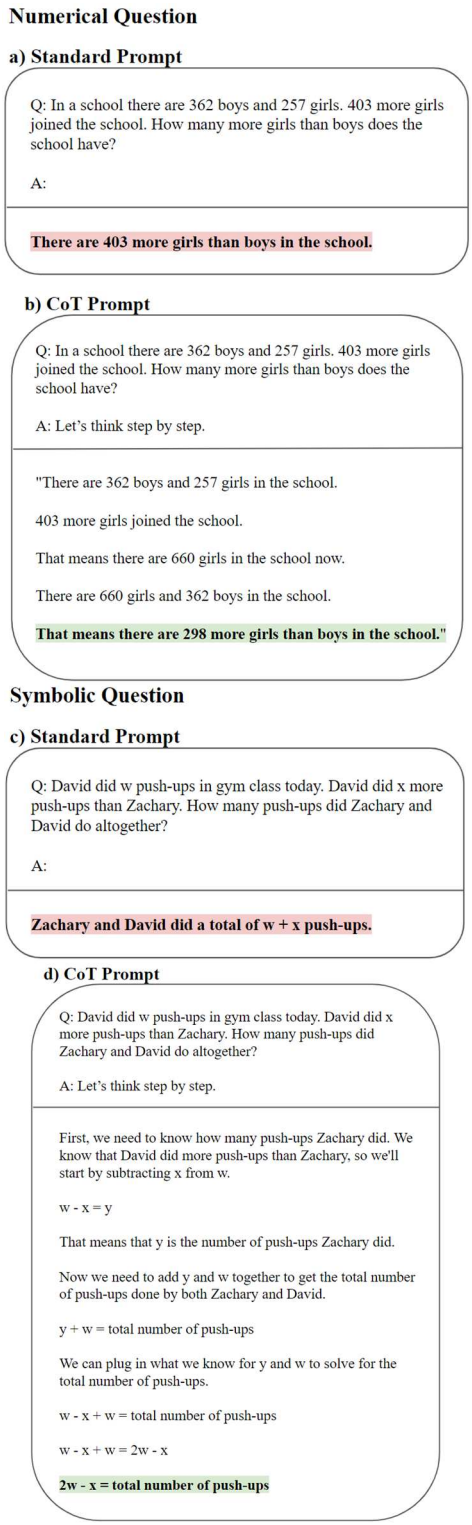

Fig. 1. Examples of GPT-3's davinci-002 being run on both numerical and symbolic prompts. The "Let's think step by step." prompt can yield much more detailed and insightful outputs from the model and in certain cases push the model towards the right answer.

II. BACKGROUND

A. LMs for downstream tasks

LMs have been used for traditional NLP tasks such as machine translation, sentiment analysis, and question answering. Recently, however, LMs have been able to solve math or logic problems to a reasonable extent. This "higher level" or capability of thought through generation speaks more to an LM's reasoning capabilities than conventional downstream tasks. While machine translation and question answering can be encoded into a model through a training corpus, mathematical thinking relies more heavily on having a broad understanding of patterns and formulas, and the awareness to know when to use what.

While LMs' ability to accurately solve a large variety of tasks is a weak measure of intelligence and understanding, it is unclear whether these models have the ability to contextualize information in an input and utilize such to answer the prompt. The hundreds of billions of parameters bolster the model's performance in natural language tasks but conversely create the black box paradigm that makes it difficult to uncover the reason behind its successes. These models can just as easily utilize the massive amounts of data that they are trained with to simply find patterns and essentially pull from training examples in order to solve a problem. This could yield a false sense of "intelligence" that results in outputs that are slightly off of the correct answer, or reasoning steps that have no relation to the problem being solved.

B. Prompting

Prompting is useful in eliciting useful and accurate information from an LM, by "priming" it with an appropriate piece of text that describes what kind of outputs one expects from the model. The wording of a prompt can modify the answer that the given LM provides. We used this capability of prompts to shift LM output towards higher accuracy on math word problems and to gain a better understanding of the thought process of the model. CoT prompting has been a step toward optimizing arithmetic/logic outputs from LMs by acting as an intermediary step before the model's response (Ling et al., 2017; Amini et al., 2019; Chen et al., 2021; Cobbe et al., 2021). While proven effective at eliciting more accurate and logical responses regarding math and reason word problems, it still severely lacks in capabilities exhibited by even elementary school children.

C. Evaluation

LMs can be evaluated either in the zero-shot or one-shot prompting. Zero-shot prompting provides the model with simply a problem and the beginning of an answer like "Let's think step by step". The LM is then expected to generate an answer. Such examples can be found in Fig. 1. One-shot prompting is similar, although it provides the model with one problem and a solution to it first, and then another problem with the same format as the zero-shot prompt. One-shot prompting is similar to the idea of *few-shot in-context learning* where a model is given a series of examples and is then told to solve the problem or predict the next word. *Zero-shot learning* is the fundamental opposite, where the prompt contains a situation

that the model has never seen before. While the one-shot prompt only includes one example, it is given a problem from the same dataset in order to expose the model to methods of solving. Every one-shot run was done using the same exact prompt or *context*, chosen because the solution to the problem had multiple steps and operations.

Each evaluation method contains prompts that (1) simply provide the prompt, (2) employ CoT reasoning by adding "Let's think step by step", and (3) prompt the model to generate intermediary variables to solve the problem (the intermediate variable prompt uses the string: "Let's think step by step. Introduce intermediate variables and solve the task symbolically.").

D. Symbolic math problems

This paper is more concerned with the ability of vanilla, i.e., non-finetuned, LMs to correctly solve math word problems in a symbolic form, where numbers are replaced by variables. Prior work has taken the finetuned approach with high accuracy [5]. Minerva is able to correctly solve much higher-level math problems than simply elementary word problems, though not completely symbolic. The symbolic prompts that we used can be seen in Fig. 1 (c) and (d) and in Fig. 2. While having a specialized LM may result in high accuracy at the specified task, benchmarks on traditional LMs can be more insightful as to whether these models can solve generic, unseen problems.

III. Data Manipulation and Uses

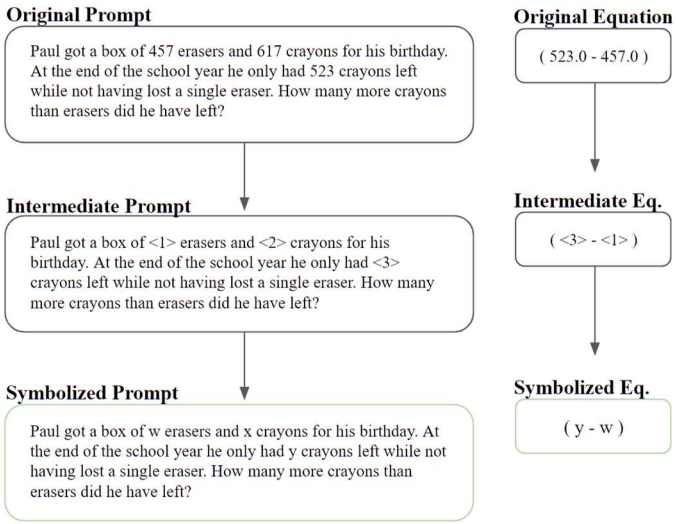

Fig. 2. The process of symbolizing a prompt as well as an equation. Each number is replaced by an intermediary variable which can then be modified again to reach the final symbolized prompt.

A. Symbolizing Dataset

To test LMs on symbolic tasks, it is necessary to have a generic dataset that could be easily modified to test the extent to which LMs can perform arithmetic. By introducing a symbolic prompting system, it is much more clear whether or not a model is able to answer a problem, and in conjunction mirrors the contextual understanding and prompt-based steps exhibited by humans. Moreover, it is much easier to generalize and modify an input to test various numerical and symbolic values. We describe the process to convert a numeric problem to a symbolic one below; see Figure 2 for a visualization.

Each number in both the question and provided answer and equation is extracted via a Python script and saved into a new column in the dataset. The numbers are replaced with specific tags <1>, <2>, …, <n>, which can then be replaced with any series of variables in the testing phase.

For our testing, we chose to use {w, x, y, z} as variables. The problems only ever have 4 numbers at most meaning that any combination of the variables can be used. In our testing, the model could distinguish between each of the variables and output a sensible expression or equation. Further testing can explore which variables, if any, tend to yield better context extraction from the model.

B. Evaluation Method

This paper primarily focuses on evaluating GPT-3 on symbolic tasks, though we have provided metrics for runs on numerical questions as a means to compare results. Both symbolic and numerical accuracies were calculated manually due to the high variance in model outputs. Answer extraction helped isolate the final output in most cases but also included unnecessary information occasionally. This meant that automatic extraction would not have been reliable in the current state and could either miss the actual output of the model or be confused by other values present.

Thus, all evaluation was done manually by referencing the correct symbolic expression and numerical answer, and in the cases of CoT outputs, to peer into the reasoning to get a further idea of the model's performance.

A third evaluation method, *two-step prompting,* was also introduced as an applicable method of symbolic prompt q&a. By converting the prompt to a symbolic one, running the model on the symbolized prompt, and then plugging back the original variables, *two-step prompting* more closely mimics techniques such as substitution and allows for the full utility of symbolic prompt accuracy.

As it is infeasible to do manual accuracy evaluation on all 1000 prompts present in the SVAMP dataset, we chose 50 random prompts (to reduce experimental uncertainty) and ran the model on these tasks. Further prompt finetuning for answer extraction can be pursued in order to successfully implement automated evaluation.

IV. Experimental Results

We performed a variety of tests to determine whether a specific method of prompting would yield higher accuracy on the SVAMP dataset. Surprisingly we did see that symbolic accuracy was able to meet that of previous papers on numerical datasets. Moreover, in cases like one-shot, the model performed with much greater accuracy on symbolic at 80%; see table 2. The higher numbers for numeric problems compared to prior work is likely because of a manual evaluation instead of automated evaluation that could lead to false negatives.

These results conflict with the expectation of LMs having lookup table-like behavior, as they seem equally adept, if not better when faced with an ambiguous prompt. Because sets of symbolic word problems do not exist extensively online—rather datasets tend to focus on numerical tasks—the section of the training corpus that is symbolic prompts is likely minimal.

Of each of the sections, we report in tables I and II the number of problems that were seen as ambiguous, generally right in reasoning but wrong in its final output, or vice-versa (without steps, with steps, intermediate variables):

TABLE I. ZERO-SHOT RESULTS

Zero-shot Symbolic	*s - without steps*	*s - with steps*	*s - intermediate*
Accuracy	48%	66%	56%
Ambiguous considered correct/incorrect	2% / 0%	2% / 0%	2% / 2%

Zero-shot Numeric	*n - without steps*	*n - with steps*	*n - intermediate*
Accuracy	70%	72%	54%
Ambiguous considered correct/incorrect	0% / 4%	0% / 0%	4% / 2%

Two-step	*n - without steps*	*n - with steps*	*n - intermediate*
Accuracy	54%	72%	58%
Ambiguous considered correct/incorrect	2% / 2%	2% / 0%	0 / 4%

a. Zero-shot results with normal, CoT, and intermediate variable prompting. The bottom row represents the number of model outputs that are considered either close to true, or at fault in a manner other than an outright incorrect answer. The first percentage represents the number (percentage of total) of these outputs that we consider to be correct, and the second, the ones we consider to be incorrect. The titles with "n" denote numeric evaluations, and the ones with "s" denote symbolic evaluations

TABLE II. ONE-SHOT RESULTS

One-shot Symbolic	*s - without steps*	*s - with steps*	*s - intermediate*
Accuracy	66%	62%	80%
Ambiguous considered correct/incorrect	2% / 0%	12% / 0%	8% / 0%

One-shot Numeric	*n - without steps*	*n - with steps*	*n - intermediate*
Accuracy	66%	60%	50%
Ambiguous considered correct	2% / 2%	0 / 2%	0 / 2%

b. Zero-shot results with normal, CoT, and intermediate variable prompting. The bottom row represents the number of model outputs that are considered either close to true, or at fault in a manner other than an outright incorrect answer. The first percentage represents the number (percentage of total) of these outputs that we consider to be correct, and the second, the ones we consider to be incorrect.

V. ANALYSIS

Largely, it is seen that symbolic prompts are solvable by LMs in specific scenarios. While zero-shot CoT does comparatively worse than its numeric counterpart, it is still able to yield similar benchmarks to previous CoT results on numeric datasets. Moreover, it is seen that CoT prompting is helpful in two-step evaluation, yielding much higher results than previous numeric CoT papers, if we give the model a benefit of doubt when it comes to ambiguous answers.

Zero-shot prompting: This seems to be the worst performing section on symbolic prompts, with high CoT accuracy, but little else. Surprisingly, numerical prompts performed extremely well and were fairly consistent between the prompts with and without CoT.

One-shot Prompting: One-shot results were much more insightful than the other sections, as the model performed consistently better on symbolic tasks than numerical ones. Most surprising is the accuracy of the model when prompted to define intermediate variables. The model did much better on this subsection than on any other. A possible explanation for this behavior and the fairly mediocre results for intermediate variable prompts throughout, is a grasp on how exactly to define these intermediary steps. In scenarios like zero-shot and two-step prompting, the model may lack an understanding in establishing intermediary steps. Through in-context learning, however, it seems like having an example input-output sequence proves to be extremely beneficial to the model.

Two-step Extraction: Given high two-step accuracy with CoT prompting, which exceeds previous benchmarks, we have provided a new, zero-shot method to compute numerical math word problems. With the large increase in accuracy from the prompt finetuning for symbolic tasks, two-step is quite versatile in providing a generic, accurate solution, that can be utilized for any array of values inputted.

Two-step prompting closely mirrors the method of *inductive bias* in which the model finds a general solution and then looks at the specifics of the input. This method of solution may be seen as more "human-like", mirroring the patterns humans pick up on and then employ to solve unseen problems.

A. *Different prompt*

The intermediate variables prompt prompted the model to introduce intermediate expressions and variables, to provide for a more rigorous solution to problems. This prompt seemed to largely confuse the model and wouldn't be much too useful in situations with minimal context. However, as proven by symbolic one-shot results in that column, the method of prompting can also yield high accuracy with little finetuning.

B. *Analysis of mistakes*

There can certainly exist some errors in accuracy calculations and determining the efficacy of the model in each situation. Since all benchmarks were evaluated manually, although unbiased evaluation was attempted, small biases could slightly alter accuracy values. Having an automated evaluation process could solve this, but again would be difficult to implement due to the nature of the model outputs.

Regarding model performance, we did see some errors in the model's reasoning and outputs: when faced with a question that is a little ambiguous or one that provides some ambiguity, the model seems to get confused and tends to infer, seemingly through context it has accumulated via its training corpus (*q#645,* or *index 35* in one-shot symbolic results).

The mathematical operation that the model uses is correct a lot of the time, but the answer isn't (i.e., if the operation is subtraction [let's say - *y*], the model is able to pick this up, but

tends to have the values wrong; $w + y - y$ instead of $w - y \Rightarrow$ final answer becomes w instead of $w-y$).

Q#355, or *index 33* in the SVAMP dataset has either an incorrect solution or incorrect question context (it should either say 80 instead of 8 or have the answer be w/x instead of $x*w$).

While each index was random, the script ended up producing the same prompt index twice. This means that there are two runs that have the exact same input.

The model seems to be able to grasp theory but sometimes forgets to perform and/or list down the operations reliably (it forgets the multiplication sign in two-step *index 3*, and thus gets the wrong answer).

There seems to be a *very* consistent multiplication error across most of the runs. In the instance of the question where the model does so, the numbers are quite large. The model sets up everything right but ultimately fails the operation. This is similar to conclusions drawn about large numbers in the Minerva paper.

VI. Conclusions and Future Work

One-shot reasoning proved to be the most consistently-well performing section out of the three, yielding 80% accuracy in the case of symbolic, intermediate variables prompts. This is to be expected given past papers on in-context learning. Overall, it seems like LMs can also solve symbolic versions of math problems reasonably well. However, there is still a lot of scope for improvement.

These experiments can be broadened to encompass (1) other LMs and (2) other datasets. Moreover, if answer extraction proves successful, more accurate benchmarks can be extracted. To further the discussion of context, testing what the cut-off for accuracy improvement is depending on the number of problems given in context could provide insightful results.

References

[1] Betz, Gregor, et al. "Critical Thinking for Language Models." ArXiv.org, 17 Dec. 2020, https://arxiv.org/abs/2009.07185.

[2] Brown, Tom B., et al. "Language Models Are Few-Shot Learners." ArXiv.org, 22 July 2020, https://arxiv.org/abs/2005.14165.

[3] Devlin, Jacob, et al. "Bert: Pre-Training of Deep Bidirectional Transformers for Language Understanding." ArXiv.org, 24 May 2019, https://arxiv.org/abs/1810.04805.

[4] Kojima, Takeshi, et al. "Large Language Models Are Zero-Shot Reasoners." ArXiv.org, 9 June 2022, https://arxiv.org/abs/2205.11916.

[5] Lewkowycz, Aitor, et al. "Solving Quantitative Reasoning Problems with Language Models." ArXiv.org, 1 July 2022, https://arxiv.org/abs/2206.14858.

[6] Patel, Arkil, et al. "Are NLP Models Really Able to Solve Simple Math Word Problems?" ArXiv.org, 15 Apr. 2021, https://arxiv.org/abs/2103.07191.

[7] Thoppilan, Romal, et al. "LAMDA: Language Models for Dialog Applications." ArXiv.org, 10 Feb. 2022, https://arxiv.org/abs/2201.08239.

[8] Wei, Jason, et al. "Chain of Thought Prompting Elicits Reasoning in Large Language Models." ArXiv.org, 13 June 2022, https://arxiv.org/abs/2201.11903.

[9] Yasunaga, Michihiro, et al. "Qa-GNN: Reasoning with Language Models and Knowledge Graphs for Question Answering." ArXiv.org, 5 Nov. 2021, https://arxiv.org/abs/2104.06378.

[10] Zhang, Susan, et al. "OPT: Open Pre-Trained Transformer Language Models." ArXiv.org, 21 June 2022, https://arxiv.org/abs/2205.01068.

Appendix

Why did we use SVAMP specifically?

The SVAMP dataset was primarily chosen because other datasets contain somewhat more complex problems that in testing yielded relatively low accuracy.

SVAMP had a few further advantages that were quite useful for our specific use case. The annotation file provided by the authors with the data had expressions that were solutions to each of the problems on top of the provided numerical answers. This allows for easy conversion and evaluation of symbolic outputs.

(Sample row section from SVAMP dataset):

Body	Question	Equation	Answer
Each pack of dvds costs 76 dollars. If there is a discount of 25 dollars on each pack	How much do you have to pay to buy each pack?	(76.0 - 25.0)	51

In preliminary testing, we find the complexity of other datasets to be true. Given the following question and model output from the AQuA-RAT dataset:

Two friends plan to walk along a 43-km trail, starting at opposite ends of the trail at the same time. If Friend P's rate is 15% faster than Friend Q's, how many kilometers will Friend P have walked when they pass each other?

We can set up a proportion to solve this problem. Let's call the distance that Friend P has walked x, and the distance that Friend Q has walked y. Then,

15/100 * x = y

We can solve for y:

y = 15/100 * x

Now, we know that the total distance of the trail is 43 km. This means that:

x + y = 43

We can substitute y in for 15/100 * x and solve for x:

x + 15/100 * x = 43

x + 15/100x = 43

x(1 + 15/100) = 43

x = 43 / (1 + 15/100)

x = approximately 41.86 km

This means that when Friend P has walked 41.86 km, they will pass Friend Q.

We see a few key errors:

1. The model incorrectly represents the relationship between Friend P and Q, having it so that Friend's P rate of travel is 15% of Friend Q's rather than 115%, or 15% more.

2. Instead of modeling the relationship between Friend P and Q's rate of travel, it erroneously has the values as total distance.