MathToT: Solving Math Problems using Tree of thought

Shrey Pandit
The University of Texas at Austin
shreypandit@utexas.edu

Abstract

Recent progress in the development of Large language models (LLMs) have shown their extensive capabilities in diverse tasks like classification, image generation and factual question answering tasks, etc. But these LLMs keep struggling with math based tasks like simple 3x3 or 4x4 multiplications. To alleviate this issue methods like scratchpad and chain of thought have shown significant performance boost. I explore the effect of methods like Tree of Thought which works step wise, and can help LLMs solve math problems more efficiently. I finally also explore the possible directions that can be dynamically be taken at every level of thought if the model is not confident about the generation. Finally the work points out the issues that exist with the current implementation from the original paper, and the possible solutions that could help improve the ToT based model performance.

1 Introduction

Problems in Mathematics can be solved in multistep way with multiple different methods to solve the same problem. This multistep solving methods was the motivation behind exploring the solving capabilities of the Large language models. Recent works have pointed out how these LLMs like GPT3.5 struggle with Math based problems. Tree of thought paper (ToT) [7] proposes a method which shows immense performance gain on tasks like crossword puzzles and creative writing. These datasets were synthetically generated by the authors to explore the benefits of ToT in tasks that require step wise thinking and decision planning. I plan to explore the performance of ToT on Math datasets like the GSM8K dataset [1] and to check for various metrics during the intermediate step wise generation. Chain of thought [6] shows the performance gain in the math datasets, this shows

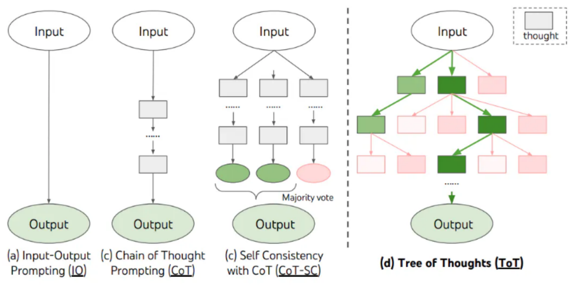

Input
Output
(a) Input-Output Prompting (IO)
(c) Chain of Thought Prompting (CoT)
Majority vote
(c) Self Consistency with CoT (CoT-SC)
thought
(d) Tree of Thoughts (ToT)

Figure 1: Figure showing how tree of thought compares to other modes of prompting

that such methods help the model solve math tasks. I hypothesise that a more extensive method like Tree of thought that bring out more explicit steps while solving a problem would reduce the mistakes a model make. Having the tree structure of exploration helps us utilize the other features of a tree model like Breadth first search, Depth first search and methods like backtracking in cases of model is not confident in generations, these methods give ToT an edge over the usual CoT methods. I summarize the contribution of this work mainly as -

- We utilize the tree of thought method on the math based dataset to evaluate its performance.
- As math problems might be partially or totally incorrect while solving I propose a new metric of evaluation to check if a solved question which is classified incorrect, up to what extend is it correct.
- Finally I discuss the possible steps that can be taken at every layer of the ToT model to make the decisions of solving more accurate and efficient.

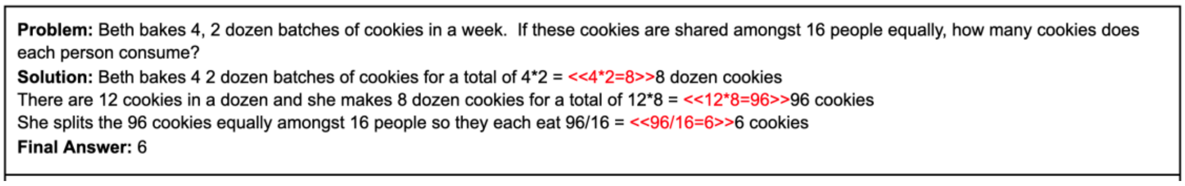

Problem: Beth bakes 4, 2 dozen batches of cookies in a week. If these cookies are shared amongst 16 people equally, how many cookies does each person consume?
Solution: Beth bakes 4 2 dozen batches of cookies for a total of 4*2 = <<4*2=8>>8 dozen cookies
There are 12 cookies in a dozen and she makes 8 dozen cookies for a total of 12*8 = <<12*8=96>>96 cookies
She splits the 96 cookies equally amongst 16 people so they each eat 96/16 = <<96/16=6>>6 cookies
Final Answer: 6

Figure 2: Example of a question from the GSM8K dataset

2 Related Work

Tree of thought prompting - Tree of thought (ToT) is an extension of the popular method Chain of thought prompting [6], which argues that Large Language Models (LLMs) are constrained by token-level, left-to-right decision-making. The paper emphasizes that initial choices significantly impact subsequent steps and proposes a tree-like thought generation structure. This structure involves self-evaluating choices through node voting to select the best node for exploration. ToT works by generating multiple different thoughts at every layer, and choosing for the best thought based on ranking. It supports features that a tree based search method encorporates like depth first search, A^* Search, Breadth first search, Monte-carlo tree search etc. using which various different thoughts could be explored efficiently.

Recent work by [4] show the possible improvement in the math solving capabilities of LLMs by proposing a similar method of clustering, however they evaluate on LLMs like PALM and GPT4 with chain of thought prompting, and leave out methods like Tree of thought prompting. The work shows significant performance gain by utilizing such clustering and self consistency methods further approving the need for above experiments.

3 Dataset and metric

I plan to use the GSM8K dataset which has around 8k math problems that might require various steps, variables to solve the problem and might have many different methods to solve the same problem. The dataset provides with a question and an answer as shown in figure 2 .

I propose two metrics in the analysis, first one being accuracy, which is the percentage of correct final responses when evaluated on the entire test dataset. This would give the broad overview of how the model is performing using the ToT method on the dataset and can be used to compare the various different LLMs.

Second Metric that I propose is a novel method of evaluating step wise correctness of the math problem, for this I use the method that involved number caching. This is might be categorised as a crude but efficient method of checking if the intermediate steps involve some wrong steps. Here I cache all the numbers that exist in solving of the math problem, from the question to intermediate steps. I use the numbers that exists in the gold truth response, like in the example above, steps involved in chain of thought prompting method (I prompt the model 2-3 times to generate multiple different sequences) and cache only those values when the CoT gives out a correct response. For example in the above answer given the number cache would involve numbers like [2,48,24,72] from the ground truth and other numbers from the CoT prompting methods. I take a union of all these caches for a given question and consider it as the gold truth cache.

For every intermediate step I check if all the number in the generated prompt is in the gold truth cache, if yes then I consider the step to be correct else I mark it as a wrong step. It is important to note that this is not the best method to know the intermediate steps, but seemed as a good basic method to evaluate the step wise performance. I take union of multiple caches to reduce the False negatives but there is still chances of existence of False negatives. I show the values of step wise performance of solutions that resulted in a correct overall outcome and a wrong overall outcome in fig 1 and 2. Also, this method is an analysis metric and needs to have the access of ground truth label, so cannot be used during training and can only be used to gain insights from the model.

4 Methodology

I plan to evaluate LLM like GPT3.5-turbo on one shot setting, and plan to extend this on other LLMs in future. Currently I have created a pipeline to

Method	Direct Prompting	Chain of thought	Complex Chain of thought	Tree of thought
GPT 3.5 turbo	15.9%	56.1%	67%	46.5%

Table 1: Table comparing the results of ToT with other prompting methods

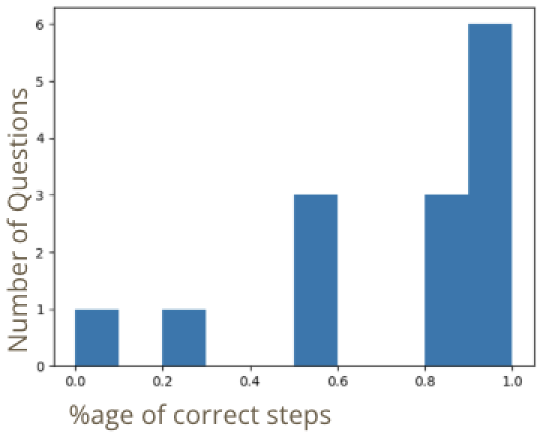


Figure 3: The step wise accuracy when the final answer is correct. X axis denotes the step wise accuracy and Y axis denotes the number of samples

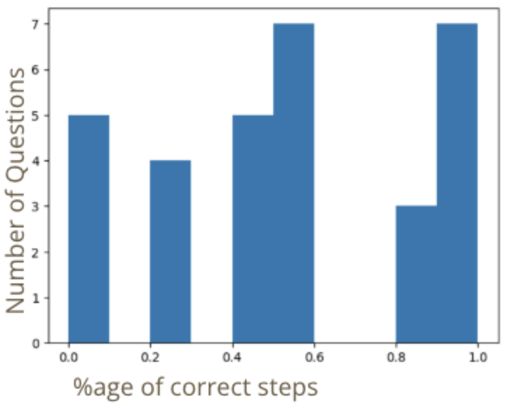


Figure 4: The step wise accuracy when the final answer is wrong. X axis denotes the step wise accuracy and Y axis denotes the number of samples

evaluate the Tree of thought prompting method on GSM8K dataset, this was not straight forward, as the current implementation had an implementation where every step was independant of the other steps. This was because the authors worked on datasets like Cross word puzzles and Gameof24 where the current state is the only necessary information needed. To solve a math problem, we can have multiple different methods but each independant step needs to have the knowledge of all previous states which will be used to determine the future steps take, to solve this issue I included a memory array in the ToT implementation that takes into account all the steps taken so far, the values of which are needed in current and future calculation.

Then the exploration of the tree based search is using greedy search as that seemed to be straight forward. We store the scores of all the ranking that are used in methods like backtracking.

At every step (new layer of the tree) we calculate the scores, and stop the exploration of the tree in case the scores are below a certain threshold and backtrack along the tree, to explore the next best combination. I also tried a method where I prompted multiple GPT models with one output each (instead of multiple output by the same GPT) which gave nearly identical performance.

In future, I plan to calculate the entropy of the model generation [3] at every layer and pass this as a metric to the ranking model, along with other features to help it decide if the exploration should be stopped and also for which option the model is confident. Finally we explore till the model has reached the answer to the question.

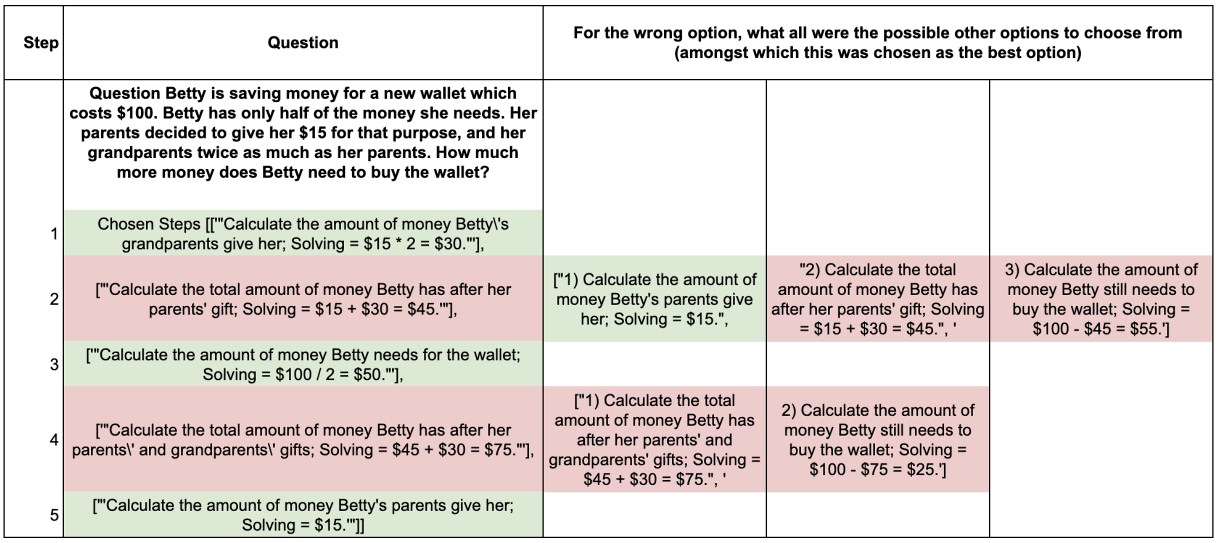
5 Results

The authors that proposed ToT evaluate the method on synthetic dataset specifically created to show the performance gain of this step wise method like ToT on tasks like Gameof24, and Crossword puzzles, however they dont evaluate on any math or real world datasets.

5.1 Baseline models

Currently there exists no work that evaluates Tree of thought prompting on GSM8K method, however there are works that evaluate the GSM8K method on Chain of thought prompting and direct prompting. We consider direct prompting model to be the primitive method to compare, which ToT outperforms by a big margin.

For more complex methods like Chain of thought and Complex chain of thought [2] we compare them with the ToT method. If we consider ToT to be a more complex varient of CoT then is expected to outperform it in the worst case, which is not the case as shown in table 1. This points out to some issue in the Tree of thought method, next in section



Step	Question	For the wrong option, what all were the possible other options to choose from (amongst which this was chosen as the best option)		
	Question Betty is saving money for a new wallet which costs $100. Betty has only half of the money she needs. Her parents decided to give her $15 for that purpose, and her grandparents twice as much as her parents. How much more money does Betty need to buy the wallet?			
1	Chosen Steps [['''Calculate the amount of money Betty\'s grandparents give her; Solving = $15 * 2 = $30.'''],			
2	['''Calculate the total amount of money Betty has after her parents' gift; Solving = $15 + $30 = $45.'''],	["1) Calculate the amount of money Betty's parents give her; Solving = $15.",	"2) Calculate the total amount of money Betty has after her parents' gift; Solving = $15 + $30 = $45.", '	3) Calculate the amount of money Betty still needs to buy the wallet; Solving = $100 - $45 = $55.']
3	['''Calculate the amount of money Betty needs for the wallet; Solving = $100 / 2 = $50.'''],			
4	['''Calculate the total amount of money Betty has after her parents\' and grandparents\' gifts; Solving = $45 + $30 = $75.'''],	["1) Calculate the total amount of money Betty has after her parents' and grandparents' gifts; Solving = $45 + $30 = $75.", '	2) Calculate the amount of money Betty still needs to buy the wallet; Solving = $100 - $75 = $25.']	
5	['''Calculate the amount of money Betty's parents give her; Solving = $15.''']]			

Figure 5: Figure showing the poor performance of ranking model which chooses a wrong option even though a correct option was in the pool of options.

7 we analyse this issue and point out to specific areas which cause this performance to drop.

6 Step wise accuracy metric

We calculate the step wise accuracy of the outputs given by the model as described in section 3. The results are reported in the figure 3 and figure 4. The figure 3 shows the step wise performance of the solution when the overall result of the question was correct. It is clear that the model hardly makes any step wise mistakes when the final answer is correct.

Interestingly we can observe that in figure 4 the existence of correct steps in an incorrect solution, along with totally wrong steps which is expected. When glanced through manually we see that the model correctly computes the steps but the overall planning of the model is weak, which is the model could not reach the final solution correctly, and simply couldn't solve the problem due to lack of complete solution. This points out to the face that these models need to have a good planing mechanism where they could take steps in a organised manner which is not a strength of GPT 3.5.

7 Potential reason for failure of ToT

We see that the ToT method does not outperform the CoT method which is a less complex variant. This points out to the ranking process of the method. The current ranking methodology used by the original paper prompts an LLM for choosing between the steps, which does not seem an ideal option to do for a math problem. This can be verified from the figure 5 where we can see that the ranking model has chosen options incorrectly despite a correct option existing in the pool of options. This incorrect choice then propagates to the other decisions and further steps taken by the model.

A solution to this issue would be to make the ranking scheme defined properly instead of just prompting LLM for it. For this I propose to introduce a rubric to the ranking scheme which would rank the pool of options on the basis of few parameters like the novelty of a step (it should not be a step that has been taken before), correctness of a step (if there is a calculation mistake it should not choose the step), and the effectiveness of choosing that step (the step should help the model reach the final answer). I also plan to introduce an entropy score which would take into account the confidence of the model while generating the output which can be used while score the option, a lower score could be offered to an option in which the model was not confident while generating it (more about this in the future work).

8 Conclusion

We try to explore the popular method of tree of thought on the math dataset, GSM8K. Several adaptations were necessary to make the ToT method compatible to solve a math problem like having a memory buffer and steps to make sure the model does not solve the entire math problem in one step. We also propose a metric to analyse the step wise accuracy of the problems and find that the model lacks the planning skills because of which it fails to correctly answer even after partially solving the question correctly. We then observe the outputs of the model and find out that the ranker used in ToT needs further modification like a rubric to correctly choose options, and that simply prompting an LLM to rank the options would not suffice in a math solving task. Finally we propose methods like entropy calculation to further aid the ranker which would help in the improvement of ToT model performance.

9 Future Work

I plan to correct the ranking methodology currently proposed by the paper, as simply prompting an LLM without any guidance is not the correct way rank a math solution. I plan to provide a rubric for evaluation and choosing of the option, which I suppose can give significant boost to the performance of the model.

Next, I plan to incorporate the self-consistency method where I will prompt various different instances of LLMs to have a pool of possible next steps instead of just one and then choosing from one of the options.

Lastly, on the basis of work done by [5] I plan to cluster the solutions and then choose an option from the clusters. This will help measure the entropy of the model while generating the output as shown by [3] and we could analyse any possible correlation of entropy with correct step generation which we can utilize to in the selection of step process.

References

[1] K. Cobbe, V. Kosaraju, M. Bavarian, M. Chen, H. Jun, L. Kaiser, M. Plappert, J. Tworek, J. Hilton, R. Nakano, C. Hesse, and J. Schulman. Training verifiers to solve math word problems, 2021.

[2] Y. Fu, H. Peng, A. Sabharwal, P. Clark, and T. Khot. Complexity-based prompting for multi-step reasoning. In *The Eleventh International Conference on Learning Representations*, 2023.

[3] L. Kuhn, Y. Gal, and S. Farquhar. Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation, 2023.

[4] Y. Liu, A. Singh, C. D. Freeman, J. D. Co-Reyes, and P. J. Liu. Improving large language model fine-tuning for solving math problems, 2023.

[5] A. Opedal, N. Stoehr, A. Saparov, and M. Sachan. World models for math story problems, 2023.

[6] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. Chi, Q. Le, and D. Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2023.

[7] S. Yao, D. Yu, J. Zhao, I. Shafran, T. L. Griffiths, Y. Cao, and K. Narasimhan. Tree of thoughts: Deliberate problem solving with large language models, 2023.