ECE 696B: Spring 2025
Trustworthy Machine Learning

Tree of Thoughts: Deliberate Problem Solving with Large Language Models

Presented by: Zhenyu Jin

THE UNIVERSITY OF ARIZONA

Questions Raised

The inference process of LLMs is still at the token level, proceeding from left to right.

Previously, Chain-of-Thought (CoT) has allowed LLMs to reason through problems step by step. However, CoT has an obvious drawback: it neither expands on intermediate steps nor incorporates foresight or backtracking.

Inspiration

Humans have two modes of decision-making: one that is fast, automatic, and unconscious, and another that is slow, deliberate, and conscious. The former is known as System 1, while the latter is referred to as System 2.

Which one does simple token-level associative decisions resemble?

Background

1. **Input-Output (IO) Prompting**

The most commonly used method—input a question and get an answer.

2. **Chain-of-Thought (CoT)**

When the relationship between the input question and the output answer is not immediately obvious or intuitive (such as in mathematical derivations), CoT enables the model to generate intermediate reasoning steps.

Under CoT, the transition from each step to the next is clear and logical, linking an otherwise disconnected input and output through a structured reasoning process.

3. **Self-Consistency with CoT (CoT-SC)**

This method involves running CoT multiple times and then making a final decision by voting on the generated outputs. CoT-SC improves the performance of CoT because LLMs can prove the same theoretical question in different ways. Additionally, by sufficiently expanding the reasoning process, the final conclusion becomes more reliable.

However, each reasoning chain in CoT-SC does not involve "local expansions" (branching into different paths). Furthermore, the voting mechanism is only effective in cases where the output space is limited (e.g., multiple-choice questions).

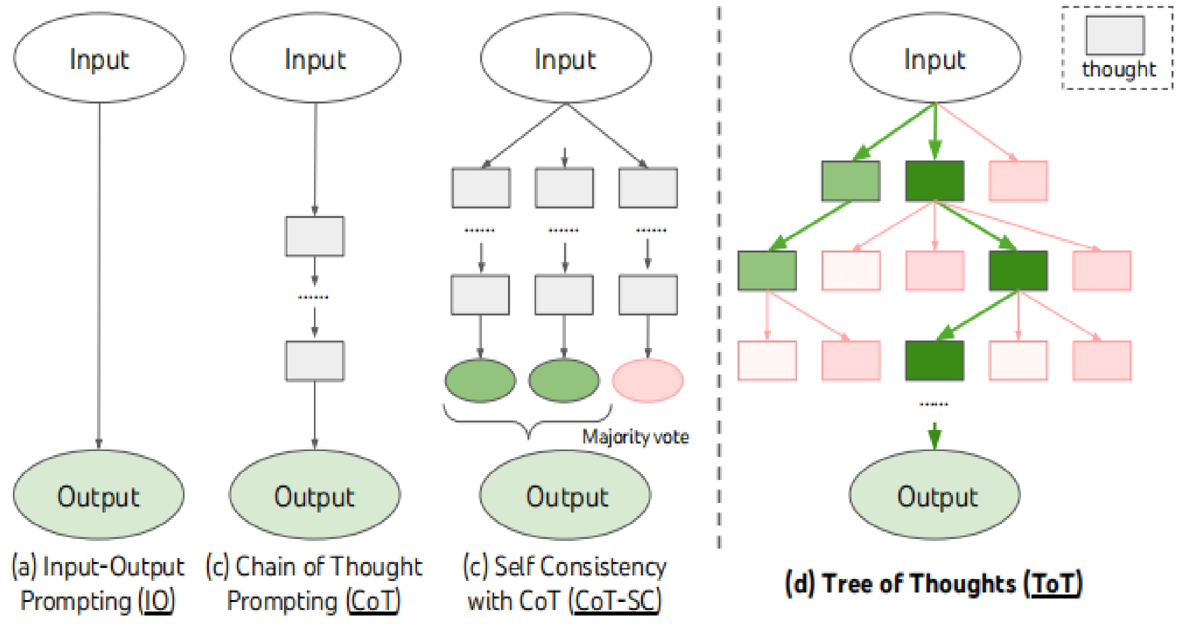
Theory Generated

This paper proposes extending the reasoning capabilities of LMs into a tree structure, referred to as Tree-of-Thoughts (ToT), where each "thought" corresponds to a node in the tree.

This paper introduces a self-evaluation method for LMs, which serves as a heuristic to guide ToT's search process.

Compared to the previously mentioned methods, ToT can expand into different sequential thought branches, forming the tree's structure.

Graphically Discription



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

Figure 1: Schematic illustrating various approaches to problem solving with LLMs. Each rectangle box represents a *thought*, which is a coherent language sequence that serves as an intermediate step toward problem solving. See concrete examples of how thoughts are generated, evaluated, and searched in Figures 2,4,6.

ToT abstracts the problem as a search task on a tree structure, and such a search task consists of four components: thought decomposition, thought generator, state evaluator and search algorithm

1. Thought Decompostion

Different problems require different decomposition methods. In the three experiments presented in this paper, a thought can be a group of words (Crossword game), an equation (24-point game), or a writing plan (creative writing).

When decomposing thoughts, the following principles should be followed:

The thought should be short enough for LMs to generate promising and distinct ideas.

The thought should also be long enough for LMs to evaluate whether it is promising.

THE UNIVERSITY OF ARIZONA

2. Thought Generator

The formula for the thought generator is:

$$G(p_\theta, s, k)$$

where p_θ represents a parameterized LLM, s is the information available when generating the next thought z_{i+1}, and k is the number of generated thoughts. There are two strategies for generating thoughts:

Generating k numbers of independent distribution of thoughts from a CoT prompt (same goal but $z^{(j)} \sim p_\theta^{CoT}(z_{i+1}|s) = p_\theta^{CoT}(z_{i+1}|x, z_{1\dots i})(j = 1\dots k)$
The formula is:
Essentially, this method applies a CoT reasoning prompt (k times) to generate k different thought branches.

Generating (a sequence of) thought proposals from a single prompt
The formula representation is: $[z^{(1)}, \dots, z^{(k)}] \sim p_\theta^{propose}(z_{i+1}^{(1\dots k)}|s)$

3.1 State Evaluator

The formula for the state evaluator is:

$$V(p_\theta, S)$$

where S represents the sequence of thoughts generated so far. The evaluator acts as a **service for the search algorithm**, helping to determine which thoughts should be explored further and which should be pruned.

Unlike pre-trained or fine-tuned models with pre-existing heuristics, this paper **uses LLMs for evaluation** (e.g., large language models trained for mathematical reasoning or the 24-point game). As observed, this method is highly flexible and effective. The state evaluation consists of two methods:

Independently evaluating a thought: $V(p_\theta, S)(s) \sim p_\theta^{value}(v|s) \forall s \in S$,
A specially designed scoring prompt is used to evaluate the state s, generating either a scalar score v or a qualitative label (such as sure/likely/impossible).

3.2 State Evaluator

(continue)

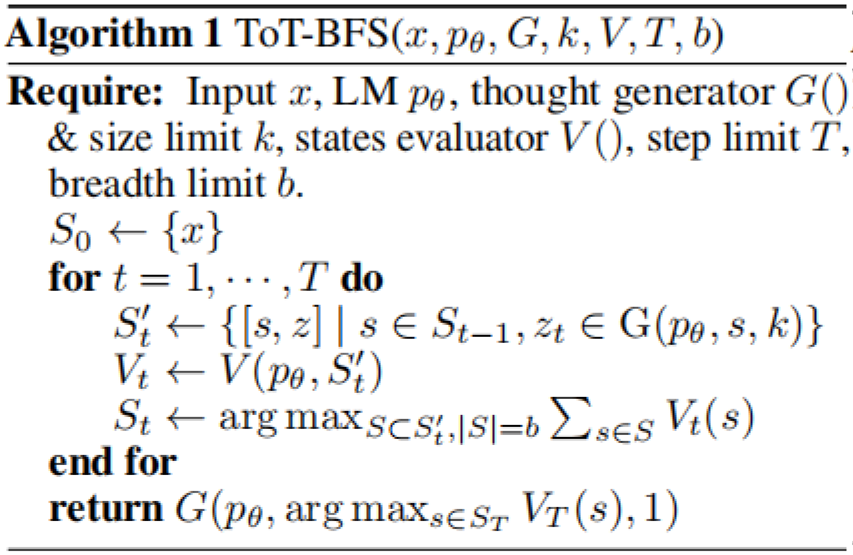
Vote between different states: $V(p_\theta, S)(s) = 1[s = s^*]$

Good states $s^* \sim p_\theta^{vote}(s^*|S)$ represents that s* is determined by voting among all states in S. When the success of a solution cannot be easily determined by scoring individual states (such as evaluating the quality or coherence of an article), comparative evaluation becomes more suitable.

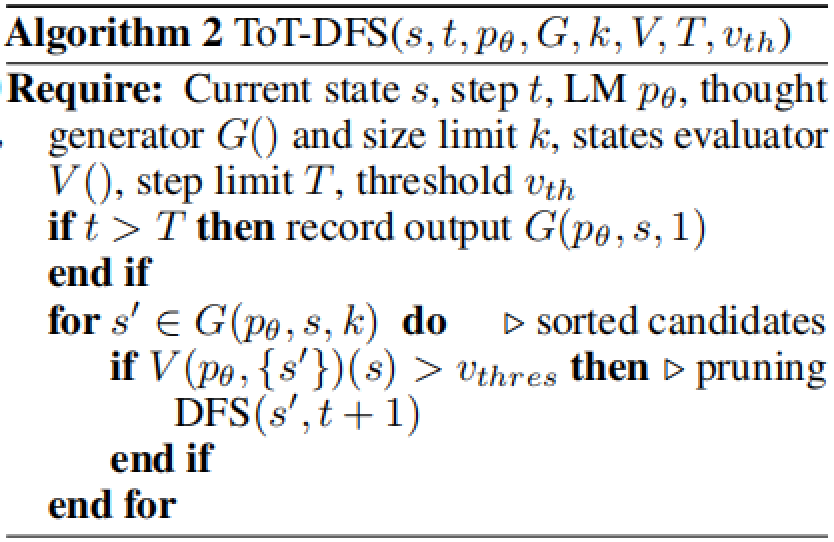
In experiments, this paper inputs prompts multiple times into the evaluator to obtain either the average score or the final voting result.

4.1 Search Algorithm

This paper adopts the most basic BFS (Breadth-First Search) and DFS (Depth-First Search) as the search algorithms.



```
Algorithm 1 ToT-BFS($x, p_\theta, G, k, V, T, b$)
Require: Input $x$, LM $p_\theta$, thought generator $G()$
    & size limit $k$, states evaluator $V()$, step limit $T$,
    breadth limit $b$.
    $S_0 \leftarrow \{x\}$
    for $t = 1, \cdots, T$ do
        $S'_t \leftarrow \{[s, z] \mid s \in S_{t-1}, z_t \in \mathrm{G}(p_\theta, s, k)\}$
        $V_t \leftarrow V(p_\theta, S'_t)$
        $S_t \leftarrow \arg\max_{S \subset S'_t, |S|=b} \sum_{s \in S} V_t(s)$
    end for
    return $G(p_\theta, \arg\max_{s \in S_T} V_T(s), 1)$
```



```
Algorithm 2 ToT-DFS($s, t, p_\theta, G, k, V, T, v_{th}$)
Require: Current state $s$, step $t$, LM $p_\theta$, thought
    generator $G()$ and size limit $k$, states evaluator
    $V()$, step limit $T$, threshold $v_{th}$
    if $t > T$ then record output $G(p_\theta, s, 1)$
    end if
    for $s' \in G(p_\theta, s, k)$ do                 ▷ sorted candidates
        if $V(p_\theta, \{s'\})(s) > v_{thres}$ then   ▷ pruning
            DFS($s', t + 1$)
        end if
    end for
```

4.2 Search Algorithm-BFS

Breadth-First Search (BFS) generates k thoughts S_t ‘ in each round. The state evaluator then assesses all thoughts, producing corresponding scores or classifications V_t. Finally, the top b most promising thoughts based on V_t are retained to form the final thought set S_t. The ultimate goal is to find the thought with the highest evaluation score at the final step.

4.3 Search Algorithm-DFS

Depth-First Search (DFS) generates k thoughts per round, then ranks them using the state evaluator, prioritizing the most promising ones for expansion. If a thought's evaluation result falls below a threshold v_th, pruning occurs, and the search backtracks to the previous state - parent node, then attempts to expand the next child node.

TOT Advantages

In theory, ToT has the following advantages:

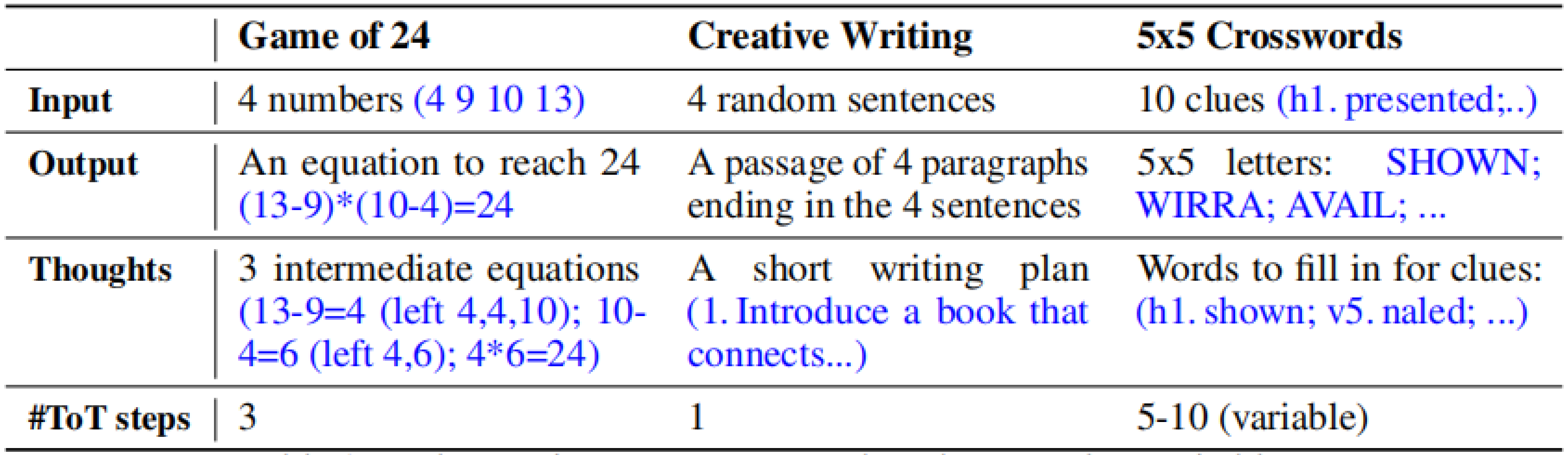
Generality: IO, CoT, and CoT-SC, and even self-correction methods are all special cases of ToT.

Modularity: Each step of ToT is highly modular.

Adaptability: ToT can be adjusted based on different problem types, LLM capabilities, and resource constraints by modifying specific parameters.

Convenience: No additional model training is required

Experiments-result in one table



	Game of 24	Creative Writing	5x5 Crosswords
Input	4 numbers (4 9 10 13)	4 random sentences	10 clues (h1. presented;..)
Output	An equation to reach 24 (13-9)*(10-4)=24	A passage of 4 paragraphs ending in the 4 sentences	5x5 letters: SHOWN; WIRRA; AVAIL; ...
Thoughts	3 intermediate equations (13-9=4 (left 4,4,10); 10-4=6 (left 4,6); 4*6=24)	A short writing plan (1. Introduce a book that connects...)	Words to fill in for clues: (h1. shown; v5. naled; ...)
#ToT steps	3	1	5-10 (variable)

Table 1: Task overview. Input, output, thought examples are in blue.

Experiment1-Game of 24

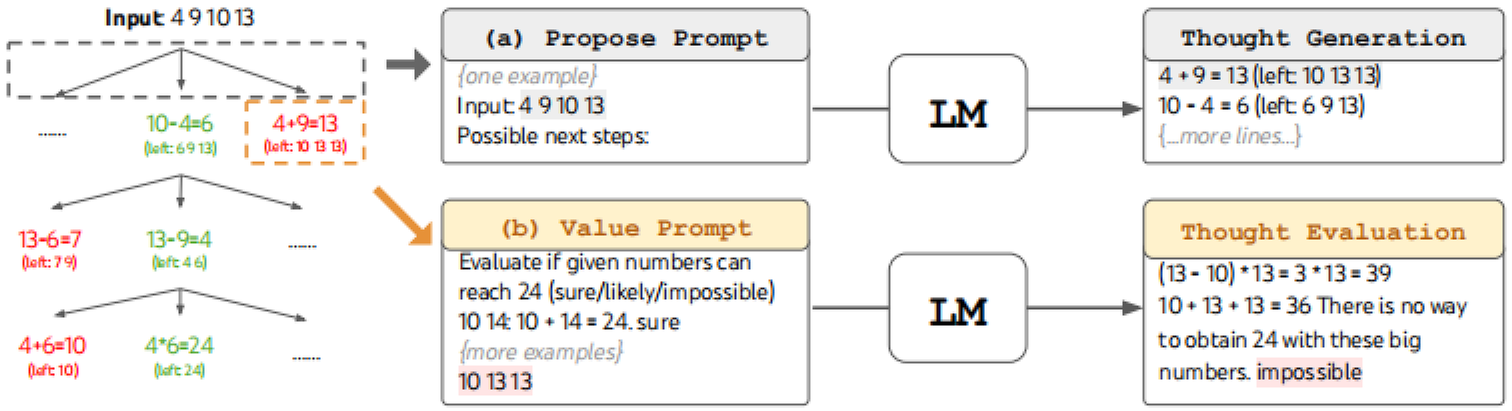


Figure 2: ToT in a game of 24. The LM is prompted for (a) thought generation and (b) valuation.

Task Setup:
Select 100 problems from http://4num.com for testing, plus five additional problems for prompt context (to guide the model on what to do).
A problem is considered successfully completed if the model provides a correct solution.

Experiment1-Game of 24

Baseline Methods:

IO: Use a prompt containing five context examples. For each problem, repeat the input 100 times and calculate the number of successful attempts.

CoT: Expand the five context examples from simple input-output pairs to step-by-step reasoning. Each problem is repeated 100 times, and success rate is measured.

CoT-SC: Same as CoT, but instead of counting the number of successes, CoT-SC determines whether success is more frequent than failure across the 100 attempts.

IO+Refine: Based on IO, if the model makes a mistake, it is prompted to reflect on the error and generate a revised answer—up to 10 refinements per question.

Tree-of-Thoughts (ToT) Approach:

Decomposed into three steps, where each step represents an equation. The model is explicitly informed of the remaining numbers and prompted to generate a set of possible next steps.

BFS search is used, retaining only the five most promising thoughts. The evaluator assesses each thought based on whether it is sure/maybe/impossible to reach 24.

Experiment1-Game of 24-Result

Method	Success
IO prompt	7.3%
CoT prompt	4.0%
CoT-SC (k=100)	9.0%
ToT (ours) (b=1)	45%
ToT (ours) (b=5)	**74%**
IO + Refine (k=10)	27%
IO (best of 100)	33%
CoT (best of 100)	49%

Table 2: Game of 24 Results.

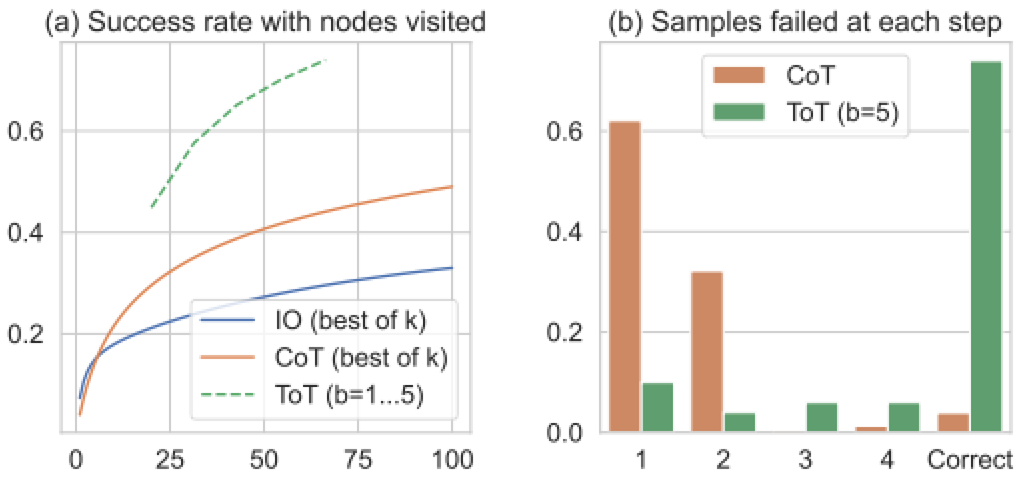


Figure 3: Game of 24 (a) scale analysis & (b) error analysis.

Even when ToT retains only one branch per layer (denoted as ToT b=1), it still achieves a 45% success rate.

The key difference from CoT is that ToT b=1 selects the most promising thought from all generated candidates, with the involvement of an evaluator.

Experiment1-Game of 24-Result

Key Findings from Figure 3:

Figure 3(a): The results indicate that regardless of how many nodes are visited, IO and CoT never outperform ToT (for any b=1...5). (The dashed line represents the aggregated ToT results across different values of b).

Figure 3(b): Computes the failure rate at each step for both CoT and ToT.

In CoT, failure is defined as the step where the thought sequence breaks down.

In ToT, failure occurs when all b generated thoughts fail at a given step.

The analysis reveals that around 60% of CoT samples fail immediately after the first step.This highlights a fundamental issue with left-to-right token-level generation, as CoT lacks the ability to revise or backtrack.

Experiment2-Creative Writing

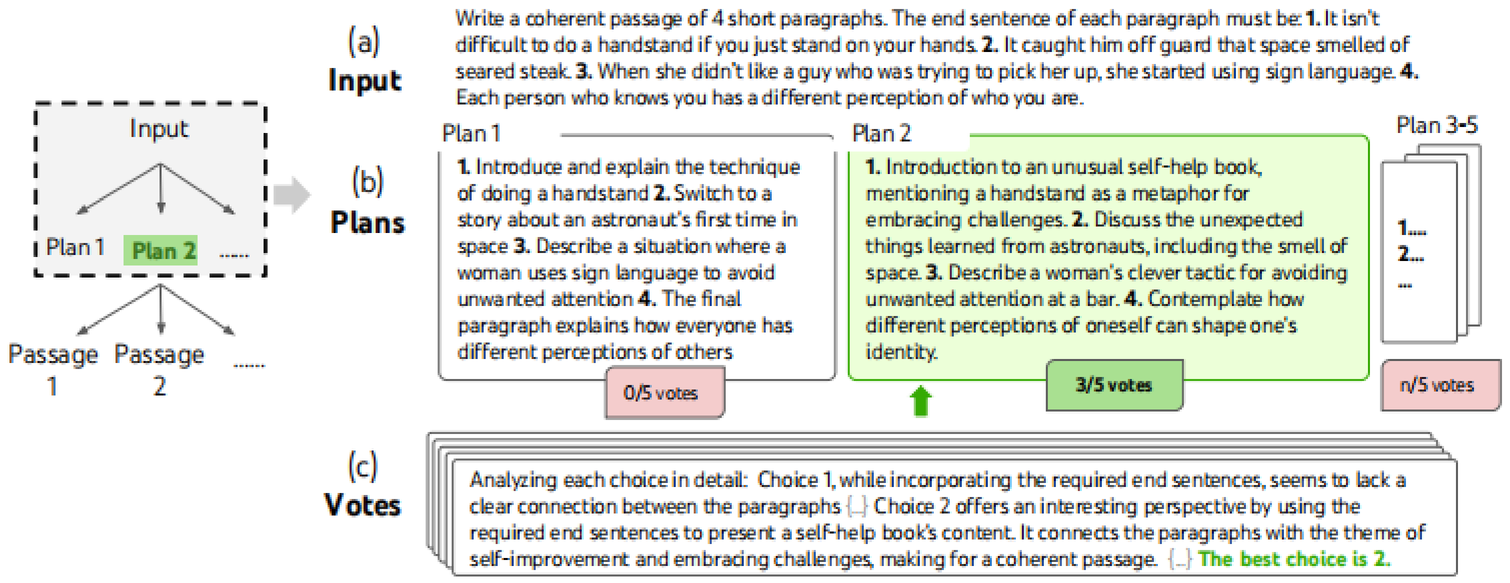

Input
Plan 1
Plan 2
......
Passage 1
Passage 2
......
(a) Input
Write a coherent passage of 4 short paragraphs. The end sentence of each paragraph must be: 1. It isn't difficult to do a handstand if you just stand on your hands. 2. It caught him off guard that space smelled of seared steak. 3. When she didn't like a guy who was trying to pick her up, she started using sign language. 4. Each person who knows you has a different perception of who you are.
(b) Plans
Plan 1
1. Introduce and explain the technique of doing a handstand 2. Switch to a story about an astronaut's first time in space 3. Describe a situation where a woman uses sign language to avoid unwanted attention 4. The final paragraph explains how everyone has different perceptions of others
0/5 votes
Plan 2
1. Introduction to an unusual self-help book, mentioning a handstand as a metaphor for embracing challenges. 2. Discuss the unexpected things learned from astronauts, including the smell of space. 3. Describe a woman's clever tactic for avoiding unwanted attention at a bar. 4. Contemplate how different perceptions of oneself can shape one's identity.
3/5 votes
Plan 3-5
1....
2...
...
n/5 votes
(c) Votes
Analyzing each choice in detail: Choice 1, while incorporating the required end sentences, seems to lack a clear connection between the paragraphs {...} Choice 2 offers an interesting perspective by using the required end sentences to present a self-help book's content. It connects the paragraphs with the theme of self-improvement and embracing challenges, making for a coherent passage. {...} The best choice is 2.

Figure 4: A step of deliberate search in a randomly picked Creative Writing task. Given the input, the LM samples 5 different plans, then votes 5 times to decide which plan is best. The majority choice is used to consequently write the output passage with the same sample-vote procedure.

Task Setup:

100 input prompts are randomly selected from http://randomwordgenerator.com.

Using GPT-4 to score the coherence of the generated text on a scale of 1-10.

Human evaluation, where people compare outputs from different methods (e.g., CoT vs. ToT) and judge which is more coherent for the same input.

THE UNIVERSITY OF ARIZONA®

Experiment2-Creative Writing

Baseline Methods:

This task is zero-shot.

IO: Directly input the prompt and let the LLM generate a constrained coherent passage. Each prompt runs 10 times.

CoT: Before generating the passage, the LLM first outputs a simple plan (intermediate step). Each prompt runs 10 times.

IO+Refine ($k \leq 5$): The LLM assesses whether the generated passage meets the initial coherence constraints. If not, it iteratively refines the passage up to 5 times.

ToT Approach:

The LLM first generates 5 plans.

A state evaluator votes on the plans and selects the one with the highest score.

Based on the selected plan, the LLM generates 5 different passages.

The evaluator votes on the best passage.

Experiment2-Creative Writing-Result

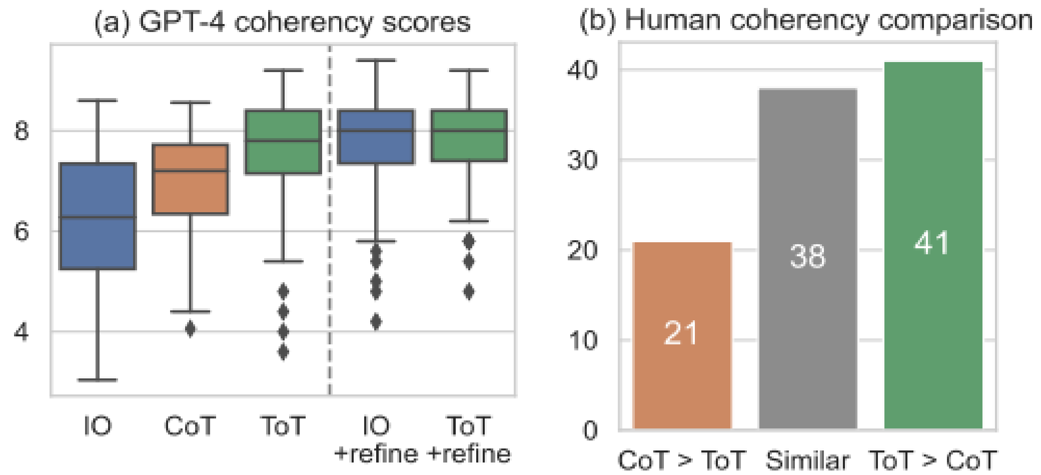


Figure 5: Creative Writing results.

Figure 5(a) shows that ToT-generated passages are rated by GPT-4 as more coherent compared to those generated by IO and CoT.

To address concerns about the rigor of GPT-4's scoring, Figure 5(b) presents human preferences when comparing ToT and CoT-generated passages. The results indicate that humans also favor ToT-generated passages more often.

Experiment3-Crosswords

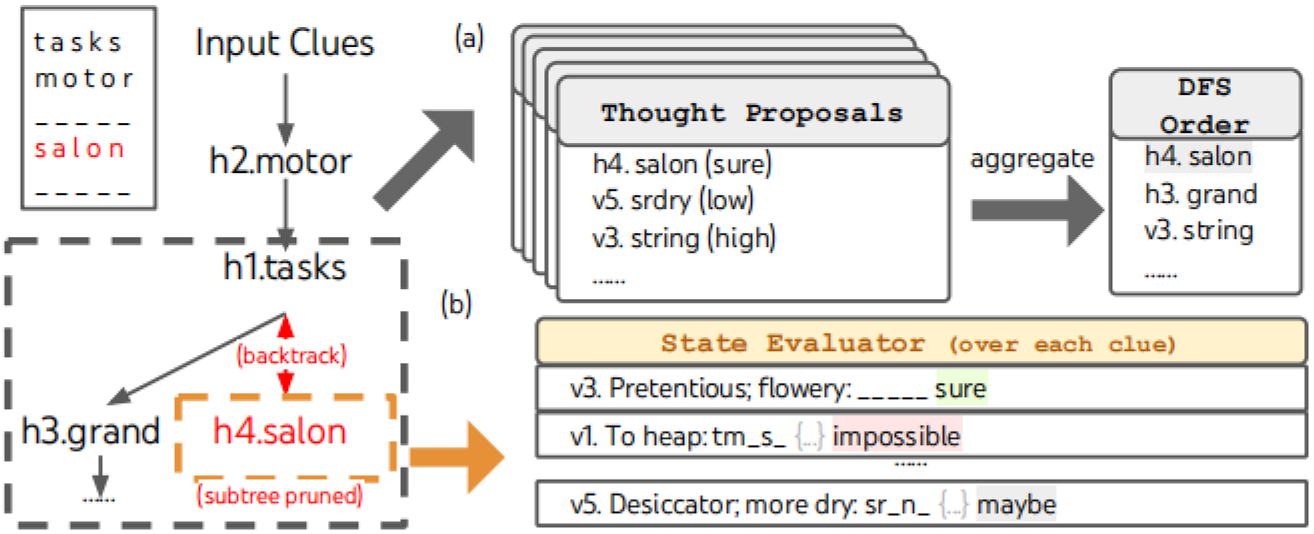


Figure 6: In Mini Crosswords, (a) how thoughts are proposed and aggregated in a priority queue for depth-first search (DFS), and (b) how a state is evaluated based on the possibility of filling in each remaining word clue, and pruned if any remaining clue is deemed not possible to fill by the LM. Then DFS backtracks to the parent state and explore the next promising thought for clue.

It is worth noting that the goal of this experiment is not to solve the crossword puzzle task but rather to explore the limitations of using LMs themselves as a heuristic method.

Experiment3-Crosswords

Task Setup:

20 crossword puzzles were selected from GooBix to evaluate ToT's guidance capability, with 5 additional puzzles used for designing prompts.

For each puzzle, the initial input consists of 5 horizontal word clues and 5 vertical word clues.

The output should be 10 words (filling all 25 letters) to complete the crossword grid.

To assess ToT's guidance effectiveness, this paper evaluates three levels of performance:

The number of correct letters (maximum of 25 per game).

The number of correct words (maximum of 10 per game).

Whether the puzzle is fully completed.

Baseline Methods:

IO: The prompt includes 5 selected input-output examples to guide the LLM. Each prompt runs 10 times, and the average result is taken.

CoT: Builds upon the IO prompt, but includes intermediate steps where previously filled words are shown in sequence: h1, …, h5, v1, …, v5.

Experiment3-Crosswords

ToT Approach:

DFS is used to expand the most promising word clues continuously. If the current state is determined to be unsolvable, backtracking occurs to explore alternative states.

When filling a new word, ToT ensures that previously filled letters are not overwritten.

Each state in ToT completes one word, so the maximum depth of ToT's search tree is 10.

Prompt Design for ToT:

Already filled letters are appended to their corresponding clues in the prompt (e.g., v1. To heap: tm_s_, as shown in the figure).

The designed prompt is input into the LLM 5 times to generate 5 ranked sequences of proposed words and placements (as shown in the Thought Proposals in the figure).

Each proposal is assigned a confidence level, and the results are aggregated to determine the final DFS search order.

State Evaluation (as shown in the figure):

The current filled words and remaining clues are input into the evaluator.

If any clue is determined to be unsolvable, the branch is immediately pruned, and backtracking occurs.

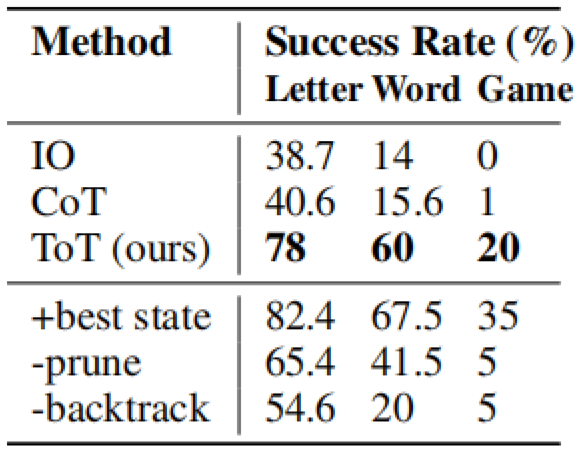
To improve efficiency:

The DFS maximum search depth is limited to 100 steps.

If the search reaches 100 steps without a solution, the deepest explored state is used as the final output.

THE UNIVERSITY OF ARIZONA®

Experiment3-Crosswords-Result



Method	Success Rate (%)		
	Letter	Word	Game
IO	38.7	14	0
CoT	40.6	15.6	1
ToT (ours)	**78**	**60**	**20**
+best state	82.4	67.5	35
-prune	65.4	41.5	5
-backtrack	54.6	20	5

Table 3: Mini Crosswords results.

The results are clear—ToT outperforms both baseline methods at every level, even successfully completing 4 full crossword puzzles.

This approach performs even better than ToT, indicating that the current heuristic (LM evaluator) still has room for improvement.

What happens without pruning?

What happens without backtracking?

"-backtrack" means no backtracking is used. Instead, at each step, the model chooses the most promising clue, overwriting previously filled letters if necessary, and limits the search depth to 20 layers.

Key Takeaways:

The results from "-prune" and "-backtrack" show that pruning and backtracking are essential for successfully solving an entire crossword puzzle.

THE UNIVERSITY OF ARIZONA.

Summary and Conclusion

Limitations:

This paper represents a preliminary exploration of LMs' search capabilities. When investigating LMs' ability to make decisions in real-world tasks, more complex challenges will arise, further testing their capabilities.

Additionally, using ToT to guide LMs in search requires more computational resources (e.g., increased GPT-4 API calls). However, ToT's high modularity allows for trade-offs between search performance and computational cost.

Lastly, this paper directly uses pre-trained LMs. If ToT-style counterfactual decision-making were used for fine-tuning (e.g., considering multiple potential next-paragraph choices instead of simply predicting the next token), it might further enhance LMs' ability to solve reasoning-intensive tasks.

Conclusion:

The associative capabilities of LMs (System 1) can be enhanced by complex reasoning abilities (System 2) through search-based problem-solving approaches.

ToT is a form of prompt engineering that provides a feasible method to restore LMs' reasoning ability to its core logical foundation. In essence, it represents a fusion of LMs' capabilities and traditional search problem-solving techniques.