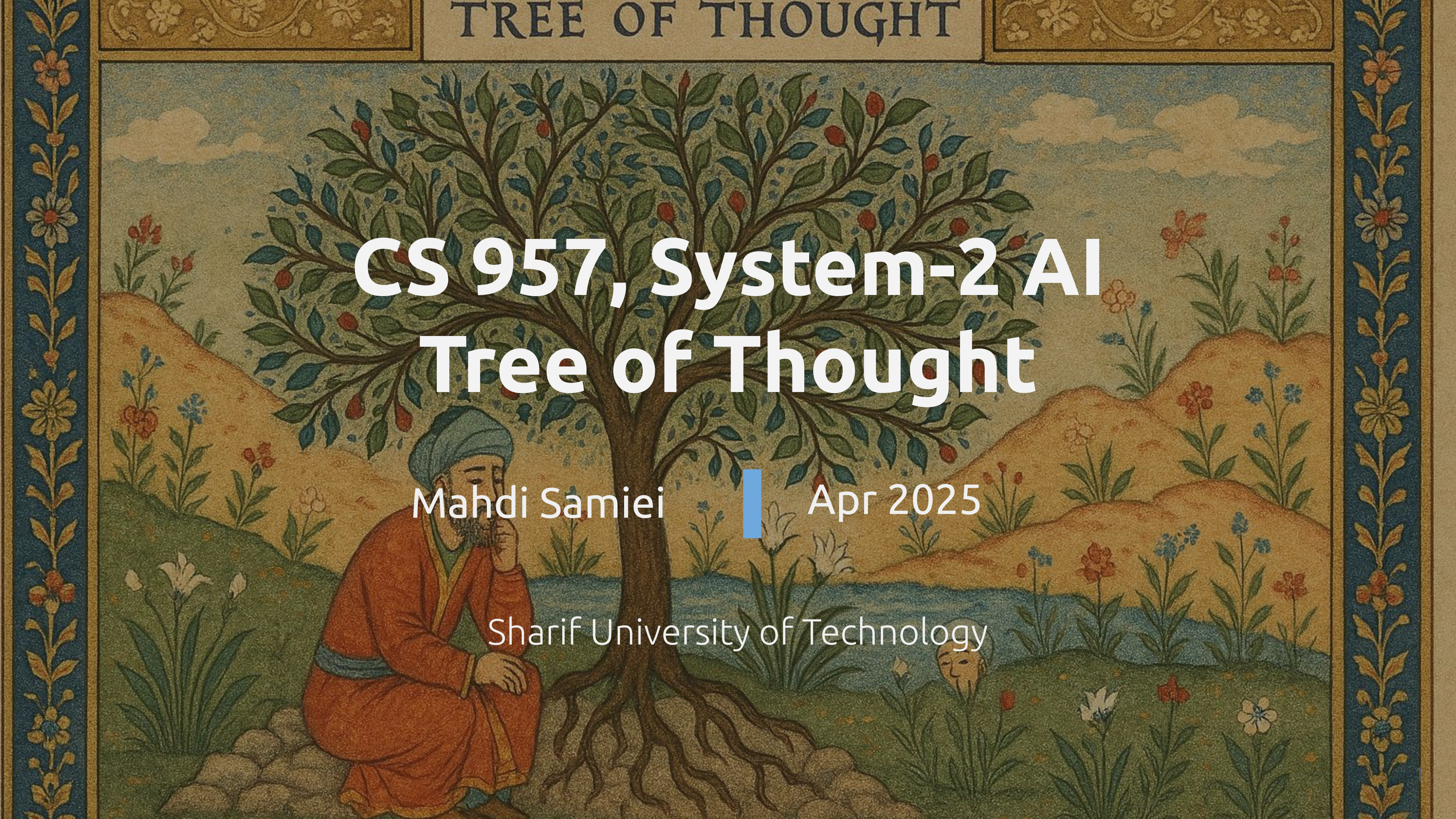


CS 957, System-2 AI
Tree of Thought

Mahdi Samiei

Sharif University of Technology

Apr 2025

Let's start with a problem: Game of 24

🧮 You are given 4 numbers. You must combine them with mathematical operators to create the number 24.

📚 For example, if you are given 4, 9, 10, 13: (13-9) * (10-4) = 24

🤔 Now you solve for 1, 4, 6, and 10 (The first guest gets a coffee)

🎉☕🍰 The winner gets coffee and cake, my treat

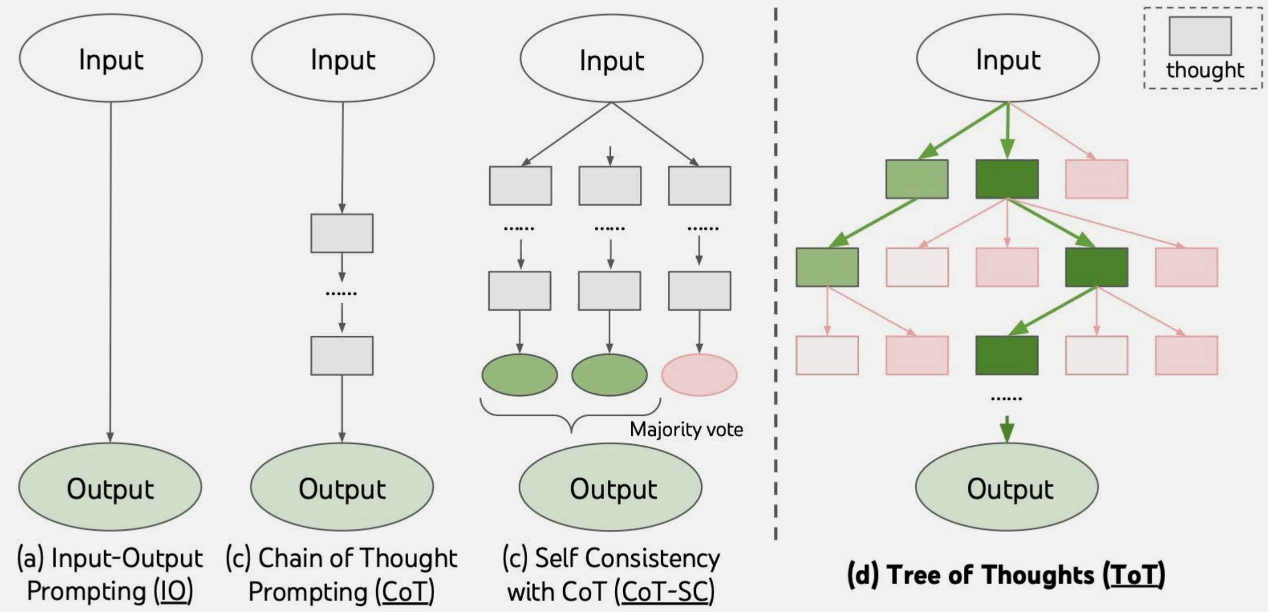
🧩 How Did you solve this?

Tree of Thoughts: Deliberate Problem Solving with Large Language Models

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, Karthik Narasimhan @ Princeton & DeepMind

Backtracking Is All You Need!

Different Types of Solution

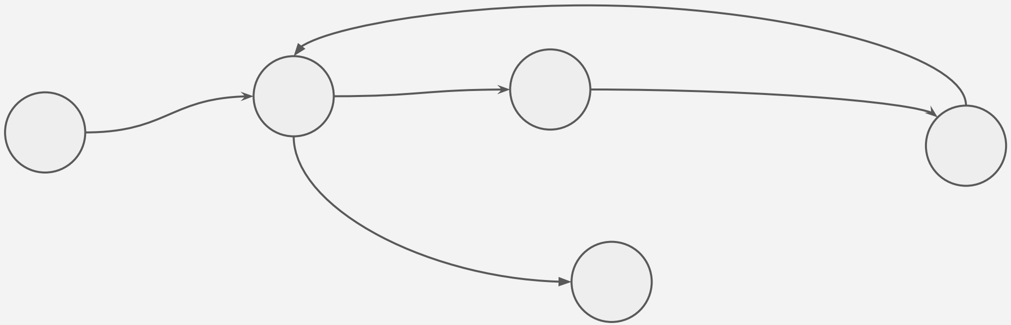


Why LLMs and CoT can't help us?

🔍 are still confined to token-level, left-to-right decision-making processes during inference.

🤔 they can fall short in tasks that require exploration, strategic lookahead, or where initial decisions play a pivotal role (combinatorial optimization)

🎯 sometimes you need to take a stab in the dark to see what to do



What is our questions?

↩️ All of them requiring exploration, strategic lookahead and backtracking

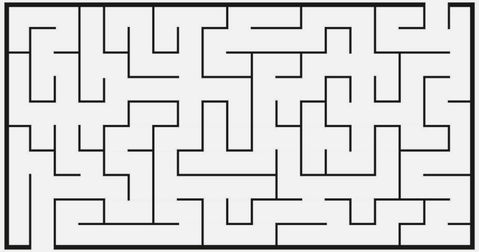
	Game of 24	Creative Writing	5x5 Crosswords
Input	4 numbers (4 9 10 13)	4 random sentences	10 clues (h1. presented;..)
Output	An equation to reach 24 (13-9)*(10-4)=24	A passage of 4 paragraphs ending in the 4 sentences	5x5 letters: SHOWN; WIRRA; AVAIL; ...
Thoughts	3 intermediate equations (13-9=4 (left 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.

Analogy to Grid Worlds

🌀 To me, these problems are like a maze where you're presented with several paths, and some of them lead to dead ends. When you're in a fog, you have to explore different routes, and if you hit a dead end, you retrace your steps and try a different path from your last known good position

↩️ If all paths don't end in God, then spiritual backtracking becomes necessary

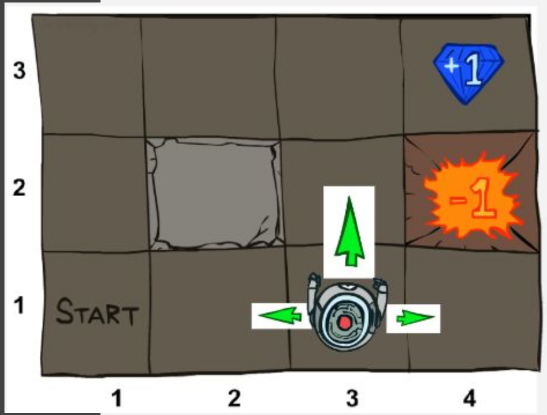


We Are Reinventing AI CS-447 Course in Terms of Reasoning

maybe you can guess what we'll see ....

Types of Problems

- ✅ Search (dfs, bfs)
- ✅ Search with Cost (ucs, A*)
- ✅ Local Search (Random Walk, Hill Climbing, ...)
- ✅ Constraint Satisfaction Problems (Backtracking)
- ✅ Search with Other Agents (Minimax)
- ❓ Non-Deterministic Search



Formalizing Thoughts vs Tokens

What is the difference between tokens and thoughts?

We first formalize some existing methods that use large language models for problem-solving, which our approach is inspired by and later compared with. We use p_θ to denote a pre-trained LM with parameters θ, and **lowercase letters** $x, y, z, s, \cdots$ **to denote a language sequence**, i.e. $x = (x[1], \cdots, x[n])$ where each $x[i]$ is a token, so that $p_\theta(x) = \prod_{i=1}^{n} p_\theta(x[i]|x[1...i])$. We use uppercase letters $S, \cdots$ to denote a collection of language sequences.

Introduce Tree of Thoughts

🤔 Problems of LLM and CoT:

- 🛣️ Locally, they do not explore different continuations within a thought process
- Globally, they do not incorporate any type of planning, lookahead, or backtracking to help evaluate these different options

🔀 Allows LMs to explore multiple reasoning paths over thoughts

🌳 frames any problem as a search over a tree

🔢 each node is a state $s = [x, z_{1 \cdots i}]$ representing a partial solution with the input and the sequence of thoughts so far

We should Answer These Questions in ToT Framework

1. How to decompose the intermediate process into thought steps (state space)
2. How to generate potential thoughts from each state (successor function)
3. How to heuristically evaluate states (heuristic function)
4. What search algorithm to use (priority of expanding)

1. Thought decomposition

⚔️ CoT vs ToT: CoT samples thoughts coherently without explicit decomposition, ToT leverages problem properties to design and decompose intermediate thought steps

🔤 Depending on different problems: a thought could be a couple of words (Crosswords), a line of equation (Game of 24), or a whole paragraph of writing plan (Creative Writing)

🛠️ Problem Engineering

📏 What characteristics should a good thought have?

- 📏 "small" enough so that LMs can generate promising and diverse samples (not a whole book!)
- 📏 "big" enough so that LMs can evaluate its prospect toward problem solving (not a token)

2. Thought generator

2. Thought generator $G(p_\theta, s, k)$**.** Given a tree state $s = [x, z_{1 \cdots i}]$, we consider two strategies to generate k candidates for the next thought step:

(a) **Sample** i.i.d. thoughts from a CoT prompt (Creative Writing, Figure $\boxed{4}$): $z^{(j)} \sim p_\theta^{CoT}(z_{i+1}|s) = p_\theta^{CoT}(z_{i+1}|x, z_{1 \cdots i})$ $(j = 1 \cdots k)$. This works better when the thought space is rich (e.g. each thought is a paragraph), and i.i.d. samples lead to diversity;

(b) **Propose** thoughts sequentially using a "propose prompt" (Game of 24, Figure $\boxed{2}$, Crosswords, Figure $\boxed{6}$): $[z^{(1)}, \cdots, z^{(k)}] \sim p_\theta^{propose}(z_{i+1}^{(1 \cdots k)} \mid s)$. This works better when the thought space is more constrained (e.g. each thought is just a word or a line), so proposing different thoughts in the same context avoids duplication.

3. State evaluator

📊 The state evaluator evaluates the progress they make towards solving the problem, serving as a heuristic for the search algorithm

🧠 Typical heuristics: programmed, learned

🤖 Third Alternative: LLM as Evaluator!

❓ But How to use LLM?

3. State evaluator: Value Strategy

(a) **Value** each state independently: $V(p_\theta, S)(s) \sim p_\theta^{value}(v|s)\ \forall s \in S$, where a value prompt reasons about the state s to generate a scalar value v (e.g. 1-10) or a classification (e.g. sure/likely/impossible) that could be heuristically turned into a value. The basis of such evaluative reasoning can vary across problems and thought steps. In this work, we explore evaluation via few *lookahead* simulations (e.g. quickly confirm that 5, 5, 14 can reach 24 via 5 + 5 + 14, or "hot_l" can mean "inn" via filling "e" in "_") plus commonsense (e.g. 1 2 3 are too small to reach 24, or no word can start with "tzxc"). While the former might promote "good" states, the latter could help eliminate "bad" states. Such valuations do not need to be perfect, and only need to be approximately helpful for decision making.

3. State evaluator: Vote (Rank) Strategy

(b) **Vote** across states: $V(p_\theta, S)(s) = \mathbb{1}[s = s^*]$, where a "good" state $s^* \sim p_\theta^{vote}(s^*|S)$ is voted out based on deliberately comparing different states in S in a vote prompt. When problem success is harder to directly value (e.g. passage coherency), it is natural to to instead compare different partial solutions and vote for the most promising one. This is similar in spirit to a "step-wise" self-consistency strategy, i.e. cast "which state to explore" as a multi-choice QA, and use LM samples to vote for it.

4. Search algorithm

Algorithm 1 ToT-BFS($x, p_\theta, G, k, V, T, b$)

Require: Input x, LM p_θ, 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_θ, 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

Benefits of ToT

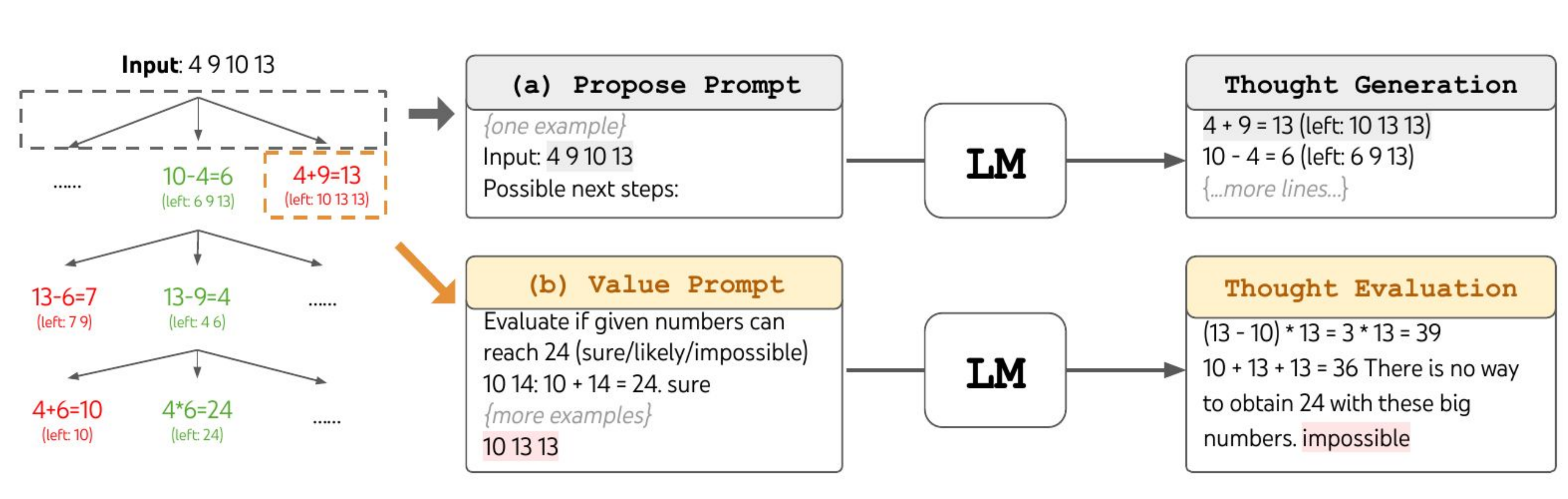
(1) **Generality**. IO, CoT, CoT-SC, and self-refinement can be seen as special cases of ToT

(2) **Modularity**. The base LM, as well as the thought decomposition, generation, evaluation, and search procedures can all be varied independently

(3) **Adaptability**. Different problem properties, LM capabilities, and resource constraints can be accommodated

(4) **Convenience.** No extra training is needed, just a pre-trained LM is sufficient

Experiments, Game of 24



Experiments, Game of 24

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.

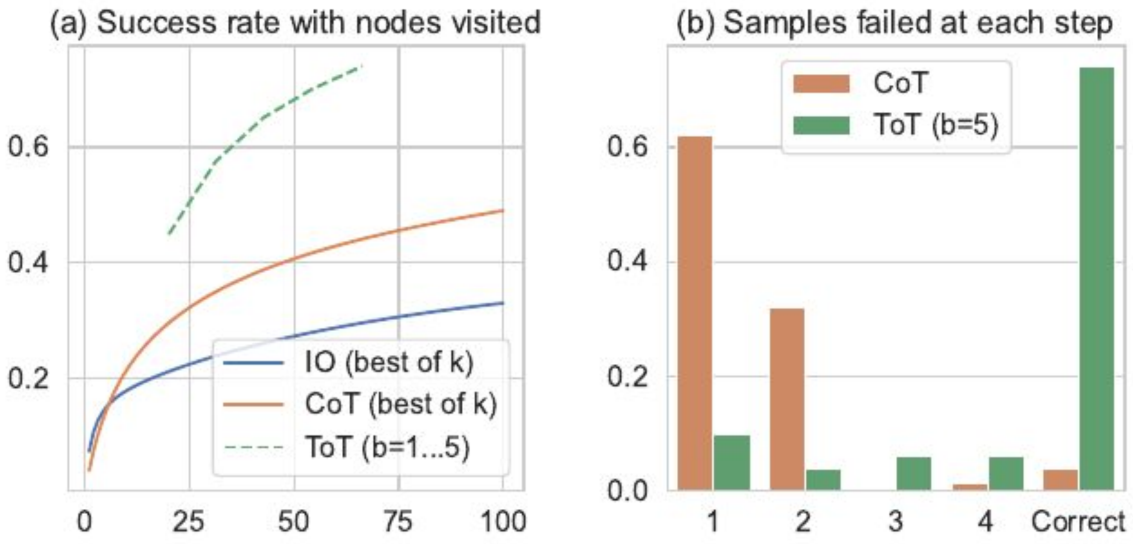


Figure 3: Game of 24 (a) scale analysis & (b) error analysis.

Experiments, Creative writing

Next, we invent a creative writing task where the input is 4 random sentences and the output should be a coherent passage with 4 paragraphs that end in the 4 input sentences respectively. Such a task is open-ended and exploratory, and challenges creative thinking as well as high-level planning.

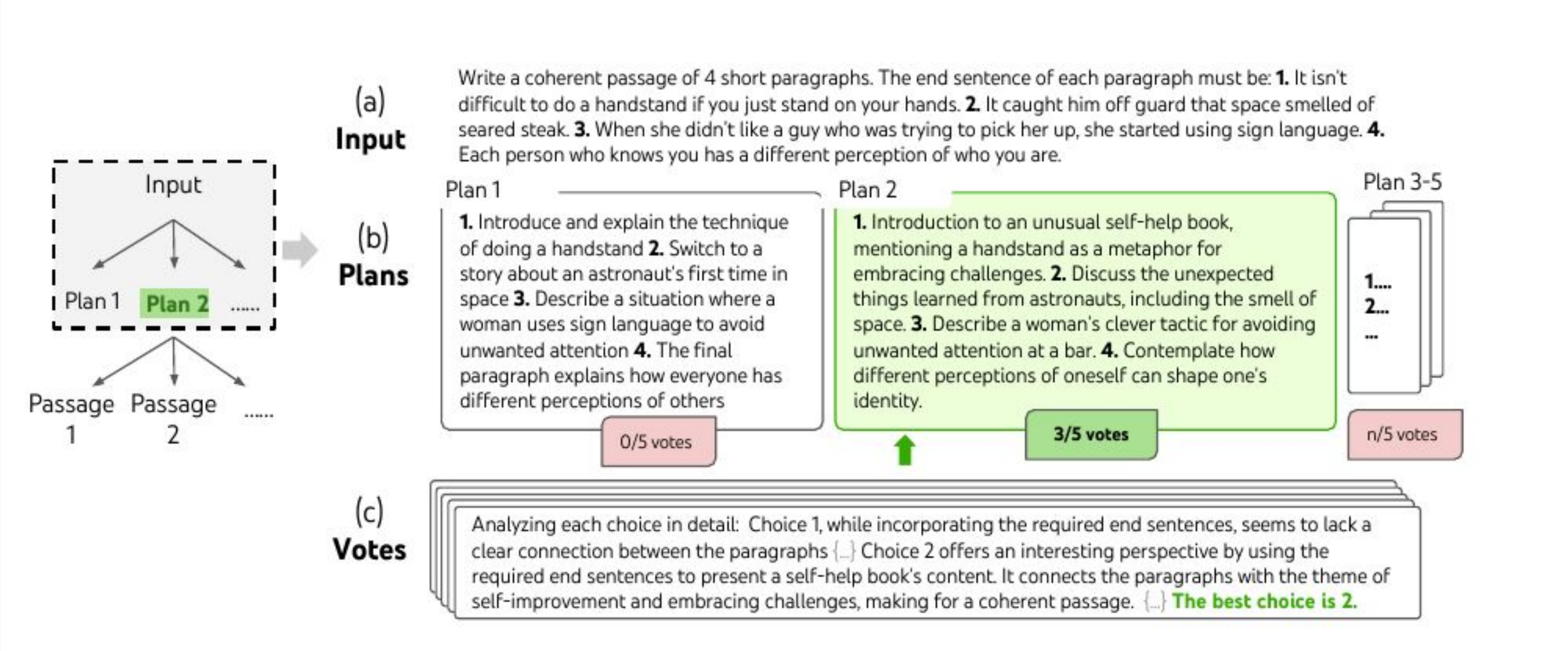
Task setup. We sample random sentences from randomwordgenerator.com to form 100 inputs, and there is no groundtruth passage for each input constraint. As we find that GPT-4 can follow the input constraints most of the time, we focus on evaluating passage coherency in two ways: using a GPT-4 zero-shot prompt to provide a 1-10 scalar score, or using human judgments to compare pairs of outputs from different methods. For the former, we sample 5 scores and average them for each task output, and we find these 5 scores usually consistent, with a standard deviation of around 0.56 on average across outputs. For the latter, we employ a subset of the authors in a blind study to compare the coherency of CoT vs. ToT generated passage pairs, where the order of passages is random flipped over 100 inputs.

Experiments, Creative writing

Baselines. Given the creative nature of the task, both IO and CoT prompts are zero-shot. While the former prompts the LM to directly generate a coherent passage given input constraints, the latter prompts the LM to first make a brief plan then write the passage, i.e. the plan serves as the intermediate thought step. We generate 10 IO and CoT samples per task. We also consider an iterative-refine ($k \leq 5$) method on top of a random IO sample for each task, where the LM is conditioned on input constraints and the last generated passage to decide if the passage is already "perfectly coherent", and if not generate a refined one.

ToT setup. We build a ToT with depth 2 (and only 1 intermediate thought step) — the LM first generates $k = 5$ plans and votes for the best one (Figure 4), then similarly generate $k = 5$ passages based on the best plan then vote for the best one. Here the breadth limit $b = 1$, as only one choice is kept per step. A simple zero-shot vote prompt ("analyze choices below, then conclude which is most promising for the instruction") is used to sample 5 votes at both steps.

Experiments, Creative writing



Experiments, Creative writing

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.

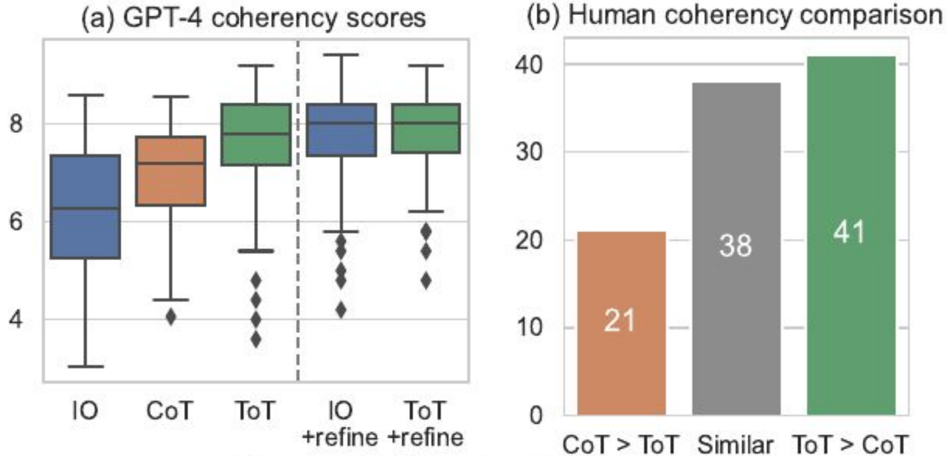


Figure 5: Creative Writing results.

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.

Remember backtracking, you will soon see related to pure-RL methods ...