INTRODUCTION TO NATURAL LANGUAGE PROCESSING

Test-Time Scaling & Inference Compute

The LLM Lifecycle (Recap)

1. Pretraining

Learning next-token prediction on web corpora (Result: Base Model).

2. Instruction Tuning (SFT)

Learning to follow instructions (Result: SFT Model).

3. Alignment (RLHF/DPO)

Learning human preferences (Result: Chat Model).

The Inference Phase

Inference

The phase where the model generates text in response to a user prompt.

Standard View

Inference is just a "read-out" of the model's frozen weights.

New Paradigm

Inference is a computational process where we can trade **extra compute** for **better answers**.

This is called **Test-Time Compute** or **Inference Scaling**.

Definition: Test-Time Compute

Test-Time Compute: The amount of computational resources (FLOPs, time, tokens) expended during the generation of an answer.

System 1 Thinking (Fast)

Immediate, intuitive response.

(Standard LLM generation)

System 2 Thinking (Slow)

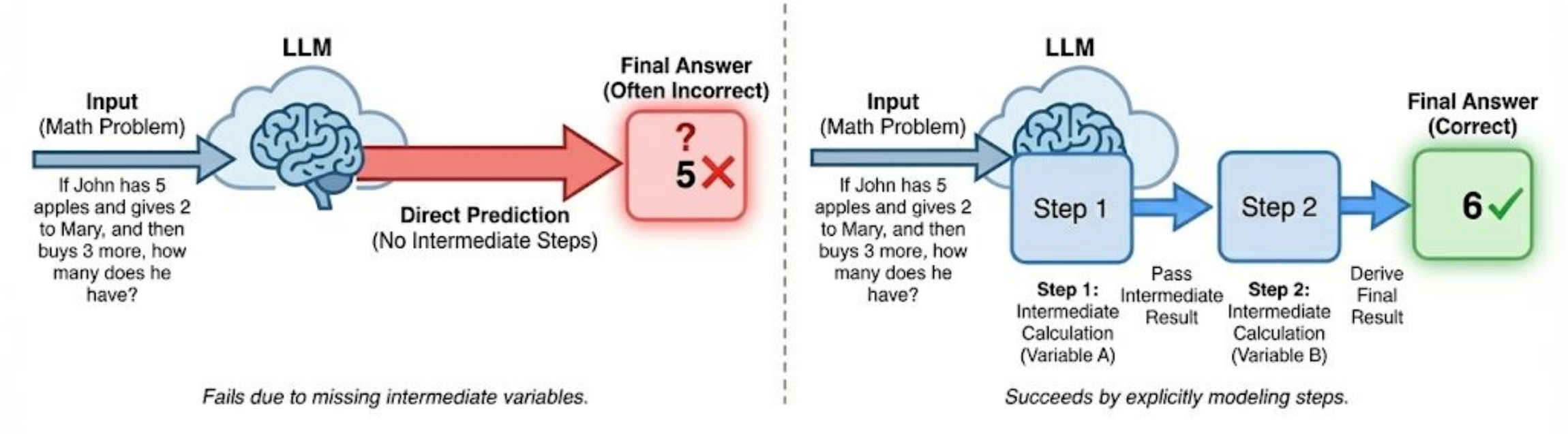
Deliberate, step-by-step reasoning.

(Test-Time Compute)

Goal: Increase accuracy on complex reasoning tasks by allowing the model to "think" before answering.

The Challenge of Complex Reasoning

- LLMs struggle with multi-step reasoning problems (e.g., math word problems, logic puzzles).
- **Why?** Standard generation forces the model to predict the final answer immediately or very quickly.
- If the reasoning requires intermediate variables or steps, a direct prediction often fails.



Chain-of-Thought (CoT) Prompting

Definition

A technique where the model is prompted to generate intermediate reasoning steps before giving the final answer.

Mechanism

Instead of just outputting "The answer is X", the model outputs:

```
"Step 1..., Step 2..., Therefore X."
```

Impact

Significant improvement on reasoning benchmarks.

Standard Prompting vs. CoT

Standard Prompting

Input: "Roger has 5 balls. He buys 2 cans of 3 balls each. How many balls does he have?"

Output: "The answer is 11."

(Risk: Guessing/Memorization)

CoT Prompting

Input: "Roger has 5 balls. He buys 2 cans of 3 balls each. How many balls does he have?"

Output: "Roger started with 5 balls. 2 cans of 3 balls each is 6 balls. 5 + 6 = 11. The answer is 11."

Why CoT Works (Intuition)

Decomposition

Breaks a hard problem into smaller, easier problems (e.g., multiplication first, then addition).

Working Memory

The generated tokens serve as an external scratchpad. The model can attend to its own previous steps.

Error Correction

Seeing an intermediate result might steer the model away from a bad final guess.

Implementing CoT (Few-Shot)

Method

Provide examples (demonstrations) in the prompt that include reasoning chains.

Prompt Example

```
Q: The cafeteria had 23 apples. Used 20.
Bought 6. How many?
A: Had 23. Used 20 → 3 left. Bought 6 →
3+6=9. Answer is 9.

Q: Roger has 5 balls...
A: [Model generates reasoning based on
pattern]
```

Implementing CoT (Zero-Shot)

Surprisingly, you don't always need examples.

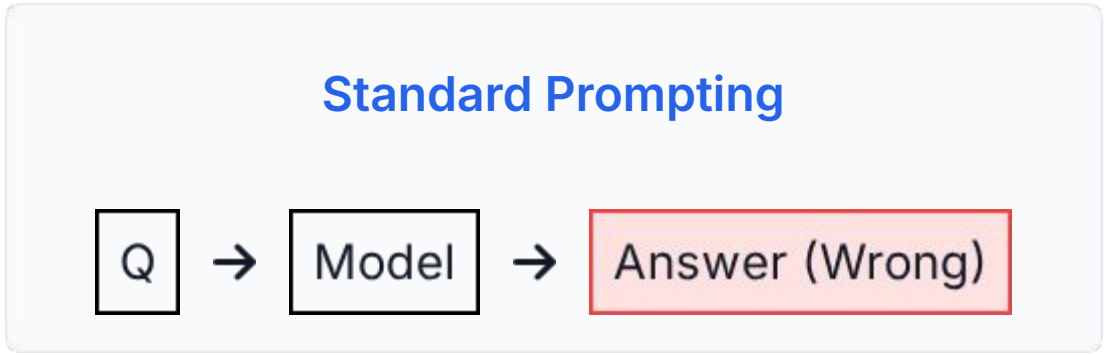
Zero-Shot CoT (Kojima et al., 2022)

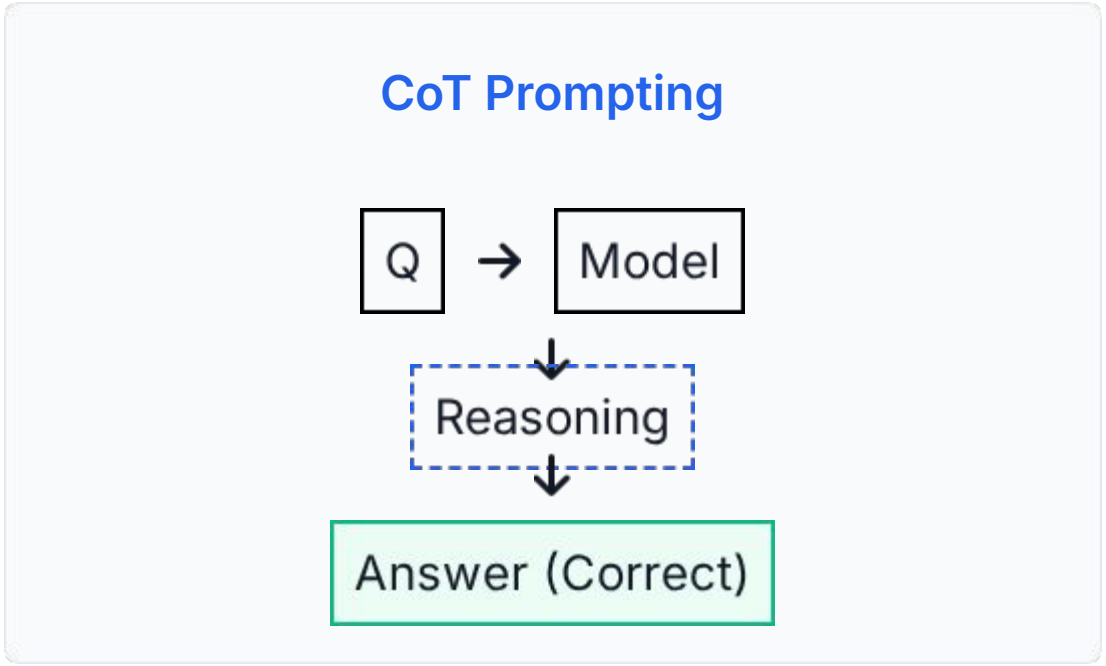
Simply append the magic phrase:

```
"Let's think step by step"
```

Result: The model spontaneously generates a reasoning chain and improves performance.

Visualizing the Improvement





Example: Math Word Problem (GSM8k)

- **Dataset:** GSM8k (Grade School Math 8k).
- **Task:** Multi-step arithmetic problems.

Standard

Large models often fail.

~20%

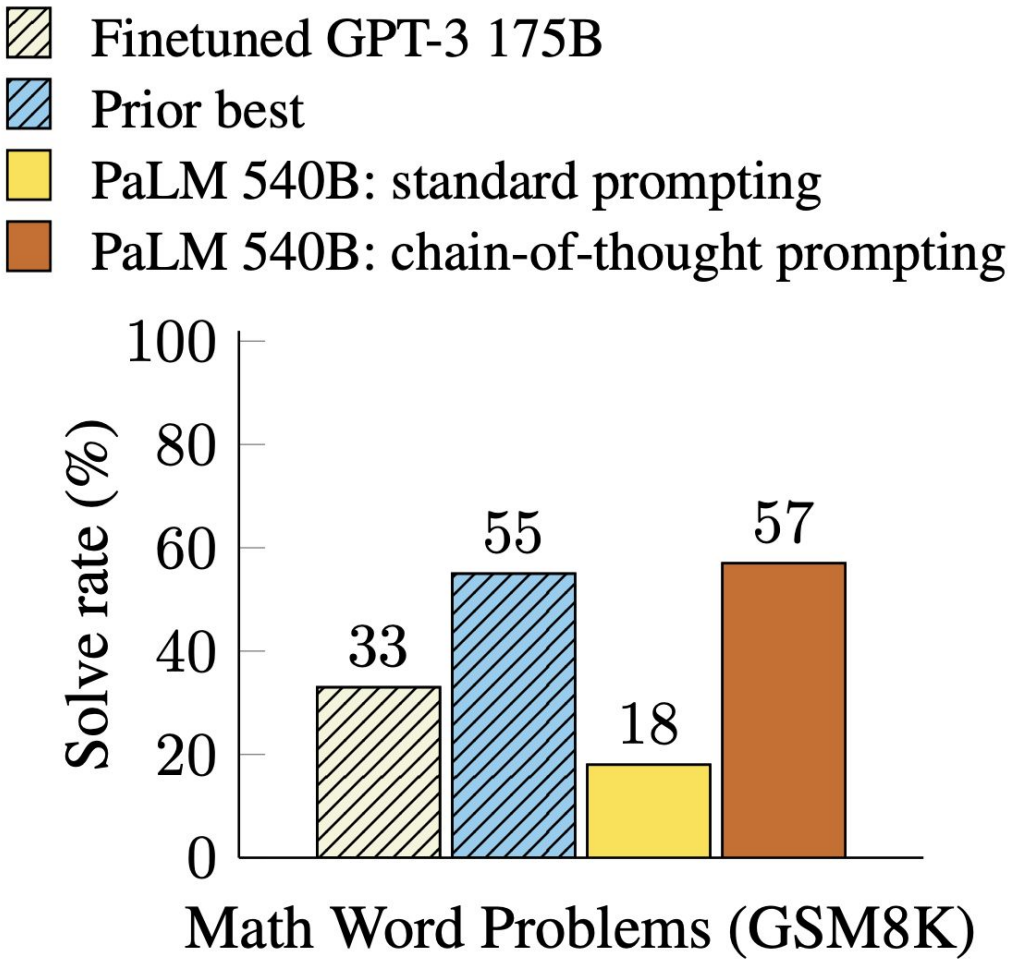
(Accuracy)

CoT

Accuracy jumps significantly.

~50%+

(Accuracy)



Reference: Wei et al. (2022).

Limitations of Pure CoT

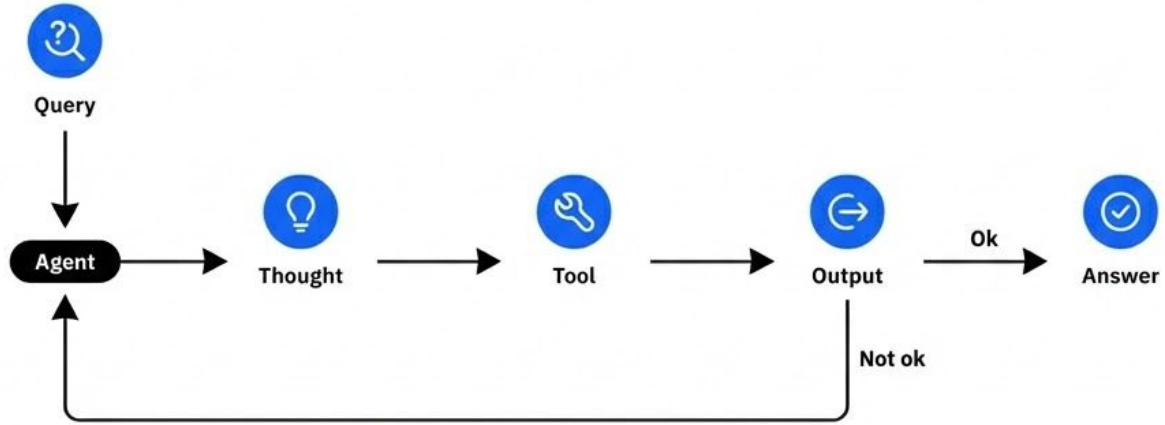
- CoT is powerful but **limited to the model's internal knowledge**.
- **Problem: Hallucination.**
- If the model doesn't know a fact (e.g., "What is the price of Bitcoin right now?"), CoT will just hallucinate reasoning steps.
- **Solution:** We need to connect reasoning to the external world.

ReAct: Synergizing Reasoning and Acting

ReAct (Yao et al., 2022): A framework that interleaves reasoning traces (thoughts) with task-specific actions.

Concept

- **Reasoning:** Helps the model create, maintain, and adjust plans.
- **Acting:** Allows the model to interface with external sources (Search APIs, Wikipedia).



ReAct Loop: Interleaving internal thought with external action.

The ReAct Loop

```
Thought: "I need to search for the current CEO of Apple."
Action: Search[Apple CEO]
Observation: (Output from API) "Tim Cook is the CEO..."
Thought: "Now I need to find his age."
Action: Search[Tim Cook Age]
Answer: "Tim Cook is 63 years old."
```

Comparing CoT vs. ReAct

Standard CoT

Question: "Who is the CEO of Apple?"

Reasoning: "Steve Jobs founded Apple. He died. Tim Cook took over. So it is Tim Cook."

Risk: Outdated knowledge.

ReAct

Question: "Who is the CEO of Apple?"

Thought: "I should check if there is a new CEO."

Action: Search Google.

Observation: "Tim Cook..."

Empirical Results for ReAct

- **Dataset:** HotpotQA (Multi-hop question answering) and Fever (Fact verification).
- **Result:**
  - ReAct outperforms standard CoT on tasks requiring factual lookups.
  - ReAct reduces hallucination by grounding thoughts in external observations.
- **Trade-off:** ReAct requires more tokens (higher latency) and external tool integration.

The Problem with One-Shot Reasoning

Even with CoT or ReAct, models make mistakes.

Problem: Error Propagation

Once a model outputs a wrong step, it often doubles down on it.

Human Analogy

When humans solve hard problems, we don't just write the answer once. We draft, check, and revise.

Reflexion: Reinforcement via Verbal Reflection

Reflexion (Shinn et al., 2023): A framework for equipping agents with dynamic memory and self-reflection capabilities.

1. Try

Agent attempts a task (using CoT or ReAct).

2. Evaluate

A heuristic or binary classifier detects failure.

3. Reflect

The agent generates a verbal "reflection" on why it failed.

4. Repeat: The agent tries again, conditioned on its own reflection.

The Reflexion Process

Trial 1

Agent generates wrong answer.

Feedback

"The answer is incorrect."

Self-Reflection

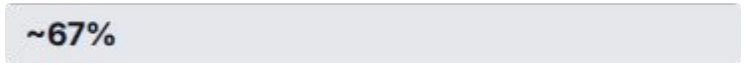
"I failed because I multiplied incorrectly in step 2. I should use a calculator tool next time or double check the math."

Trial 2

Reflexion Results

HumanEval (Coding)

GPT-4 (Base):



GPT-4 + Reflexion:

Why?

Coding is a perfect domain for Reflexion because compilers provide perfect feedback (Error Messages). The model effectively "debugs" its own reasoning trace.

Reflexion vs. Chain-of-Thought

Reflexion is "Iterative Correction" while ToT is "Non-linear Search.

Chain-of-Thought

A single linear pass of reasoning.

If the train leaves the tracks, it crashes.

Reflexion

A loop. It allows the model to "backtrack" and correct its own policy without updating weights.

Memory: Utilizes a long-term memory (buffer of reflections) to improve over time.

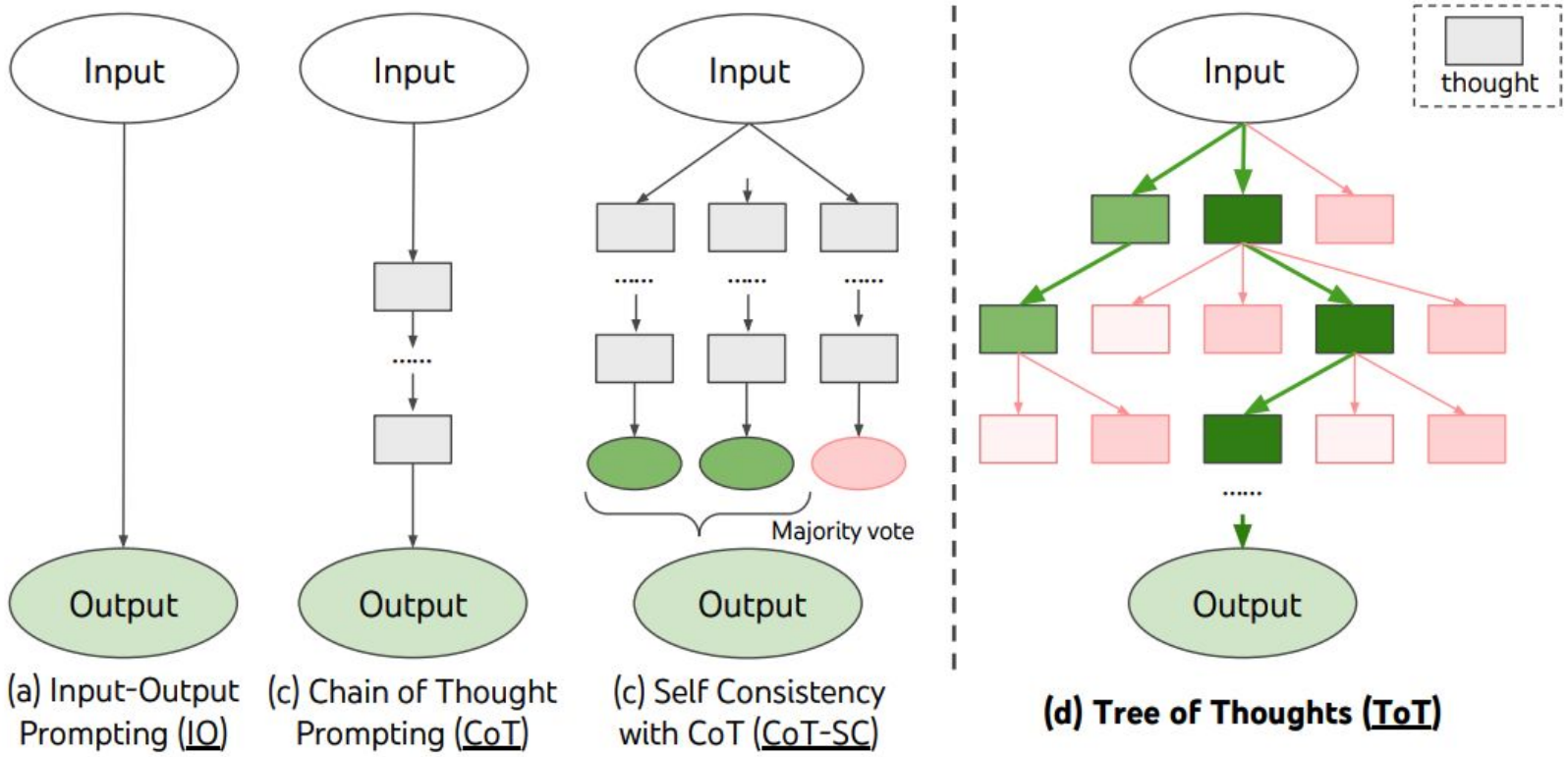
From Chains to Trees

- CoT is linear.
- ReAct is linear (with loops).

Tree of Thoughts (ToT)

(Yao et al., 2023)

Generalizes CoT to a search tree. Instead of generating one thought, generate multiple possible next steps.



ToT Algorithm

1. Decomposition

Break the problem into steps.

2. Generation

At each step, generate candidates.

3. Evaluation

Self-evaluate each candidate (e.g., "Is this step promising?").

4. Search

Use BFS (Breadth-First Search) or DFS (Depth-First Search) to explore the tree.

ToT Example: Game of 24

Task: Given 4 numbers (e.g., 4, 9, 10, 13), use arithmetic to get 24.

Standard CoT fails here because it is "greedy"—it commits to a dead end and cannot backtrack. ToT succeeds because it explores branches and effectively "prunes" bad math paths before committing to a final answer.

CoT

Fails often (4% success).

Commits to a bad path early (e.g., "4+9=13...").

ToT

Generates branches.

- **Branch 1:** "4+9=13..." (Evaluate: Bad path).
- **Branch 2:** "13-10=3..." (Evaluate: Promising).

Success: ~74%

CoT Prompt Template

Format: Question: [Input Question] Let's think step by step. [Reasoning Steps] Therefore, the answer is [Final Answer].

ReAct Prompt Template

Format: Question: [Input] Thought 1: [Reasoning] Action 1: [Tool Call] Observation 1: [Tool Output] Thought 2: [Reasoning] ... Action N: Finish[Answer]

Reflexion Prompt Template

Format: Question: [Input] Attempt 1: [Wrong Answer] Feedback: [Error Message] Reflection: I failed because... I should... Attempt 2: [New Reasoning based on Reflection]

The Cost of Reasoning

All these methods (CoT, ReAct, Reflexion, ToT) share a common theme:

Trading Compute for Intelligence.

We spend more tokens at inference time to get a better answer.

Scaling Compute vs. Scaling Data

Training Scaling

Train on more data = Better model.

(Fixed cost paid once).

Inference Scaling

Think for longer = Better answer.

(Variable cost paid per query).

Optimization: For hard problems, it is often cheaper to use a smaller model that "thinks" (CoT) than a larger model that answers immediately.

Reasoning Methods Comparison

Method	Reasoning Type	External Tools?	Self-Correction?	Cost
Standard	Implicit	No	No	Low
CoT	Linear	No	No	Medium
ReAct	Linear + Action	Yes	No	High
Reflexion	Loop	Yes/No	Yes	Very High
ToT	Branching	Yes/No	Yes	Highest

System 2 Distillation

Can we have our cake and eat it too?

Idea

Use a slow, reasoning-heavy method (e.g., ReAct or ToT) to generate good answers.

Distillation

Finetune a model on just the final (Question, Answer) pairs.

Result

Compressing the "System 2" thinking back into "System 1" weights for faster inference later.

Limitations of Reasoning

Faithfulness

Does the model actually use the reasoning to find the answer, or is it just post-hoc rationalization?

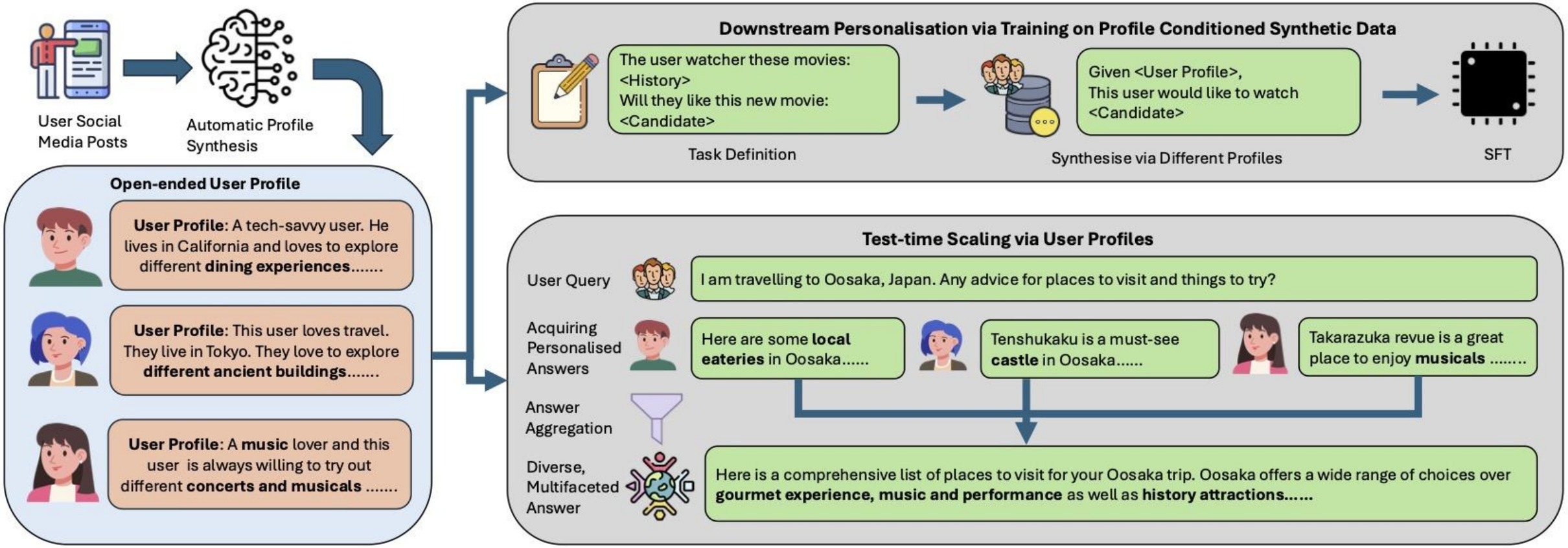
Error Propagation

In CoT, one wrong step ruins the whole chain.

Looping

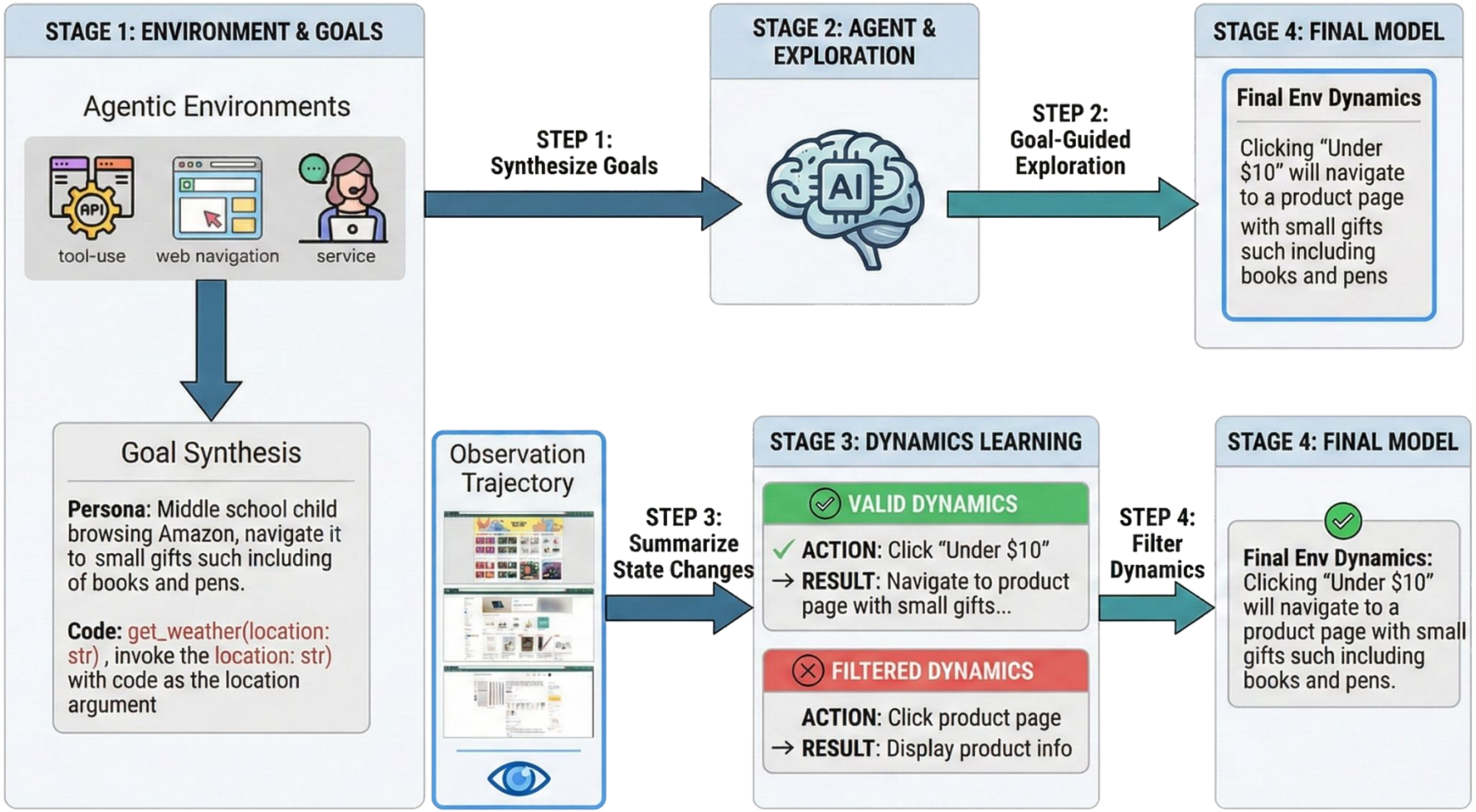
ReAct agents can get stuck in infinite loops of searching for the same term.

Recent Results in Test-time Scaling



Li 2026, *Realistic User Profiles for Train-time Personalization and Test-time Mixture-of-Profiles Scaling*.

Recent Results in Test-time Scaling



Chen 2026, *Test-time Adaptation for LLM Agents via Environment Interaction*.

Summary (1/2)

- **Test-Time Compute:** We can improve performance after training by changing how we generate text.
- **Chain-of-Thought (CoT):** Linear reasoning steps improve math and logic.
- **ReAct:** Interleaves reasoning with external actions to solve knowledge-heavy tasks and reduce hallucination.

Summary (2/2)

- **Reflexion:** Adds a feedback loop where the model critiques its own past attempts to self-correct.
- **Tree of Thoughts:** Treats reasoning as a search problem (BFS/DFS) over the space of possible thoughts.
- **Scaling:** We are moving from "Static Inference" to "Dynamic Reasoning" where compute is proportional to problem difficulty.

The Future of Inference

Future LLM systems will likely be "Agents" rather than just chatbots.

Dynamic Decision Making

- "Do I know this answer?" (Direct Answer).
- "Do I need to think?" (CoT).
- "Do I need to use a tool?" (ReAct).
- "Did I make a mistake?" (Reflexion).

References

- **CoT:** Wei et al. (2022). "Chain-of-Thought Prompting Elicits Reasoning in Large Language Models".
- **Zero-Shot CoT:** Kojima et al. (2022). "Large Language Models are Zero-Shot Reasoners".
- **ReAct:** Yao et al. (2022). "ReAct: Synergizing Reasoning and Acting in Language Models".
- **Reflexion:** Shinn et al. (2023). "Reflexion: Language Agents with Verbal Reinforcement Learning".