Prompting and Reasoning

Robert Minneker

2026-02-17

Sources

Content derived from: J&M Ch. 7 (Large Language Models)

Today's Lecture:

- In-context learning: zero-shot and few-shot prompting
- Chain-of-thought and structured reasoning strategies
- Structured prompting for controlled outputs
- Agentic workflows: planning, tool use, and failure modes

Learning Objectives

By the end of this lecture, you will be able to:

1. Explain how in-context learning enables task adaptation without parameter updates
2. Design effective zero-shot and few-shot prompts following principled design guidelines
3. Apply chain-of-thought prompting and compare advanced reasoning strategies
4. Use structured prompting to enforce schema-constrained outputs
5. Describe agentic LLM architectures and identify their failure modes

Roadmap for our Final Push

BUILDING

TODAY

Inference-Time Control

Prompting, decoding, self-consistency

Training-Time Control

SFT/PEFT, continued pretraining, pref. tuning

System-Time Augmentation

RAG, tools, agents

UNDERSTANDING & GOVERNING

Understanding

Interpretability, causal methods

Measuring

Evaluation, contamination, protocols

Governing & Shipping

Safety engineering, deployment constraints

Running Example: Medical Triage Assistant

Throughout this lecture, we'll develop prompts for a concrete scenario: **a medical triage assistant that helps patients describe symptoms and routes them to appropriate care**.

Scenario:

Setting: Hospital front-desk chatbot for a busy emergency department
Goal: Classify patient urgency (emergency / urgent / routine), suggest department
Constraints: Must never diagnose, always recommend professional evaluation, handle anxiety sensitively
Key challenge: Get this right with prompting alone — no finetuning budget

- As we cover each prompting strategy, we'll ask: *how would you prompt the triage assistant?*

Part 1: In-Context Learning

State Change: Learning Without Training

Finetuning adapts model weights. In-context learning adapts model behavior through the prompt alone, keeping all parameters frozen.

In-context learning (ICL) enables task adaptation by conditioning on natural language prompts

The model processes a prompt p and input x to produce output y with frozen weights θ:

$$y = \text{LLM}_\theta(p, x)$$

Finetuning

Updates parameters θ using labeled data

Expensive, persistent

In-Context Learning

Steers frozen model via prompt context

Cheap, ephemeral

- The prompt acts as an inductive bias, tapping into latent knowledge from pretraining

Zero-shot prompting provides only a task description, while few-shot adds demonstrations

Prompt Type	What the Model Sees	Example
Zero-shot	Task description only	`Translate to French: 'cat'`
Few-shot	k input-output demonstrations	`cat → chat, dog → chien, house → ?`

- Few-shot prompting provides k exemplars so the model infers the pattern from demonstrations
- The model is not updating parameters — it performs statistical pattern completion over the context

Prompt wording, formatting, and example order significantly affect model predictions

The output distribution is sensitive to prompt structure:

$$P(y \mid x, p_1) \neq P(y \mid x, p_2) \quad \text{if} \quad p_1 \neq p_2$$

Wording

"Summarize" vs. "In one sentence, summarize the main argument for a graduate audience"

Formatting

Whitespace, punctuation, and template consistency alter tokenization

Example Order

The ordering of few-shot exemplars introduces recency bias

> **Prompt sensitivity is real:** Zhao et al. (2021) showed that simply reordering few-shot examples can swing GPT-3 accuracy from **near-chance to near-SOTA** on the same benchmark — a 30+ percentage point difference from permutation alone.

- Diverse, balanced exemplars calibrate outputs and reduce unwanted bias

Small prompt changes produce dramatically different triage decisions

Bad Prompt

```
"Classify this patient's urgency: chest pain after exercise"
```

→ "Routine — likely muscle soreness from exercise"

Improved Prompt

```
"You are a medical triage assistant. Classify urgency as EMERGENCY, URGENT, or ROUTINE. Err on the side of caution. Always recommend professional evaluation.\n\nPatient: chest pain after exercise"
```

→ "URGENT — Chest pain requires professional evaluation regardless of suspected cause. Recommend immediate medical assessment."

Think-Pair-Share: Choosing a Prompting Strategy

For our triage assistant, would you use zero-shot or few-shot prompting? Consider:

- **Latency:** How fast must the response be? (few-shot uses more tokens)
- **Accuracy:** How much does task performance improve with examples?
- **Label space:** Is {EMERGENCY, URGENT, ROUTINE} clear enough without examples?
- **Calibration:** Do you need the model to express uncertainty?

Discuss with a neighbor for 2 minutes, then share your recommendation.

Part 2: Chain-of-Thought and Structured Reasoning

State Change: Beyond Direct Answers

Standard prompting asks for a final answer. Chain-of-thought prompting elicits intermediate reasoning steps, dramatically improving performance on complex tasks.

Chain-of-thought prompting decomposes complex tasks into explicit intermediate steps

CoT biases the model toward generating a *reasoning trace* rather than a single direct answer:

Example: Arithmetic Word Problem

Prompt:

```
"If Alice has 3 apples and buys 2 more,
how many does she have? Let's think
step by step."
```

→

LLM Output:

```
"Alice starts with 3. She buys 2 more.
3 + 2 = 5. Alice has 5 apples."
```

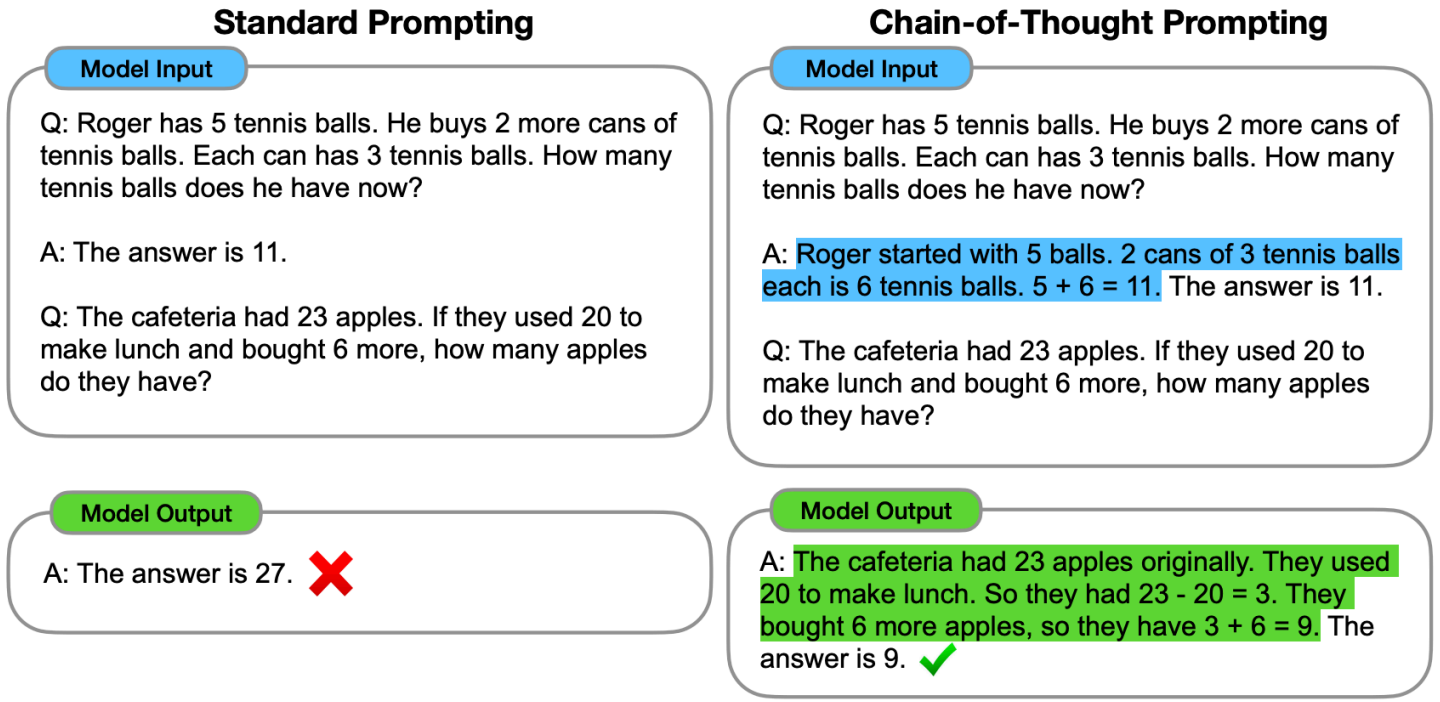


Figure 1: Chain-of-thought prompting enables large language models to tackle complex arithmetic, commonsense, and symbolic reasoning tasks. Chain-of-thought reasoning processes are highlighted.

Standard vs. chain-of-thought prompting from Wei et al. (2022)

- The model breaks complex problems into tractable subproblems, avoiding shallow heuristics
- CoT is most effective for tasks requiring multi-step logic: arithmetic, symbolic reasoning, multi-hop QA

Zero-shot CoT adds a trigger phrase while few-shot CoT provides worked examples

Zero-shot CoT

Append "*Let's think step by step.*" to the prompt

Leverages the model's learned prior for multi-step explanations

Few-shot CoT

Include exemplars with full reasoning traces

Provides direct inductive bias via demonstration

- On GSM8K, GPT-3 accuracy rises from below 20% (direct) to over 50% with zero-shot CoT
- Even a single trigger phrase dramatically changes model behavior due to learned priors

Adding “Let’s think step by step” produces dramatic accuracy gains across reasoning tasks (Kojima et al., 2022)

	Arithmetic					
	SingleEq	AddSub	MultiArith	GSM8K	AQUA	SVAMP
zero-shot	74.6/**78.7**	**72.2/77.0**	17.7/22.7	10.4/12.5	22.4/22.4	58.8/58.7
zero-shot-cot	**78.0/78.7**	69.6/74.7	**78.7/79.3**	**40.7/40.5**	**33.5/31.9**	**62.1/63.7**
	Common Sense		Other Reasoning Tasks		Symbolic Reasoning	
	Common SenseQA	Strategy QA	Date Understand	Shuffled Objects	Last Letter (4 words)	Coin Flip (4 times)
zero-shot	**68.8/72.6**	12.7/**54.3**	49.3/33.6	31.3/29.7	0.2/-	12.8/53.8
zero-shot-cot	64.6/64.0	**54.8**/52.3	**67.5/61.8**	**52.4/52.9**	**57.6**/-	**91.4/87.8**

- A single phrase appended to the prompt consistently doubles accuracy on math reasoning benchmarks
- The effect is most pronounced on tasks requiring multi-step arithmetic

Chain-of-thought helps reasoning tasks but not simple pattern matching

CoT Helps

✓ Multi-step arithmetic
✓ Multi-hop reasoning
✓ Logical deduction
✓ Complex triage decisions

CoT Doesn't Help (or Hurts)

✗ Simple factual recall
✗ Single-label classification
✗ Pattern matching / extraction
✗ Tasks where the answer is obvious

- **For our triage assistant:** CoT helps when symptoms are ambiguous ("chest pain + shortness of breath + recent travel" requires reasoning about multiple possibilities)

⚠ **GPT-4 and the Bar Exam**

GPT-4 scored in the 90th percentile on the Bar Exam (vs. GPT-3.5's 10th percentile). **Takeaway:** Benchmarks can overstate reasoning ability — multiple-choice format may not reflect true open-ended legal reasoning.

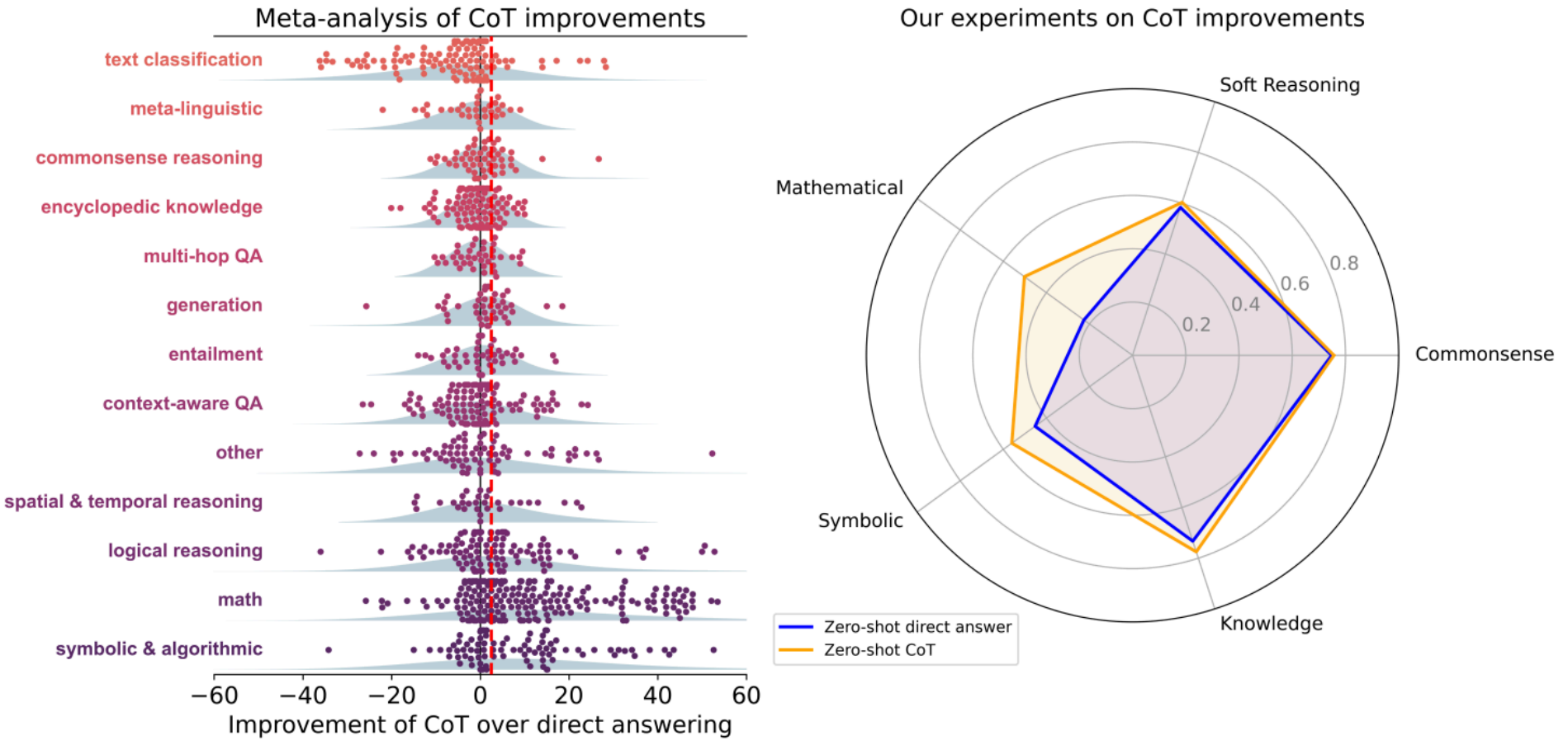


Figure 1: Left: meta-analysis of CoT literature; each point is a reported delta of CoT over direct answering for some (LLM, task) pair. Right: average performance of using zero-shot CoT v.s. direct answer prompts across five general reasoning categories, covering 20 datasets with 14 LLMs evaluated on each. In both sets of results, math and other kinds of symbolic reasoning are the domains that consistently see substantial improvements from CoT (red dotted line indicates the mean improvement from CoT across experiments).

Self-consistency, tree-of-thought, and least-to-most extend basic chain-of-thought

Self-Consistency

Sample N chains, take majority vote

$\hat{y} = \text{mode}\{f(C_i)\}$

Tree-of-Thought

Branch-and-explore reasoning paths with backtracking

BFS over reasoning states

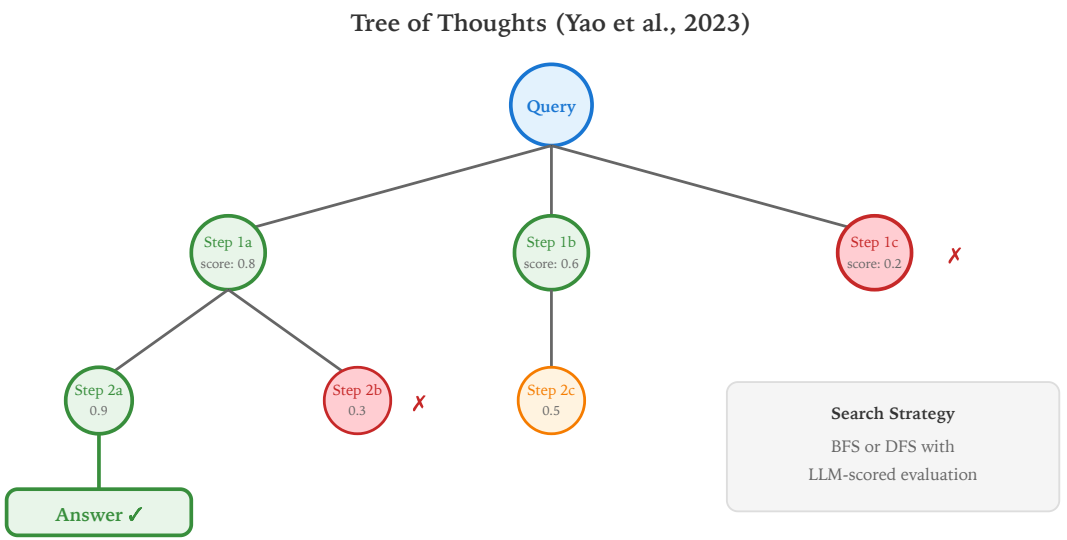
Least-to-Most

Decompose into ordered sub-questions, solve bottom-up

Hierarchical decomposition

- Self-consistency reduces stochasticity by aggregating over multiple reasoning chains
- These methods trade compute for accuracy — more reasoning paths yield more reliable answers

Tree-of-Thought explores multiple reasoning branches and backtracks from dead ends



- Unlike linear CoT, ToT can backtrack from unpromising paths and explore alternatives
- The LLM serves as both the generator (proposing steps) and the evaluator (scoring them)

Choosing a reasoning strategy involves compute-accuracy tradeoffs

Strategy	Compute Cost	Robustness	Best For
Basic CoT	1×	Low	Simple multi-step problems
Self-Consistency	N×	High	When you need reliable answers (triage!)
Tree-of-Thought	N×M×	Highest	Creative/open-ended exploration
Least-to-Most	K×	Medium	Hierarchical decomposition

- For our triage assistant, **self-consistency** is the right choice: sample 5 triage classifications, take the majority — a wrong classification has real consequences

Concept Check

Why does self-consistency improve over a single chain-of-thought? Under what conditions might it fail? Think about the relationship between answer diversity and aggregation quality.

Discussion (2 min)

Consider a multi-hop science question: "What element has the highest electronegativity, and what compounds does it commonly form?" Would you use basic CoT, self-consistency, or least-to-most prompting? Why?

Hallmark Discoveries in Prompting and Reasoning

In-Context Learning as a Scale-Era Behavior

Large language models can perform new tasks from a few demonstrations in the prompt — no gradient updates required. This emergent capability fundamentally changed how practitioners interact with models.

Chain-of-Thought as Eliciting Latent Computation Traces

Wei et al. (2022) showed that prompting models to "show their work" unlocks reasoning capabilities already present in the weights — intermediate tokens serve as a scratchpad that transforms single-step prediction into multi-step computation.

Zero-Shot CoT Triggers as Steering Priors

Kojima et al. (2022) demonstrated that a single phrase — "Let's think step by step" — activates latent reasoning without any demonstrations, revealing that models encode reasoning priors that can be unlocked through simple prompt design.

Part 3: Structured Prompting Techniques

State Change: Controlling Output Format

Reasoning strategies improve answer quality. Structured prompting goes further by constraining the format of model outputs, enabling integration with downstream systems.

Structured prompting constrains LLM outputs to machine-readable formats like JSON or XML

By specifying output schemas, we reduce output entropy and enable robust downstream parsing:

$$f_{\text{LLM}}(x; P) \in \mathcal{S}$$

where $\mathcal{S}$ is the set of valid outputs (e.g., all well-formed JSON objects).

Example: Information Extraction

Prompt:

```
"Extract fields and reply in JSON: {
'name': ..., 'date': ..., 'location':
... }"
```

→

Output:

```
{"name": "Alice", "date": "2026-02-17",
"location": "Seattle"}
```

- Schema-constrained prompts support automation: API calls, data validation, pipeline integration

Prompt templates and role instructions reinforce consistent output structure

Role Prompting

"You are an information extraction system. Output results in valid JSON only."

Sets behavioral context

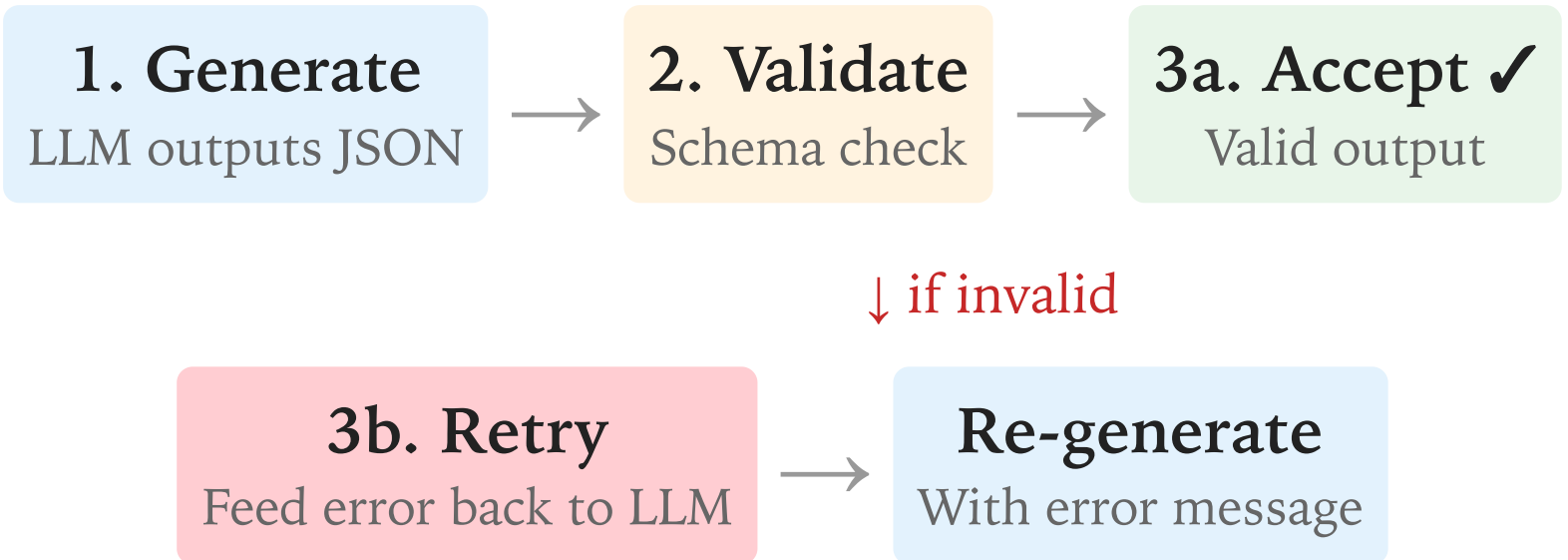
Template Prompting

"Return your answer as: <result><answer>...</answer></result>"

Defines exact output shape

- System instructions (role prompts) establish context and reinforce schema adherence
- Clear templates improve reliability across repeated calls and diverse inputs

The generate-validate-retry pattern ensures reliable structured outputs




```
Triage output schema:
{"urgency": "EMERGENCY|URGENT|ROUTINE",
 "reasoning": "string",
 "recommended_department": "string",
 "disclaimer": "Always seek professional medical advice"}
```

Retrieval-augmented prompting injects external context to ground LLM responses in facts

$$\text{Prompt}(q) = \text{concat}\left(\mathcal{R}(q),\ q\right)$$

- RAG mitigates knowledge cutoff by dynamically providing up-to-date or domain-specific information
- **Key prompting pattern for RAG:** inject evidence, instruct to cite, handle context limits
- We'll cover RAG architecture in depth in a later lecture — today, focus on how the *prompt* changes

Concept Check

A biomedical QA system retrieves 20 PubMed abstracts for each query, but the LLM's context window is 4,096 tokens. What strategies could you use to handle this? What are the trade-offs?

Part 4: Agentic Workflows

State Change: From Answering to Acting

Prompting strategies so far produce text. Agentic workflows enable LLMs to reason, plan, and invoke external tools to complete tasks in the real world.

The ReAct framework interleaves reasoning traces with tool-calling actions

LLMs alternate between generating intermediate thoughts and invoking external tools:

$$\pi : (x, h) \to (r, a)$$

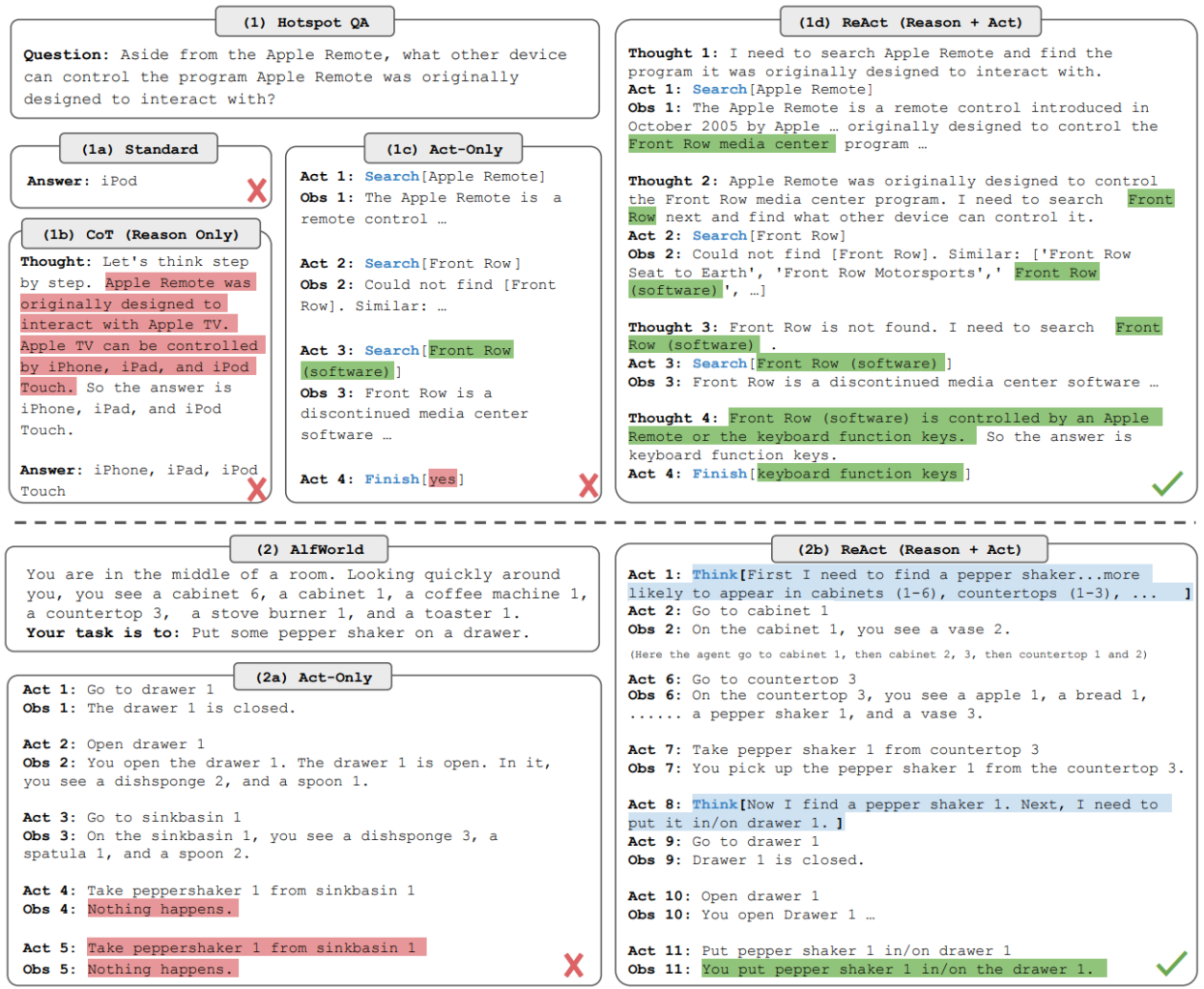
where r is a reasoning step, a is an action (API/tool call), and h is the interaction history.

ReAct Loop Example

Reason `"To answer, I need the current weather in Paris."`

Act `weather_api("Paris") → {"temp": "12°C", "condition": "cloudy"}`

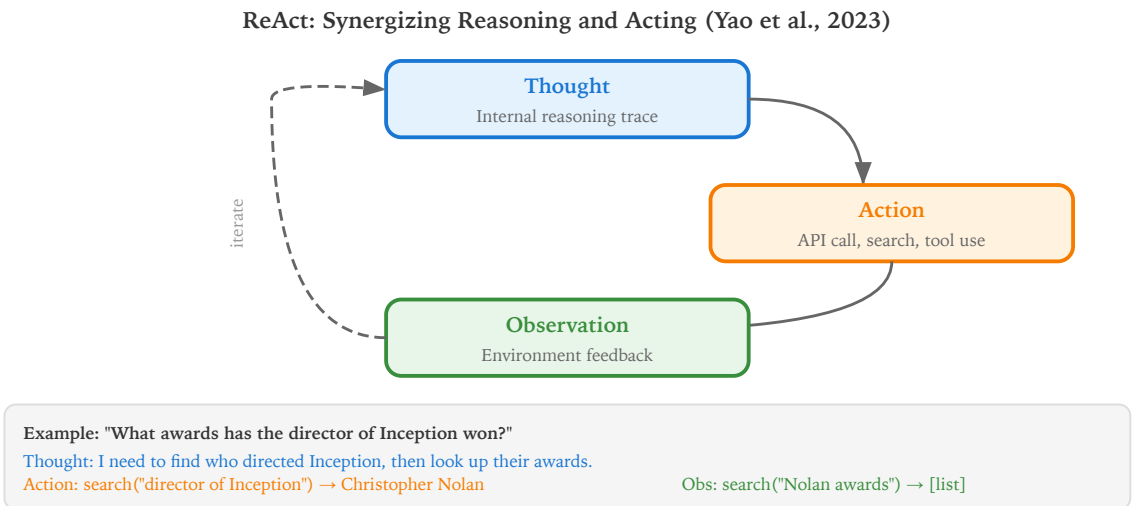
Answer `"It is currently 12°C and cloudy in Paris."`



ReAct framework comparison from Yao et al. (2023): reasoning-only vs. action-only vs. ReAct

- Tool calls are generated as output tokens, extending LLM capacity beyond language modeling

ReAct grounds each reasoning step in observable actions and environmental feedback



- ReAct agents outperform pure reasoning (CoT) and pure acting (tool-only) approaches by combining both
- The observation step grounds the model in real data, reducing hallucination

Agents combine planning, tool use, and reflection — depth in Feb 24

- **Planning:** Decompose goals into sub-tasks with explicit success criteria
- **Tool use:** Invoke APIs, run code, query databases — covered in depth Feb 24
- **Observe-act loop:** Agents iterate on environment feedback (ReAct pattern)

For our triage assistant, a simple agent loop might:

1. classify urgency
2. look up relevant medical guidelines via RAG
3. verify the classification is consistent with guidelines

Agentic pipelines face compounding errors, hallucination, and prompt injection attacks

For a pipeline of n steps where each step has error probability p_i:

$$P_{\text{failure}} = 1 - \prod_{i=1}^{n}(1 - p_i)$$

Compounding Errors

Early mistakes cascade through all downstream steps

Hallucination

Plausible but incorrect facts or invalid reasoning steps

Prompt Injection

Adversarial inputs override system instructions and safety guardrails

- Reliability decreases exponentially with pipeline depth, even when individual step error rates are low

Concept Check

If each step in a 5-step agentic pipeline has a 10% error rate, what is the overall failure probability? What does this imply about designing agentic systems?

Summary: Prompting and Reasoning – Key Takeaways

1. **In-context learning** enables task adaptation through prompts alone, with no parameter updates — the prompt serves as an inductive bias over frozen weights
2. **Prompt design matters**: wording, formatting, and example order significantly affect model behavior; principled prompt engineering is essential
3. **Chain-of-thought prompting** elicits step-by-step reasoning, yielding large accuracy gains on arithmetic, logic, and multi-hop tasks
4. **Advanced strategies** (self-consistency, tree-of-thought, least-to-most) trade additional compute for more reliable reasoning
5. **Structured prompting** constrains outputs to machine-readable formats, enabling integration with production systems
6. **Agentic workflows** extend LLMs from text generators to planners and actors, but face compounding errors, hallucination, and prompt injection risks

3 prompting heuristics you can apply tomorrow:

1. **Be explicit:** Specify role, format, constraints, and edge case handling in every prompt
2. **Test variants:** Run 3+ prompt phrasings and measure variance before committing
3. **Match compute to stakes:** Use self-consistency for high-stakes decisions, basic CoT for everyday tasks

Coming Up Next

Finetuning & Adaptation (Feb 19)

When prompting hits its limits — insufficient domain knowledge, inconsistent behavior, or the need for persistent style changes — we turn to training-time adaptation.

- Full finetuning vs. parameter-efficient methods (LoRA, adapters)
- Instruction tuning and RLHF/DPO alignment
- Choosing between prompting, finetuning, and RAG