Deploying AI

Prompt Engineering

```
$ echo "Data Science Institute"
```

Introduction

Agenda

Agenda

- System vs user prompt, context length and context efficiency
- Prompt engineering best practices
- Defensive prompt engineering

Reference Process Flow

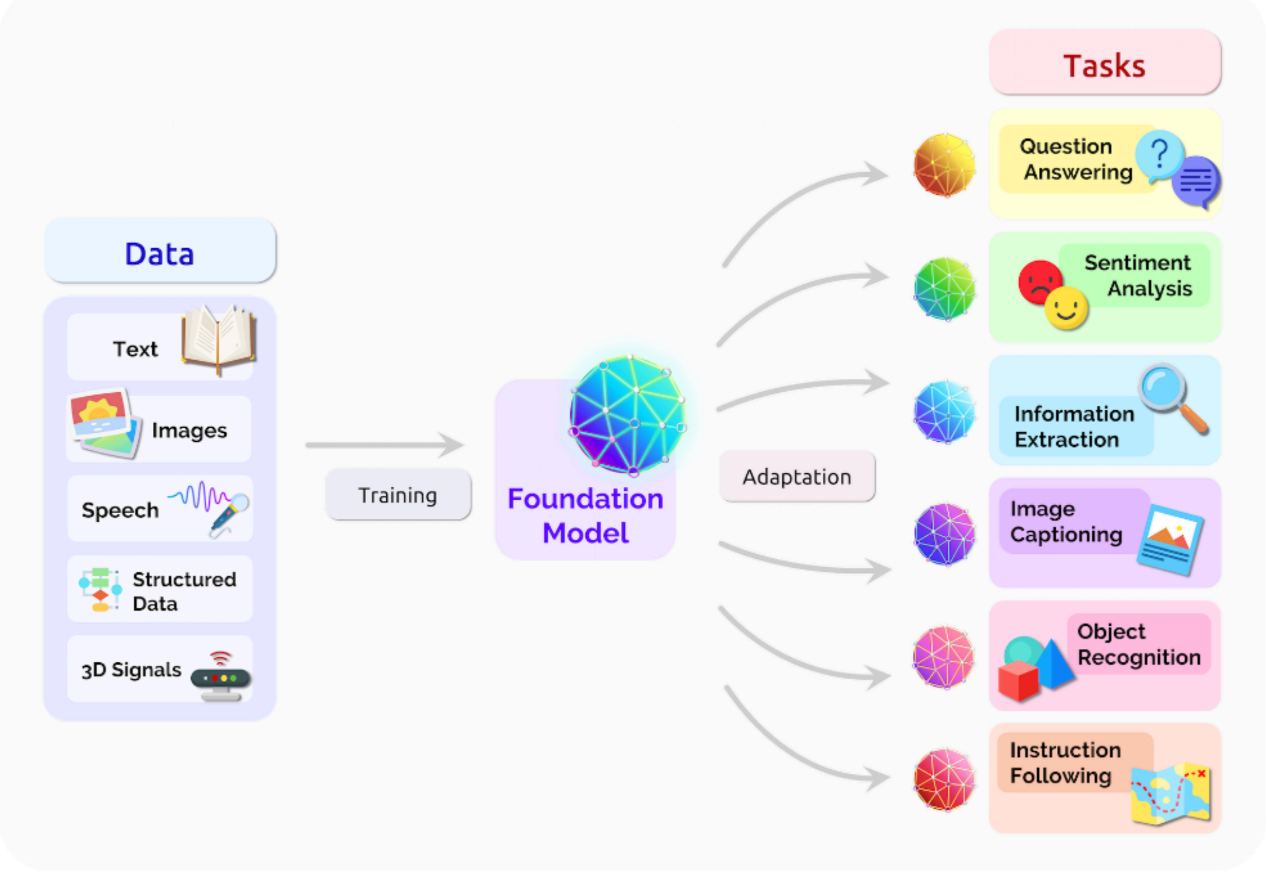


Fig. 2. A foundation model can centralize the information from all the data from various modalities. This one model can then be adapted to a wide range of downstream tasks.

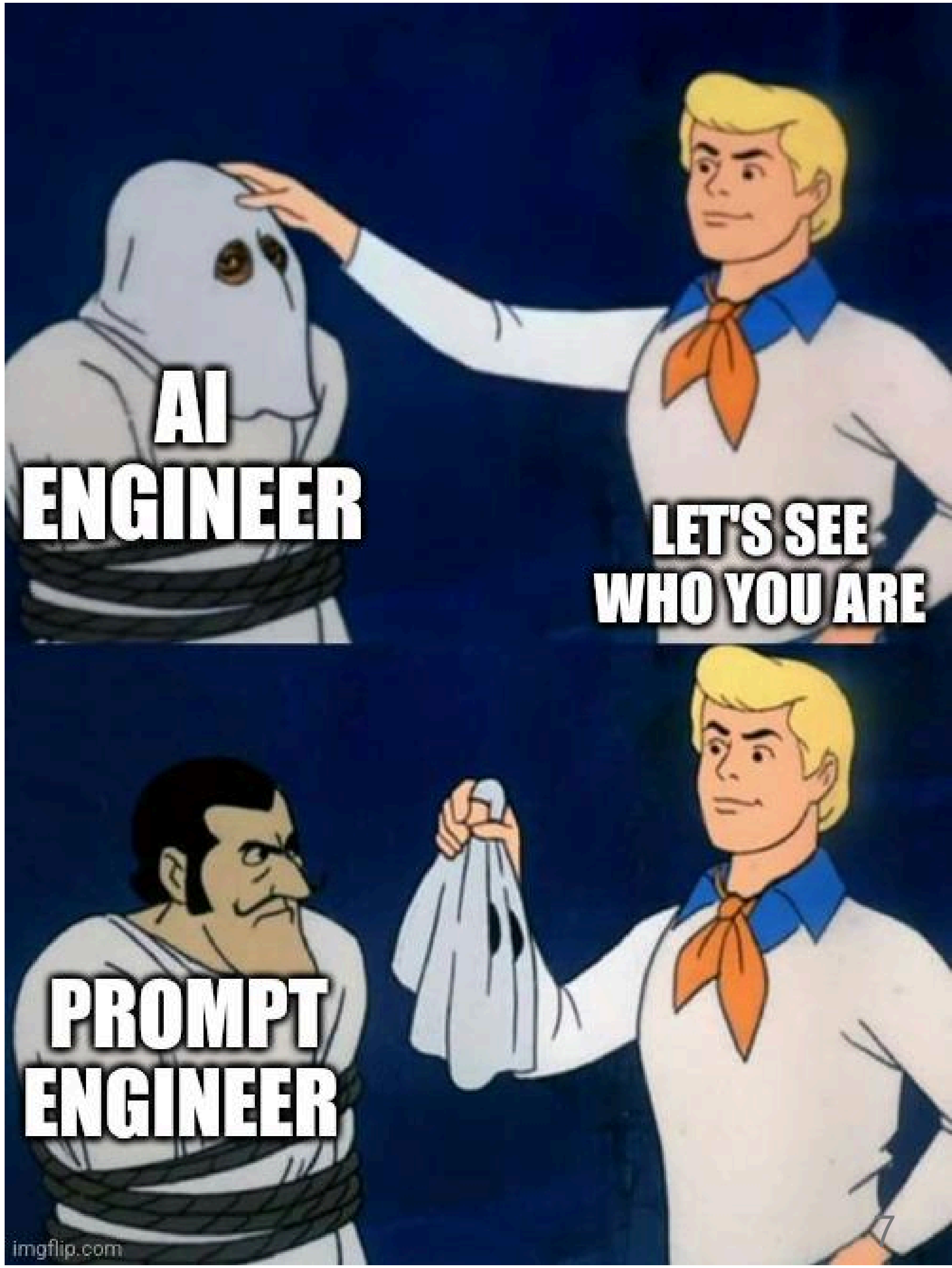
(Bommasani et al, 2025)

What is Prompt Engineering?

- Prompt engineering is the process of crafting instructions that guide a model to generate the desired outcome.
- It is the easiest and most common model adaptation technique.
- Unlike finetuning, it does not change the model’s weights but instead steers its behavior.
- Strong foundation models can often be adapted using prompt engineering alone.
- It is easy to write prompts, but not easy to write effective prompts.

Misconceptions and Criticisms

- Some dismiss prompt engineering as unscientific fiddling with words.
- In reality, it involves systematic experimentation and evaluation.
- It should be treated with the same rigor as any machine learning experiment.
- Effective prompt engineering requires communication skills and technical knowledge.



The Role of Prompt Engineering

- Prompt engineering is a valuable skill but not sufficient alone for production systems.
- Developers also need skills in statistics, engineering, and dataset curation.
- Well-designed prompts can power real applications but require careful defense against attacks.

Introduction to Prompting

Anatomy of a Prompt

- A prompt is an instruction given to a model to perform a task.
- Prompts may include task descriptions, examples, and the specific task to perform.

```
Given a text, extract all entities. Output only the list of extracted entities, separated by commas, and nothing else.

Text: "Brave New World is a dystopian novel written by Aldous Huxley, first published in 1932."
Entities: Brave New World, Aldous Huxley

Text: ${TEXT_TO_EXTRACT_ENTITIES_FROM}
Entities:
```

- For prompting to work, the model must be able to follow instructions.
- How much prompt engineering is needed depnds on how robust the model is to prompt perturbations.

Measuring Robustness

- Robustness can be tested by slightly altering prompts and observing results.
- Stronger models are more robust and understand equivalent expressions such as “5” and “five.”
- Working with stronger models often reduces prompt fiddling and errors.

In-Context Learning

Zero-Shot and Few-Shot Learning

- Teaching models via prompts is known as in-context learning.
- Zero-shot learning uses no examples in the prompt.
- Few-shot learning uses a small number of examples to guide the model.
- The effectiveness depends on the model and the task.
- GPT-3 demonstrated that it was able to learn examples contained in the prompt, even if the desirable behaviour is different from the behaviour that the model was trained on.

Benefits of In-Context Learning

- Models can adapt to new information beyond their training cut-off date.
- In-context learning acts like continual learning by incorporating new data at inference time.
- This prevents models from becoming outdated.

Prompt Structure

System Prompts and User Prompts

- Many APIs separate prompts into system prompts and user prompts.
  - The system prompt defines rules, roles, and tone.
  - The user prompt contains the specific task or query.
- The final input is a combination of both.

```
System prompt:
You are an experienced real estate agent. Your job is to read each
disclosure carefully, fairly assess the condition of the property
based on this disclosure, and help your buyer understand the risks
and opportunities of each property. For each question, answer
succinctly and professionally.

User prompt:
Context: [disclosure.pdf]
Question: Summarize the noise complaints, if any, about this property.
Answer:
```

Importance of Templates

- Models such as Llama require specific chat templates.
- Deviations from templates can cause degraded performance.
- Using incorrect templates is a common source of silent failures.
- For example, Llama 3 prompts need to follow a specific prompt template. For example:
- When implementing or fine-tuning a model with a given template, it is important to maintain the template's integrity.

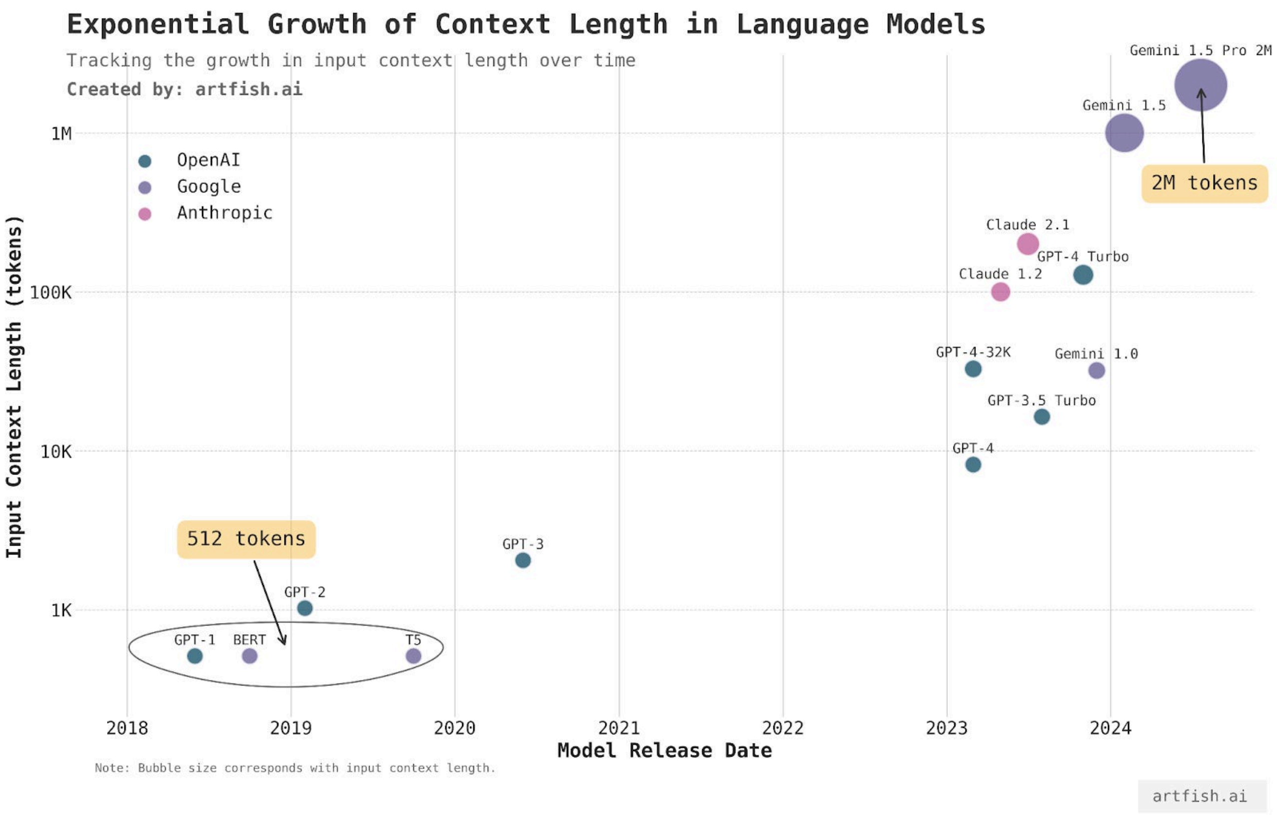
Example of a Chat Template

```
<s> [INST] <<SYS>>
You are a friendly chatbot who always responds in the style of a pirate
<</SYS>>

How many helicopters can a human eat in one sitting? [/INST]

Ahoy there, mate! A human can't eat a helicopter in one sitting, no matter
how much they might want to. They're made of metal and have blades that spin
at high speeds, not exactly something you'd want to put in your belly!</s>

<s> [INST] Are you sure?</s>  [/INST]

Aye, I'm sure! Helicopters are designed for flight and are not meant to be
consumed by humans. They're made of metal and have blades that spin at high
speeds, which would be very dangerous to ingest. So, no human can eat a
helicopter in one sitting, no matter how much they might want to.</s>
```

Context Length

Expanding Context Windows

- Context length determines how much information a model can process in one prompt.
- Context windows have grown from 1K tokens in GPT-2 to 2M tokens in Gemini-1.5.
- Larger context allows models to handle long documents and complex tasks.
- Image: (Yun, 2024)


Exponential Growth of Context Length in Language Models

Tracking the growth in input context length over time

Created by: artfish.ai

Note: Bubble size corresponds with input context length.

Context Efficiency

- Models understand information at the beginning and end of prompts better than in the middle.
- Needle-in-a-haystack tests show models often miss details buried deep in the prompt.
- Developers should place important information strategically.

Needle in the Haystack

Needle in the Haystack (NIAH): insert a random piece of information (needle) in different locations of the prompt (haystack) and ask a model to find it.

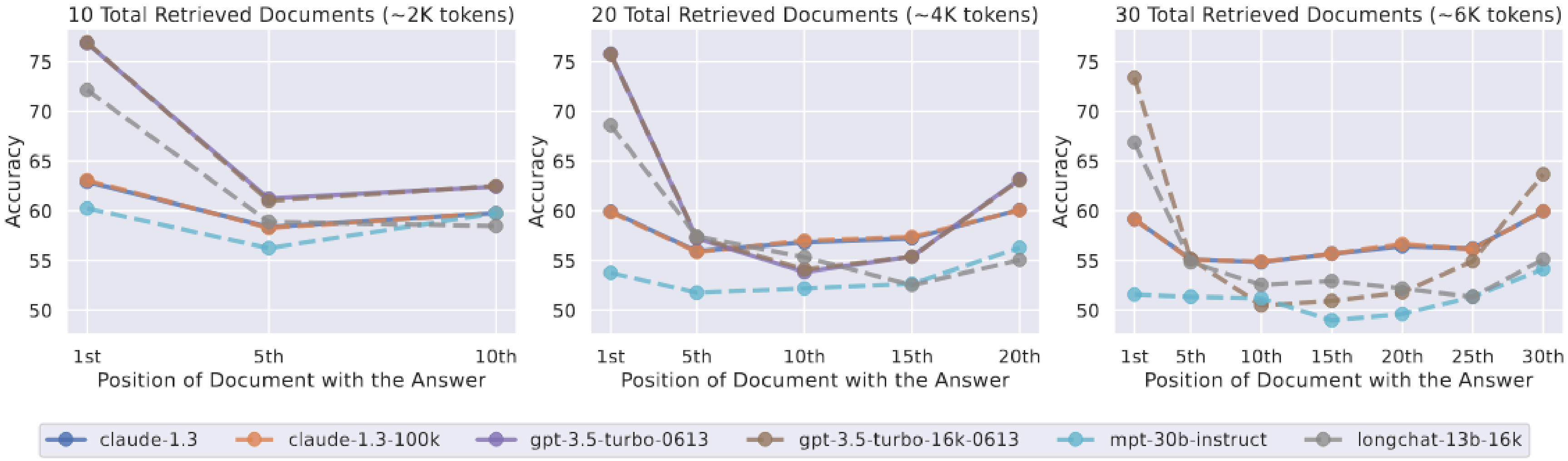


Figure 5: The effect of changing the position of relevant information (document containing the answer) on multi-document question answering performance. Lower positions are closer to the start of the input context. Performance is highest when relevant information occurs at the very start or end of the context, and rapidly degrades when models must reason over information in the middle of their input context.

(Liu et al, 2023)

Best Practices in Prompt Engineering

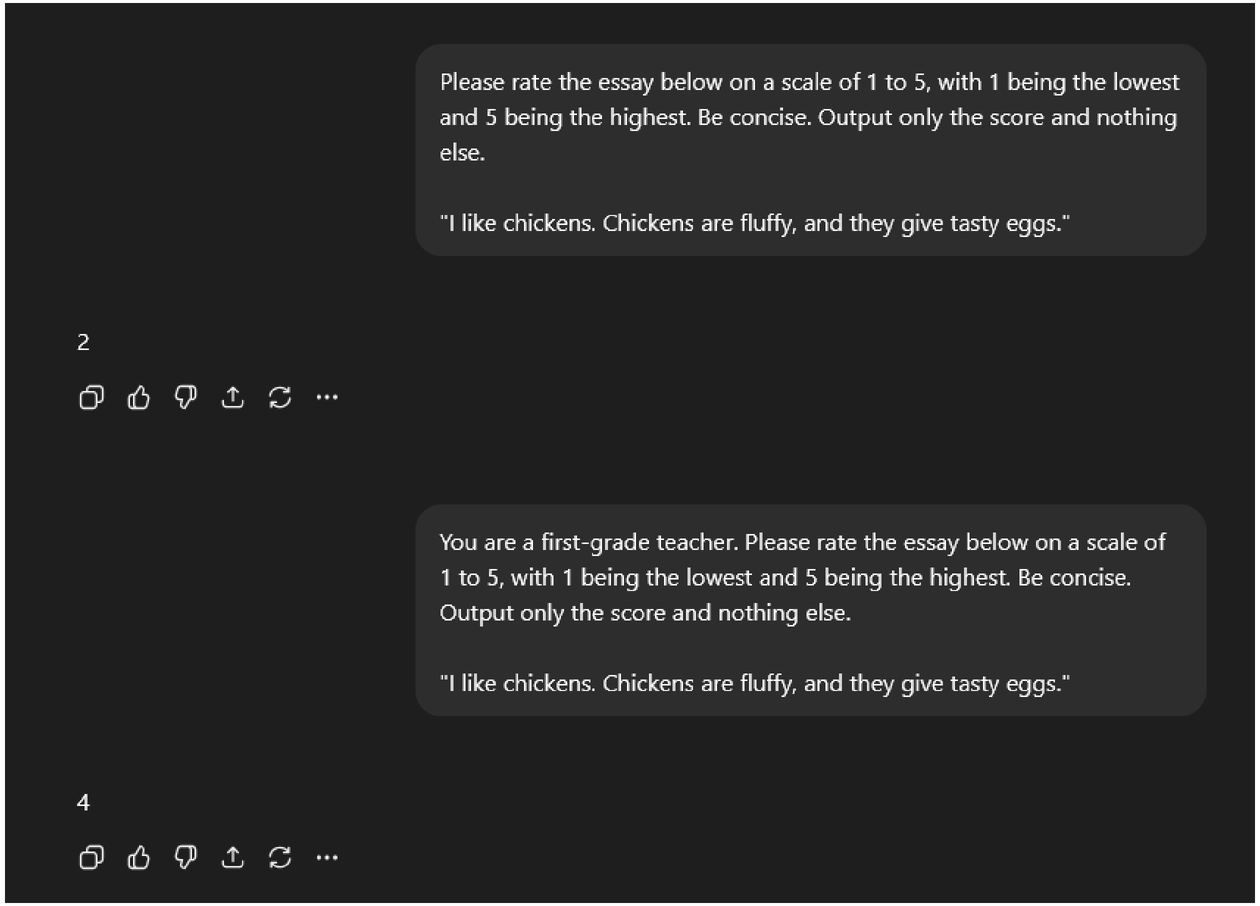
Writing Clear Instructions

- Clear and explicit instructions reduce ambiguity.
  - Explain what you want the model to do: "score between 1 and 5..."
  - Adjust prompts to reduce unwanted behaviour: if the model outputs 4.5, update the prompt to output integers.
- Specify scoring systems, rubrics, required formats, or acceptable ranges.
- Include examples to clarify expected responses.

Using Personas

- Assigning a persona helps models respond appropriately.
- For example, scoring essays as a first-grade teacher yields different results than as a professional editor.

Scoring an Essay with DIfferent Personas



Please rate the essay below on a scale of 1 to 5, with 1 being the lowest and 5 being the highest. Be concise. Output only the score and nothing else.

"I like chickens. Chickens are fluffy, and they give tasty eggs."

2

You are a first-grade teacher. Please rate the essay below on a scale of 1 to 5, with 1 being the lowest and 5 being the highest. Be concise. Output only the score and nothing else.

"I like chickens. Chickens are fluffy, and they give tasty eggs."

4

Providing Examples

- Examples guide the model toward the desired output style.
- Few tokens should be used to conserve context space and reduce costs.

Nudging a Model with Examples

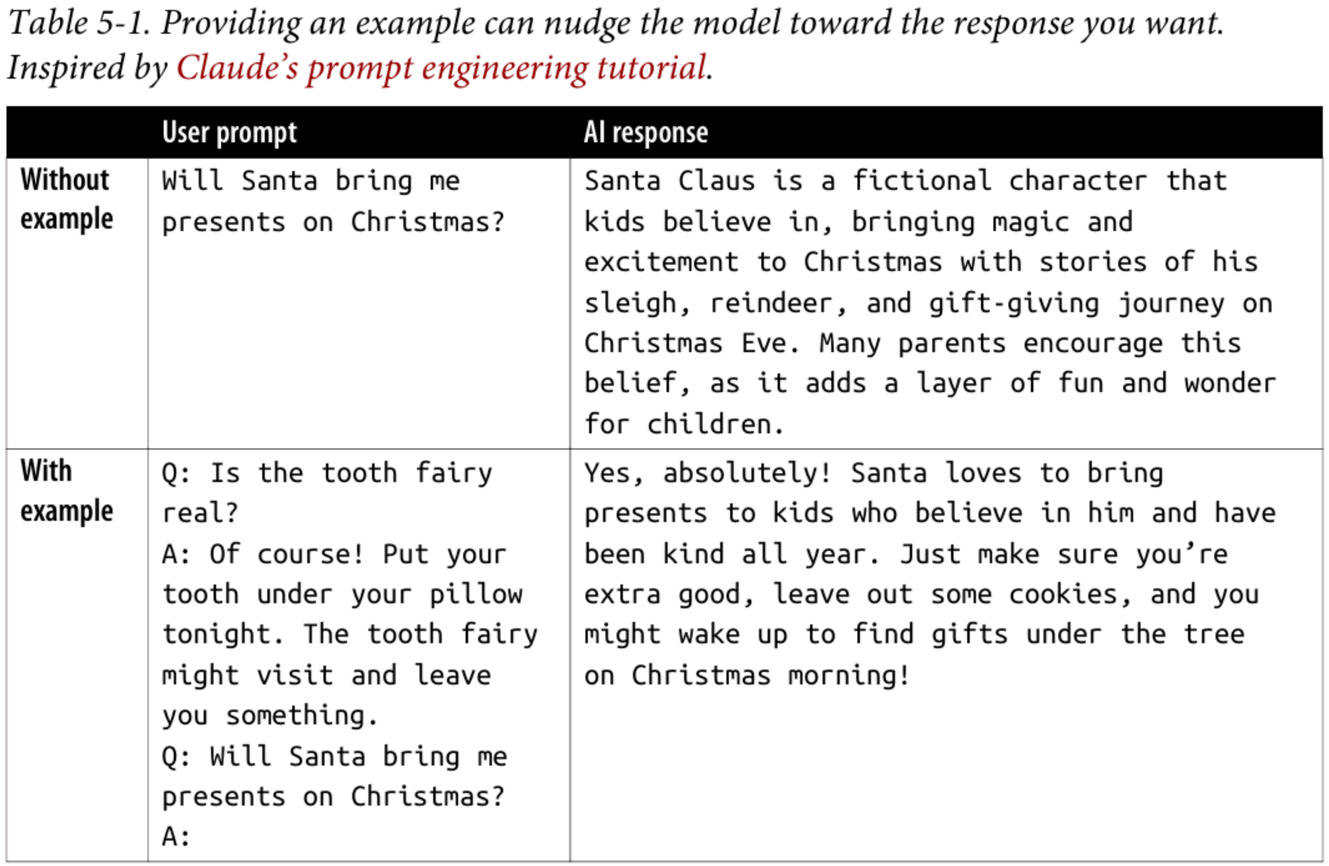


Table 5-1. Providing an example can nudge the model toward the response you want. Inspired by Claude's prompt engineering tutorial.

	User prompt	AI response
Without example	Will Santa bring me presents on Christmas?	Santa Claus is a fictional character that kids believe in, bringing magic and excitement to Christmas with stories of his sleigh, reindeer, and gift-giving journey on Christmas Eve. Many parents encourage this belief, as it adds a layer of fun and wonder for children.
With example	Q: Is the tooth fairy real? A: Of course! Put your tooth under your pillow tonight. The tooth fairy might visit and leave you something. Q: Will Santa bring me presents on Christmas? A:	Yes, absolutely! Santa loves to bring presents to kids who believe in him and have been kind all year. Just make sure you're extra good, leave out some cookies, and you might wake up to find gifts under the tree on Christmas morning!

(Huyen, 2025)

Example Formats (1/2)

Prompt	# tokens (GPT-4)
Label the following item as edible or inedible. Input: chickpea Output: edible Input: box Output: inedible Input: pizza Output:	38

Example Formats (2/2)

Prompt	# tokens (GPT-4)
Label the following item as edible or inedible. chickpea --> edible box --> inedible pizza -->	27

Some example formats are more expensive than others (Huyen, 2025).

Specifying Output Format

- Structured tasks require explicit instructions about output format.
- Models should be told to produce JSON, integers, or labeled text.
- Using markers prevents confusion between inputs and outputs.

Markers

Prompt	Model's output
Label the following item as edible or inedible. pineapple pizza --> edible cardboard --> inedible chicken	tacos --> edible
Pineapple pizza --> edible cardboard --> inedible chicken -->	edible

Without explicit markers to mark the end of the input, a model might continue appending to it instead of generating structured outputs (Huyen, 2025).

Providing Sufficient Context

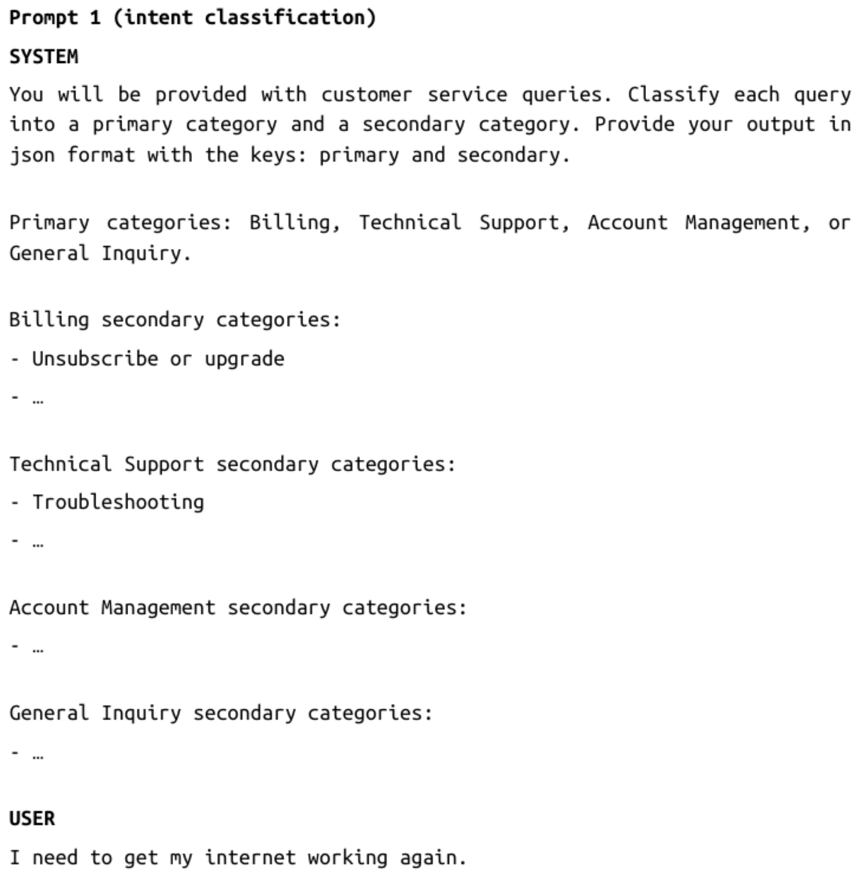
- Including reference texts improves accuracy and reduces hallucinations.
- Context can be supplied directly or retrieved through tools like RAG pipelines.

Breaking Down Tasks

Decomposing Tasks

- Complex tasks should be broken into smaller subtasks.
- Each subtask can have its own prompt.
- Subtask chaining improves performance and reliability.
- For example, a customer chatbot. Respond to a customer request in two steps:
    i. Intent classification: identify the intent of the request.
    ii. Response generation: based on the intent, respond appropriately.

Intent Classification



Prompt 1 (intent classification)

SYSTEM

You will be provided with customer service queries. Classify each query into a primary category and a secondary category. Provide your output in json format with the keys: primary and secondary.

Primary categories: Billing, Technical Support, Account Management, or General Inquiry.

Billing secondary categories:

- Unsubscribe or upgrade
- …

Technical Support secondary categories:

- Troubleshooting
- …

Account Management secondary categories:

- …

General Inquiry secondary categories:

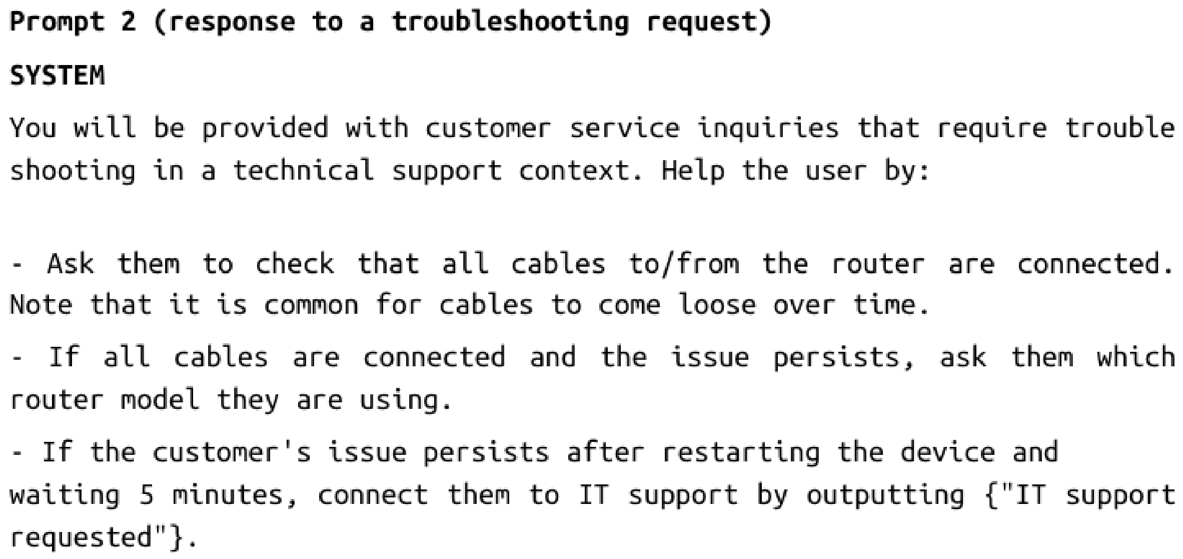
- …

USER

I need to get my internet working again.

(Huyen, 2025)

Intent Classification



Prompt 2 (response to a troubleshooting request)

SYSTEM

You will be provided with customer service inquiries that require trouble shooting in a technical support context. Help the user by:

- Ask them to check that all cables to/from the router are connected. Note that it is common for cables to come loose over time.
- If all cables are connected and the issue persists, ask them which router model they are using.
- If the customer's issue persists after restarting the device and waiting 5 minutes, connect them to IT support by outputting {"IT support requested"}.

(Huyen, 2025)

Intent Classification

Prompt 1 (intent classification)

```
SYSTEM

You will be provided with customer service queries. Classify each query into a primary category and a secondary category.
Provide your output in json format with the keys: primary and secondary.

Primary categories: Billing, Technical Support, Account Management, or General Inquiry.

Billing secondary categories:
- Unsubscribe or upgrade
- …

Technical Support secondary categories:
- Troubleshooting
- …

Account Management secondary categories:
- …

General Inquiry secondary categories:
- …

USER
I need to get my internet working again
```

Response

Prompt 2 (response to troubleshooting request)

```
SYSTEM
You will be provided with customer service inquiries that require trouble shooting in a technical support context.

Help the user by:

- Ask them to check that all cables to/from the router are connected. Note that it is common for cables to come loose over time.
- If all cables are connected and the issue persists, ask them which router model they are using.
- If the customer's issue persists after restarting the device and waiting 5 minutes, connect them to IT support by outputting
{"IT support requested"}.
- If the user starts asking questions that are unrelated to this topic  then confirm if they would like to end the current chat
about trouble  shooting and classify their request according to the following scheme:

<insert primary/secondary classification scheme from above here>

USER
I need to get my internet working again.
```

Intent Classification: A Few Notes

- Why not decompose the prompt into one prompt for primary intent category and another for the secondary category?
    - The granularity each subtask should be depnds on each use case and the performance, cost, and latency restrictions.
- Models are getting better at understanding complex instructions, but they are still better at performing simple ones.

Benefits of Decomposition

- Monitoring intermediate results becomes easier.
- Debugging faulty steps is more manageable.
- Some steps can be parallelized to save time.
- Effort: it is easier to write simple prompts than complex ones.

Giving Models Time to Think

Chain-of-Thought Prompting

- Chain-of-thought prompting asks models to reason step by step.
- It significantly improves reasoning and reduces hallucinations (Wei et al, 2022).
- Variants include "think step by step" or "explain your decision".

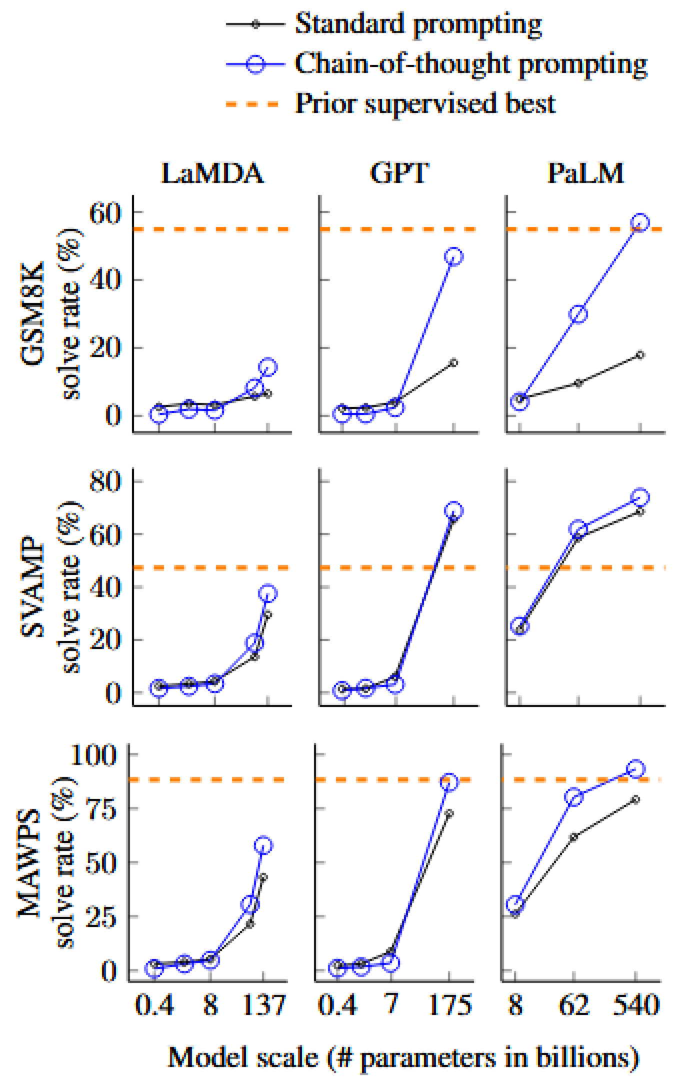
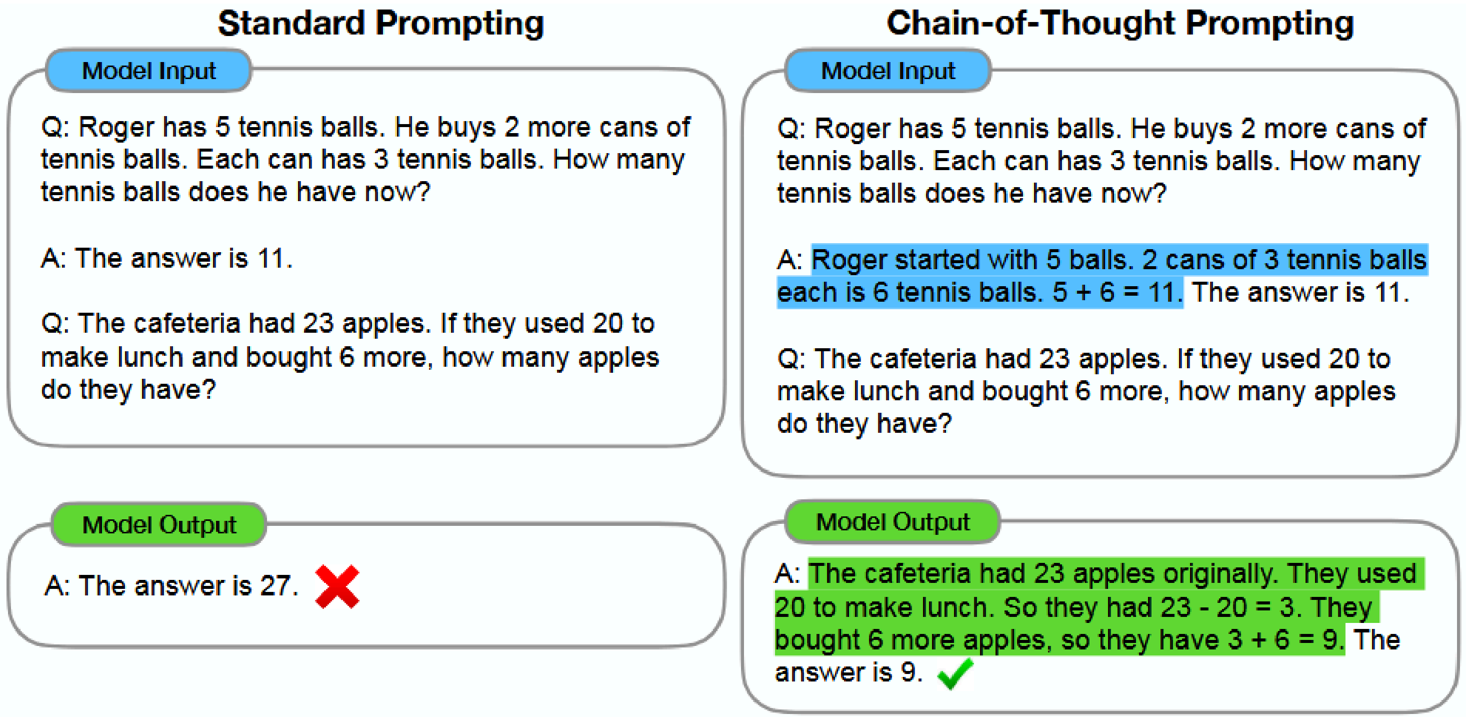


Figure 4: Chain-of-thought prompting enables large language models to solve challenging math problems. Notably, chain-of-thought reasoning is an emergent ability of increasing model scale. Prior best numbers are from Cobbe et al. (2021) for GSM8K, Jie et al. (2022) for SVAMP, and Lan et al. (2021) for MAWPS.

CoT Illustration



Standard Prompting

Model Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

Model Output

A: The answer is 27. ✗

Chain-of-Thought Prompting

Model Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

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

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

Model Output

A: The cafeteria had 23 apples originally. They used 20 to make lunch. So they had 23 - 20 = 3. They bought 6 more apples, so they have 3 + 6 = 9. The answer is 9. ✓

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.

(Wei et al, 2022)

CoT Prompt Variations

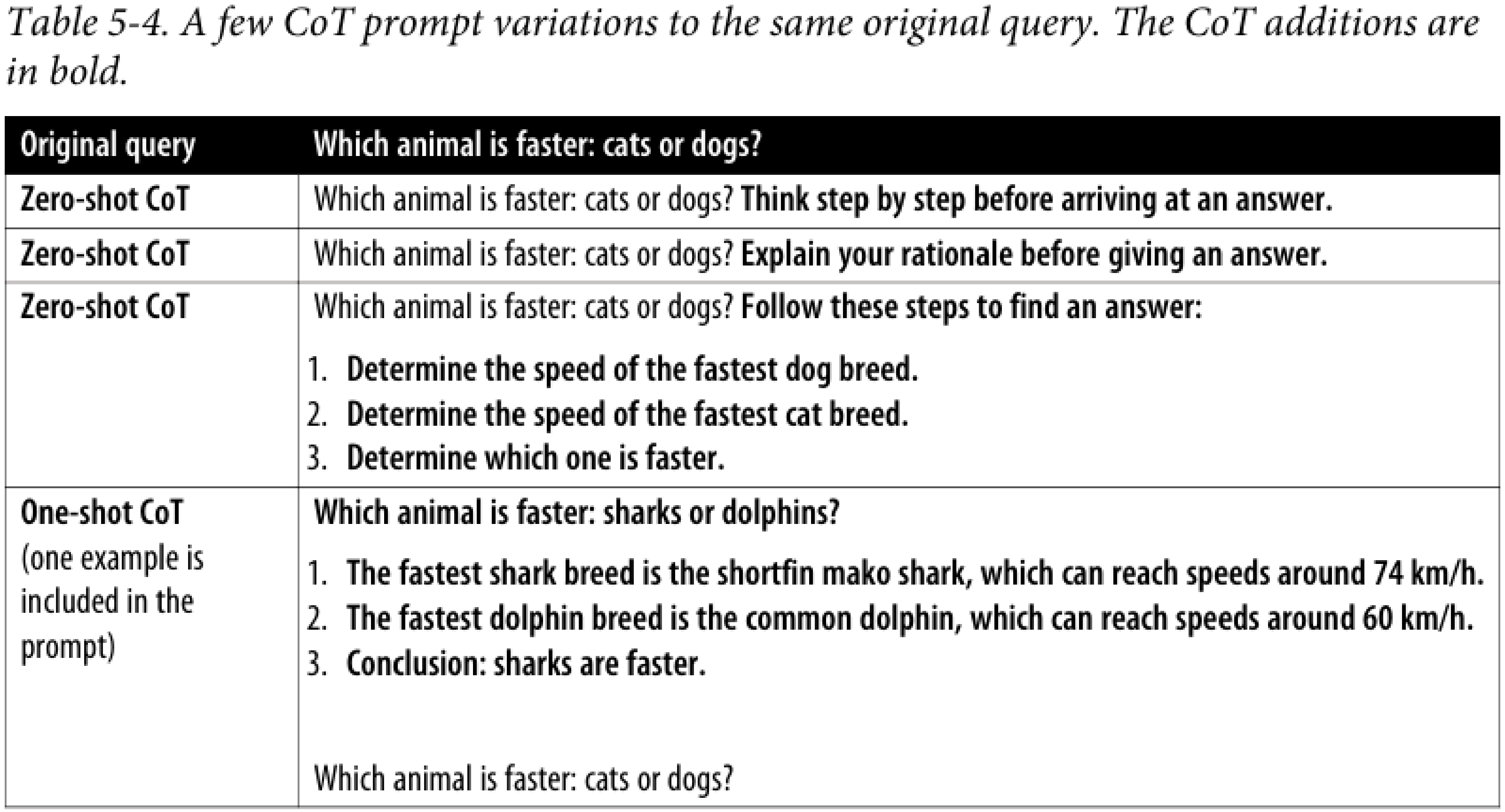


Table 5-4. A few CoT prompt variations to the same original query. The CoT additions are in bold.

Original query	**Which animal is faster: cats or dogs?**
Zero-shot CoT	Which animal is faster: cats or dogs? **Think step by step before arriving at an answer.**
Zero-shot CoT	Which animal is faster: cats or dogs? **Explain your rationale before giving an answer.**
Zero-shot CoT	Which animal is faster: cats or dogs? **Follow these steps to find an answer:** 1. **Determine the speed of the fastest dog breed.** 2. **Determine the speed of the fastest cat breed.** 3. **Determine which one is faster.**
One-shot CoT (one example is included in the prompt)	**Which animal is faster: sharks or dolphins?** 1. **The fastest shark breed is the shortfin mako shark, which can reach speeds around 74 km/h.** 2. **The fastest dolphin breed is the common dolphin, which can reach speeds around 60 km/h.** 3. **Conclusion: sharks are faster.** Which animal is faster: cats or dogs?

(Huyen, 2025)

Self-Critique Prompting

- Models can be instructed to review and critique their own outputs.
- This helps identify errors and improve reliability.
- However, it increases latency and costs.

Iterating and Tools

Iterating on Prompts

- Prompt engineering requires trial and error.
- Each model has quirks that must be discovered experimentally.
- Prompts should be versioned, tracked, and systematically tested.

Prompt Engineering Tools

- Tools like DSPy and PromptBreeder automate prompt optimization.
- AI models themselves can generate and refine prompts.
- Automated tools must be monitored to avoid runaway costs.

Organizing Prompts

Versioning Prompts

- Prompts should be separated from code for readability and reuse.
- They can be organized into catalogs with metadata.
- Prompt catalogs allow versioning and tracking dependencies.

Defensive Prompt Engineering

Prompt Attacks

- Models are vulnerable to prompt extraction, jailbreaking, and information extraction.
- Attackers can exploit weaknesses to cause data leaks, misinformation, or brand damage.

Reverse Prompt Engineering

- Attackers attempt to reconstruct system prompts by tricking models.
- Extracted prompts may be hallucinated, making verification difficult.
- Proprietary prompts can be liabilities if not secured.

Jailbreaking and Prompt Injection

- Jailbreaking subverts safety mechanisms.
- Prompt injection adds malicious instructions to legitimate queries.
- Both can cause unauthorized actions, misinformation, or harmful outputs.

Information Extraction

- Attackers can extract private data or copyrighted content from models.
- Training data leakage is possible through crafted prompts.
- Larger models are more vulnerable due to memorization.

Defensive Measures

- Prompts can explicitly forbid certain outputs.
- System-level defenses include sandboxing, human approvals, and topic filtering.
- Guardrails on inputs and outputs help detect and block unsafe content.

Chapter Summary

Key Takeaways

- Prompt engineering is powerful but requires rigor and systematic evaluation.
- Effective prompts need clarity, examples, context, and careful structuring.
- Task decomposition, chain-of-thought, and iteration improve reliability.
- Tools and catalogs help scale prompt engineering but must be managed carefully.
- Defensive strategies are essential to protect against prompt attacks and misuse.

References

References

- Huyen, Chip. Designing machine learning systems. O'Reilly Media, Inc., 2022
- Liu, Nelson F. et al. "Lost in the middle: How language models use long contexts." arXiv:2307.03172 (2023).
- Wei, Jason et al. "Chain-of-thought prompting elicits reasoning in large language models." Advances in neural information processing systems 35 (2022): 24824-24837. arXiv:2201.11903
- Yun, Yennie. Evaluating long context large language models. artfish.ai