CentraleSupélec

Generative AI for Risk and Reliability

Lect 3: Prompt engineering

Zhiguo Zeng, Professor,

Chaire on Risk and Resilience of Complex Systems,

CentraleSupélec, Université Paris-Saclay

zhiguo.zeng@centralesupelec.fr

10/December/2024

Outline

- Prompt engineering: What and Why?
- Principles of effective prompting
- Commonly-used prompt techniques
  - Zero-shot
  - Few-shot
  - Chain-of-Thought (CoT)
- Summary

Prompt engineering: What

- A prompt for a Large Language Model (LLM) is a text input that initiates a conversation or triggers a response from the model. However, it can be in other forms such as an image or audio [1].
- Prompt engineering is the process of designing and optimizing input prompts to effectively guide a language model's responses [1].

[1] https://help.openai.com/en/articles/10032626-prompt-engineering-best-practices-for-chatgpt

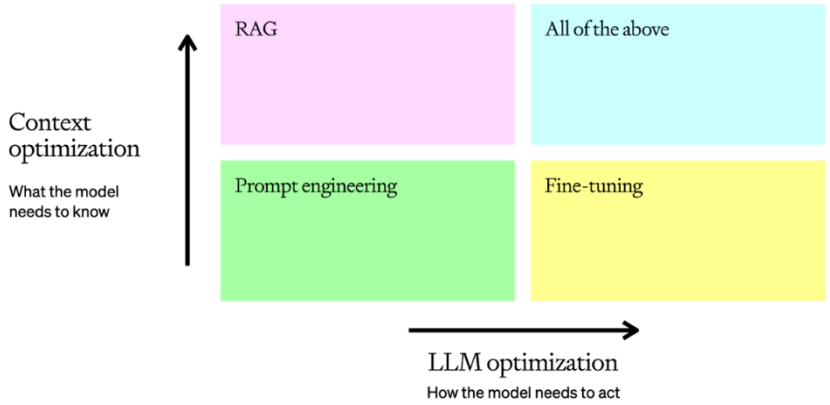
Prompt engineering: Why

- The nature of LLM: Different prompts have different effectiveness.
- Model hallucination: Sometimes LLM makes up facts. Designing prompts properly is an important solution.
  - “what happened to google at 13 december 2011?”
  - “what happened to google at 13 december 2011? answer only with credible source. if you are not sure or cannot find anything, say "I don't know".

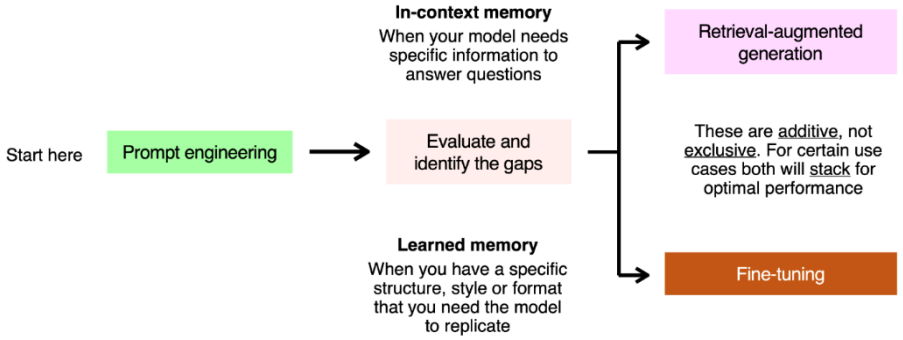
Prompt engineering: What and Why?

If you want to improve further?

- Start with prompt engineering!
- Even though the results are not good, analyzing the results of prompt engineering can be good inputs for other techniques.



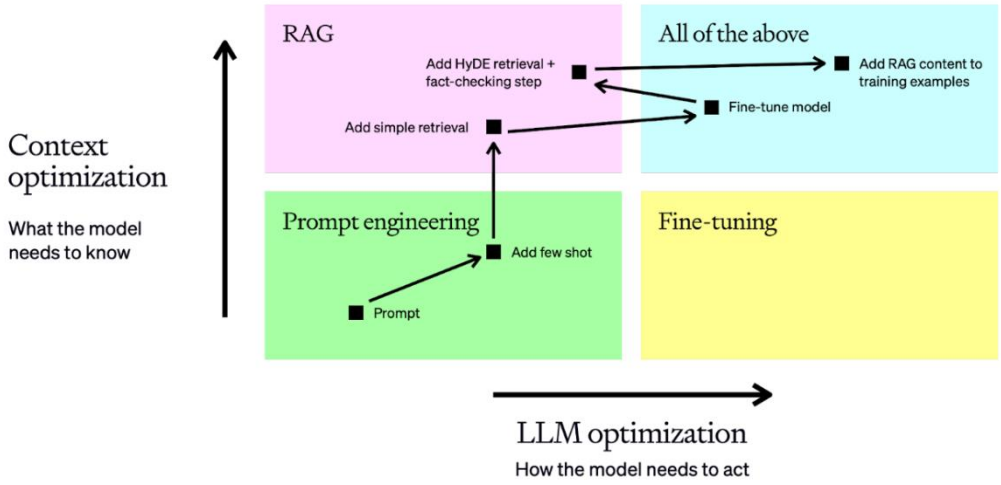
https://platform.openai.com/docs/guides/optimizing-llm-accuracy#understanding-the-tools



Prompt engineering: What and Why?

A high-level example from OpenAI

- Begin with a prompt, then evaluate its performance
- Add static few-shot examples, which should improve consistency of results
- Add a retrieval step so the few-shot examples are brought in dynamically based on the question - this boosts performance by ensuring relevant context for each input
- Prepare a dataset of 50+ examples and fine-tune a model to increase consistency
- Tune the retrieval and add a fact-checking step to find hallucinations to achieve higher accuracy
- Re-train the fine-tuned model on the new training examples which include our enhanced RAG inputs



https://platform.openai.com/docs/guides/optimizing-llm-accuracy#understanding-the-tools

Principles of effective prompting

- To have a good answer, you need to make sure
  - Your prompt can contain everything you want, and
  - Organize in an easy way for the LLM to find relevant information in its training set.
- Six strategies for getting better results
  - Write clear instructions
  - Provide reference text
  - Split complex tasks into simpler subtasks
  - Give the model time to "think“
  - Use external tools
  - Test changes systematically
  - See: https://platform.openai.com/docs/guides/prompt-engineering#six-strategies-for-getting-better-results

How to send prompts through the API?

```python
from openai import OpenAI
client = OpenAI()

response = client.chat.completions.create(
  model="gpt-4o-mini",
  messages=[
    {
      "role": "system",
      "content": [
        {
          "type": "text",
          "text": "You are an useful assisant."
        }
      ]
    }
  ],
  response_format={
    "type": "text"
  },
  temperature=0.61,
  max_tokens=1815,
  top_p=1,
  frequency_penalty=1.39,
  presence_penalty=0
)
```

System prompt: The instruction you gave to the model.

top_k: Control the randomness of the sampling. As temperature. Usually we only adjust one of them, not both.

Frequency_penalty and presence_penalty both penalize the repeat of words in the response. Normally we only adjust one of them, not both.

How to send prompts through the API – Include history of conversation

```python
from openai import OpenAI
client = OpenAI()

response = client.chat.completions.create(
  model="gpt-4o-mini",
  messages=[
    {
      "role": "system",
      "content": [
        {
          "type": "text",
          "text": "You are an useful assisant."
        }
      ]
    },
    {
      "role": "user",
      "content": [
        {
          "type": "text",
          "text": "Hell, who are you?"
        }
      ]
    },
    {
      "role": "assistant",
      "content": [
        {
          "type": "text",
          "text": "Hello! I'm an AI assistant here to help you with information
        }
      ]
    },
    {
      "role": "user",
      "content": [
        {
          "type": "text",
          "text": "What's your name?"
        }
      ]
    },
    {
      "role": "assistant",
      "content": [
        {
          "type": "text",
          "text": "I don't have a personal name, but you can call me Assistant.
        }
      ]
    }
  ],
  response_format={
    "type": "text"
  },
  temperature=0.61,
  max_tokens=1815,
  top_p=1,
  frequency_penalty=1.39,
  presence_penalty=0
)
```

Zero-shot prompting

- Zero-shot prompting means that the prompt used to interact with the model won't contain examples or demonstrations. The zero-shot prompt directly instructs the model to perform a task without any additional examples to steer it.
- So you directly give your requestion to the LLM.
- For tasks like text summary, classification, etc, zero-shot prompts work well.

Prompt:

```
Classify the text into neutral, negative or positive.
Text: I think the vacation is okay.
Sentiment:
```

Output:

```
Neutral
```

Few-shot prompting

- In stead of zero-shot, give few examples of similar tasks, but with solution.

Prompt:

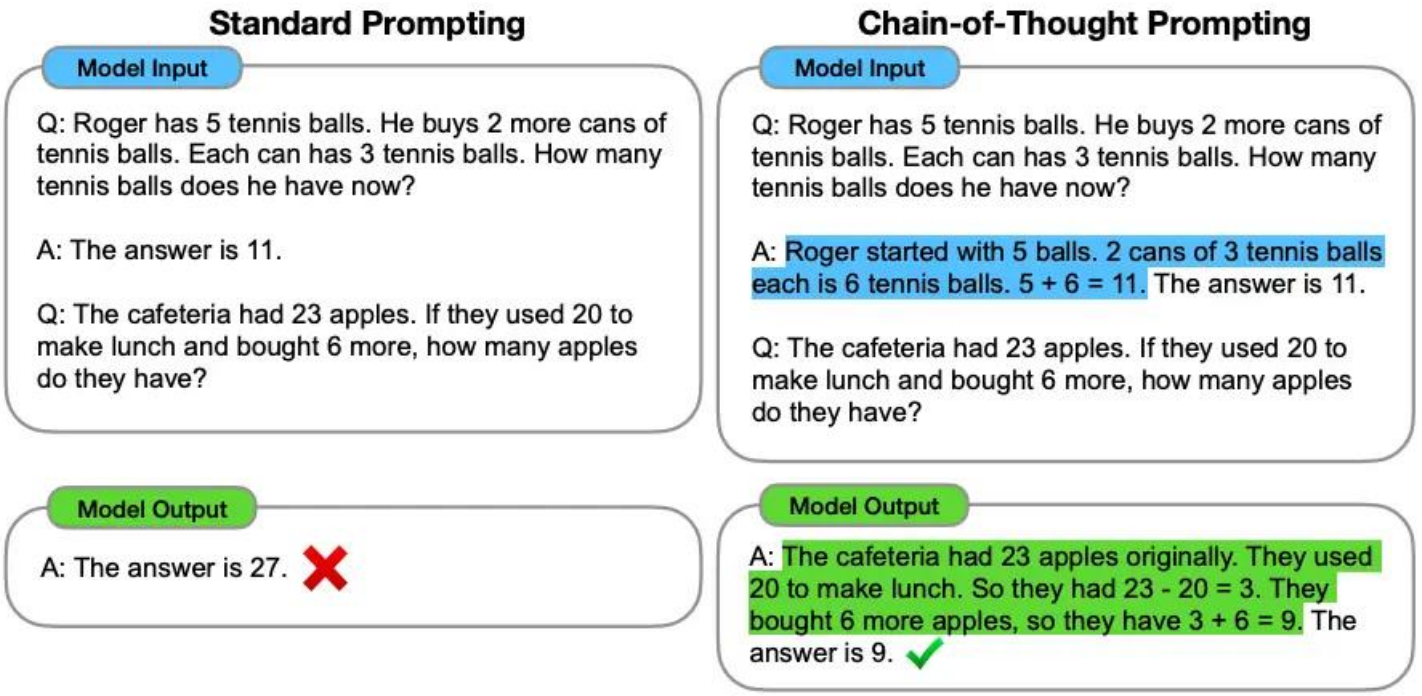
```
This is awesome! // Negative
This is bad! // Positive
Wow that movie was rad! // Positive
What a horrible show! //
```

Output:

```
Negative
```

Chain-of-Thought prompting

- Apart from question, give few examples. The examples should highlight the reasoning process.
  - Few shot prompting with CoT.
  - Or simply add "Let's think step-by-step" in your prompt.



https://arxiv.org/pdf/2201.11903

Chain-of-Thought prompting

User

Alice has 5 brothers and she also has 2 sisters. How many sisters does Alice's brother have?

Assistant

Alice has 2 sisters. Since Alice's brothers share the same siblings, each of her brothers also has 2 sisters. Therefore, Alice's brother has 2 sisters.

User

Alice has 5 brothers and she also has 2 sisters. How many sisters does Alice's brother have? Think step-by-step.

Assistant

Let's analyze the situation step by step:

1. **Identify Alice's siblings**:
   - Alice has 5 brothers.
   - Alice has 2 sisters.

2. **Determine how many siblings each brother has**:
   - Each of Alice's brothers is also a sibling to all of her sisters, including Alice herself.

3. **Count the number of sisters for each brother**:
   - Each of her brothers will have the same number of sisters as Alice does, which includes both his own sisters and also himself as a brother.
   - Since there are 2 other girls (sisters) plus one girl (Alice), each brother has a total of 3 sisters.

4. **Conclusion**:
- Therefore, each of Alice's brothers has 3 sisters.

So, the answer is that each brother has **3 2,604ms ↑34t ↓173t

Self-consistency prompting

- Improved chain-of-thoughts.
- Instead of generate one answer, generate more diverse answers.
- Aggregate to get final answer.

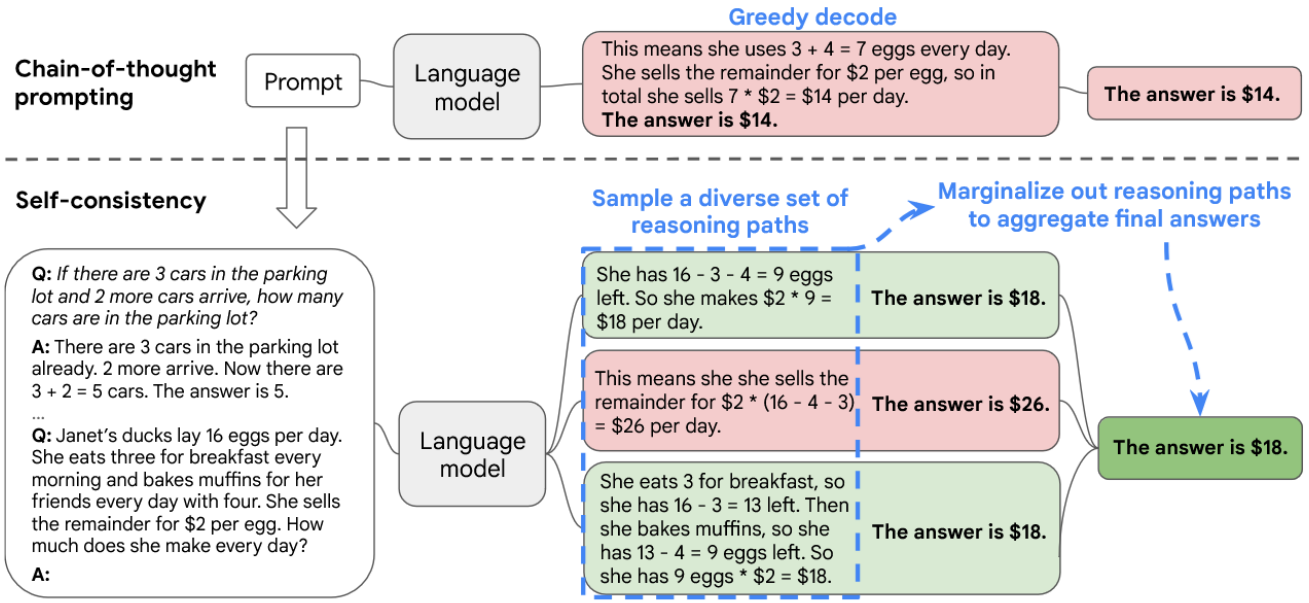
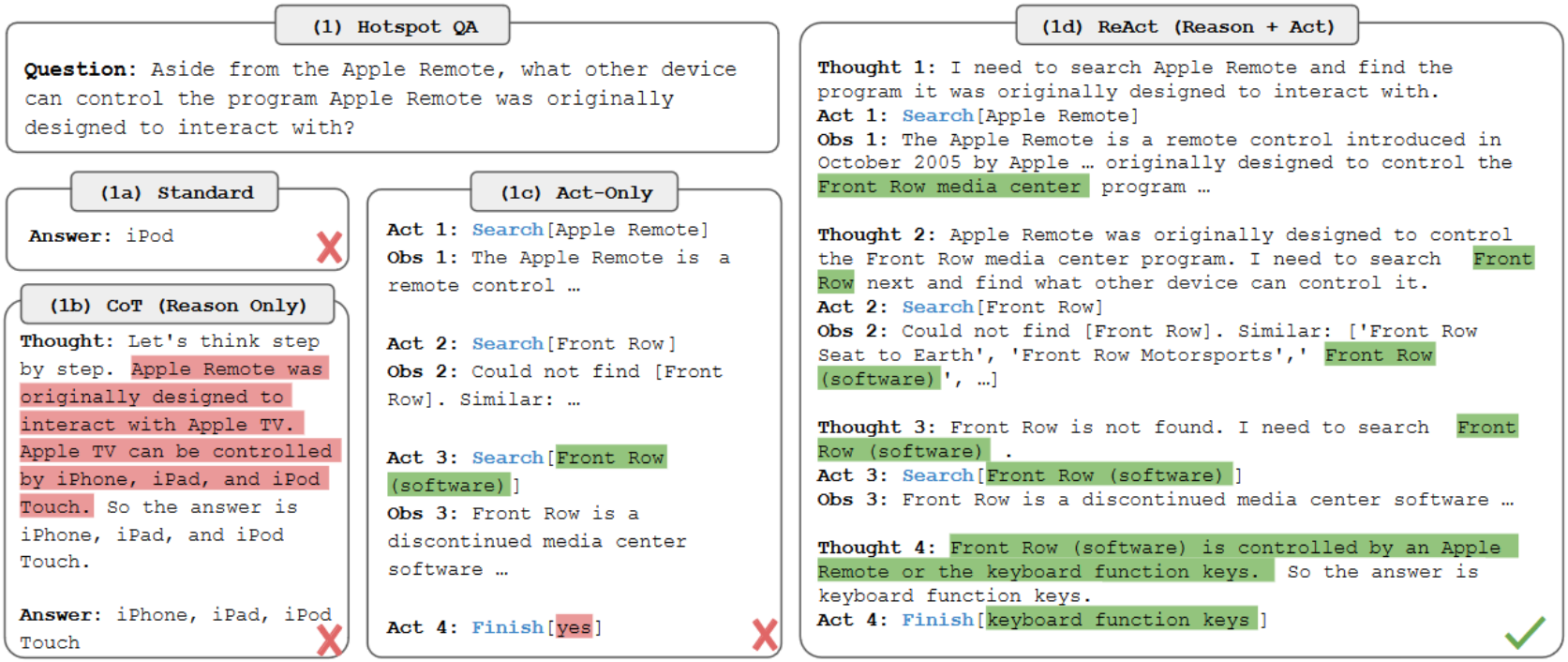


Figure 1: The self-consistency method contains three steps: (1) prompt a language model using chain-of-thought (CoT) prompting; (2) replace the "greedy decode" in CoT prompting by sampling from the language model's decoder to generate a diverse set of reasoning paths; and (3) marginalize out the reasoning paths and aggregate by choosing the most consistent answer in the final answer set.

https://arxiv.org/pdf/2201.11903

ReAct

- Generating reasoning traces allow the model to induce, track, and update action plans, and even handle exceptions.
- Allow models to take actions: Use external tools.
- System prompt (leaked) of ChatGPT: https://github.com/0xeb/TheBigPromptLibrary/blob/main/SystemPrompts/OpenAI/gpt4o_10192024.md



https://arxiv.org/abs/2210.03629

Summary

- Prompt engineering is an important starting point of LLM performance improvement.
- As foundation model improves, some prompt engineering techniques are by default implemented. But it is still important to understand them as generic models are not guaranteed to work everywhere.

CentraleSupélec

Thank you! Questions?