

Tutorial 1: Prompt Engineering

Guidance for Assignment 1

CSC 6203 Large Language Models

Junying Chen | Sep. 13, 2024

Context

1. Prompt Engineering

1. Assignment One of Our Course

1. Colab Practice

First of All, a Reminder

1. Please visit our course website more frequently. We will update the course materials and the latest updates on our website.

Course website: <https://llm-course.github.io>

CSC 6203 Large Language Models

Teaching Complex B201, Friday 13:30-16:20, Sep. 4th - Dec. 13, 2024

Autumn 2024

This course offers a comprehensive study of Large Language Models (LLMs). We'll explore architecture engineering, training techniques, efficiency enhancements, and prompt engineering. Students will gain insights into the application of LLMs in various domains, tools integration, privacy and bias issues, as well as their limitations and alignment. The curriculum includes guest lectures on advanced topics and in-class presentations to stimulate practical understanding. This course is ideal for anyone seeking to master the use of LLMs in their field.

本课程提供对大语言模型 (LLM) 的全面学习。我们将探索大模型的架构工程、提示工程、训练技术、效率提升。学生将深入了解大语言模型在各个领域的应用、工具集成、隐私和偏见问题及其局限性和对齐。该课程包括高级主题的客座讲座和课堂演示，以激发实践理解。本课程对于任何想要掌握大语言模型在其领域的使用的人来说都是理想的选择。

2. **Join our WeChat Group:** Please join our course WeChat group to stay updated and connect with classmates. This will also help ensure that assignments are submitted on time :)



1. Prompt Engineering

What is Prompt engineering

Explanation by OpenAI

Prompt engineering are strategies and tactics for getting better results from large language models (sometimes referred to as GPT models) like GPT-4o.







More Specific Definitions

An AI prompt is a carefully crafted instruction given to an AI model to generate a specific output. These outputs can range from text and images to videos or even music.

Prompt engineering means writing precise instructions that guide AI models like **ChatGPT** to produce specific and useful responses. It involves designing inputs that an AI can easily understand and act upon, ensuring the output is relevant and accurate.

Why it's important

Prompt engineering is essential for improving the performance of AI in various tasks, such as answering customer inquiries, generating content, or analyzing data. [Compare two ChatGPT examples below:](#)

Example 1	Example 2
<p data-bbox="511 502 904 543">Describe the process of wine production.</p> <p data-bbox="511 584 873 655"> Wine production involves harvesting grapes, fermenting them, aging the wine, and then bottling it.</p> <p data-bbox="511 955 904 988"> Message ChatGPT </p>	<p data-bbox="966 502 1358 584">Describe the process of wine production in three steps: harvesting, fermentation, and aging. Include one detail for each step.</p> <p data-bbox="966 638 1358 868"> <ol style="list-style-type: none">1. Harvesting: Grapes are picked at peak ripeness to ensure the best flavor.2. Fermentation: The grapes are crushed and fermented, turning sugars into alcohol.3. Aging: The wine is aged in barrels to develop its flavor before bottling.</p> <p data-bbox="966 955 1358 988"> Message ChatGPT </p>

If we refine our request, we get a response that aligns more with what we need.

Why it's important

Rather than a rigid technical skill, prompt engineering is more about mastering the art of communicating with AI to achieve **consistent, controllable, and repeatable** results.

Although the term "**engineering**" suggests a highly technical process, it's actually all about using strategic thinking and problem-solving to interact effectively with AI.

More Clearer Examples

Prompt for a text-to-image model

```
photography, person standing in front of the Eiffel  
Tower at sunrise, wearing a red coat and holding a cup  
of coffee --ar 3:2
```

If we change the first word, we get images that will emulate different styles.



[1] Photography



[2] Painting








[3] Pencil drawing



[4] Claymation

The application of prompt engineer?

Prompt engineering can be applied to most large models that can accept text input. There have so many common use cases.

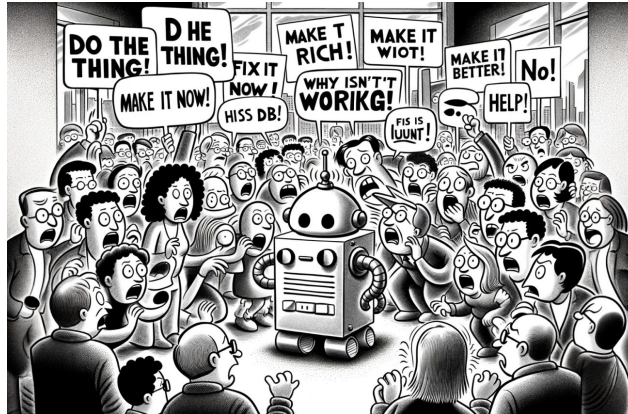
Popular prompt engineering applications		
 Customer Service	Handle specific types of customer queries, ensuring accurate and context-aware responses.	"Provide a solution for a common billing issue" vs. "Address a customer complaint about a late fee."
 Content Creation	Match desired tone, style, and structure, enhancing the relevance of generated text.	"Write a friendly blog post about travel tips" vs. "Generate a formal report on market trends."
 Data Analysis and Document Processing	Design prompts to target specific data points and formats, improving the precision and usefulness of extracted information.	"Summarize the key findings of this financial report" vs. "Extract all instances of revenue data from these documents."
 Educational Tools	Adapt to different learning levels and topics, creating more effective educational content.	"Explain the water cycle for a 5th-grade science class" vs. "Describe the water cycle for college-level environmental science."
 Image Generation	Give detailed specifications to guide AI in producing images with specific styles and elements.	"Generate an image of a futuristic cityscape at night" vs. "Create a drawing of a medieval castle in daylight."

Limitations of prompt engineering

When it comes to prompt engineering, there's both good news and bad news.

The good news is that AI has reached a point where it can understand natural language (NLP technologies), allowing us to express our needs in plain, descriptive terms without needing to code.

The “bad” news, however, is that you still need to have a clear understanding of what you want, **describe it in detail, and communicate your request effectively.**



Prompting Techniques

To get the most out of your prompts, you should understand different types of AI prompts.

Next, we will introduce some common prompting techniques.

1. Zero-Shot Prompting

Large language models (LLMs) today, such as GPT-3.5 Turbo, GPT-4, and Claude 3, are tuned to follow instructions and are trained on so many data. Large-scale training makes these models capable of performing some tasks in a "zero-shot" manner.

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.

Prompt:

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

Output:

```
Neutral
```

Note that in the prompt above we didn't provide the model with any examples of text alongside their classifications, the LLM already understands "sentiment" – that's the zero-shot capabilities at work.

2. Few-Shot Prompting

While large-language models demonstrate remarkable zero-shot capabilities, they still fall **short on more complex tasks when using the zero-shot setting**.

Few-shot prompting can be used as a technique to enable in-context learning where we provide demonstrations in the prompt to steer the model to better performance. The demonstrations serve as conditioning for subsequent examples where we would like the model to generate a response.

Prompt:

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



Output:

```
Negative
```

We still get the correct answer, even though the labels have been randomized. Note that we also kept the format, which helps too. In fact, with further experimentation, it seems the newer GPT models we are experimenting with are becoming more robust to even random formats.

3. Chain-of-Thought Prompting

Chain-of-thought (CoT) prompting enables complex reasoning capabilities through intermediate reasoning steps. You can combine it with few-shot prompting to get better results on more complex tasks that require reasoning before responding.

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. ✅

3. Chain-of-Thought Prompting

Prompt:

```
The odd numbers in this group add up to an even number: 4, 8, 9, 15, 12, 2, 1.  
A: Adding all the odd numbers (9, 15, 1) gives 25. The answer is False.  
The odd numbers in this group add up to an even number: 17, 10, 19, 4, 8, 12, 24.  
A: Adding all the odd numbers (17, 19) gives 36. The answer is True.  
The odd numbers in this group add up to an even number: 16, 11, 14, 4, 8, 13, 24.  
A: Adding all the odd numbers (11, 13) gives 24. The answer is True.  
The odd numbers in this group add up to an even number: 17, 9, 10, 12, 13, 4, 2.  
A: Adding all the odd numbers (17, 9, 13) gives 39. The answer is False.  
The odd numbers in this group add up to an even number: 15, 32, 5, 13, 82, 7, 1.  
A:
```

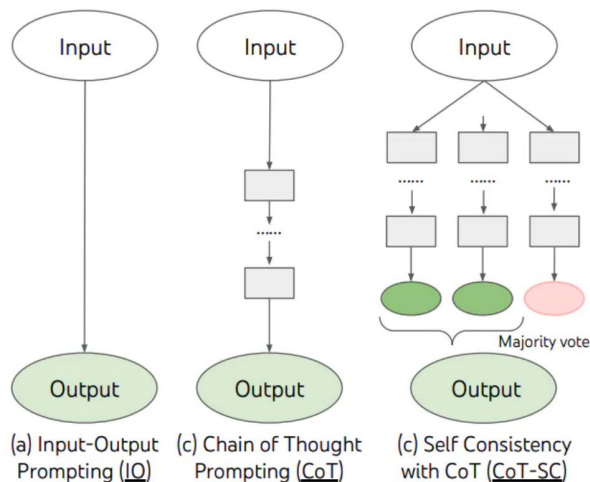
Output:

```
Adding all the odd numbers (15, 5, 13, 7, 1) gives 41. The answer is False.
```

Wow! We can see a perfect result when we provided the reasoning step.

4. Self-Consistency Prompting

Self-Consistency aims "to replace the naive greedy decoding used in chain-of-thought prompting". The idea is to sample multiple, diverse reasoning paths through few-shot CoT, and use the generations to select the most consistent answer. This helps to boost the performance of CoT prompting on tasks involving arithmetic and commonsense reasoning.

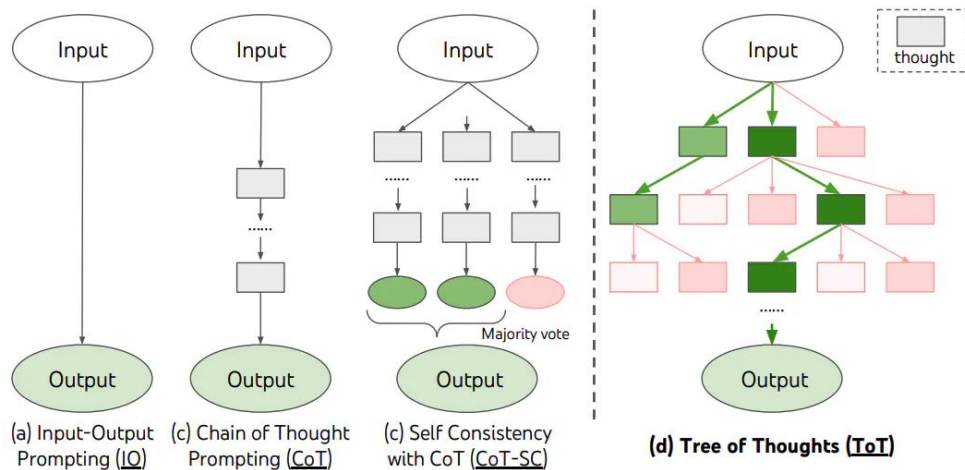


Computing for the final answer involves a few steps but for the sake of simplicity, we can see that there is already a majority answer emerging so that would essentially become the final answer.

5. Tree of Thoughts (ToT) Prompting

Tree of Thoughts (ToT) is a framework that generalizes over chain-of-thought prompting and encourages exploration over thoughts that serve as intermediate steps for general problem solving with language models.

ToT maintains a tree of thoughts, where thoughts represent coherent language sequences that serve as intermediate steps toward solving a problem. This approach enables an LM to self-evaluate the progress through intermediate thoughts made towards solving a problem through a deliberate reasoning process.



The LM's ability to generate and evaluate thoughts is then combined with search algorithms (e.g., breadth-first search and depth-first search) to enable systematic exploration of thoughts with lookahead and backtracking.

6. Retrieval Augmented Generation (RAG)

Retrieval Augmented Generation (RAG) takes an input and retrieves a set of relevant/supporting documents given a source (e.g., Wikipedia). The documents are concatenated as context with the original input prompt and fed to the text generator which produces the final output.

This makes RAG adaptive for situations where facts could evolve over time. This is very useful as LLMs's parametric knowledge is static. RAG allows language models to bypass retraining, enabling access to the latest information for generating reliable outputs via retrieval-based generation.

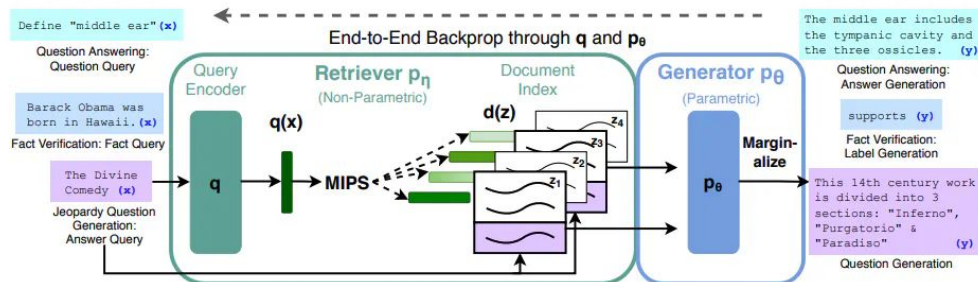


Figure 1: Overview of our approach. We combine a pre-trained retriever (*Query Encoder + Document Index*) with a pre-trained seq2seq model (*Generator*) and fine-tune end-to-end. For query x , we use Maximum Inner Product Search (MIPS) to find the top-K documents z_i . For final prediction y , we treat z as a latent variable and marginalize over seq2seq predictions given different documents.

Meta AI researchers introduced Retrieval Augmented Generation (RAG) to address **knowledge-intensive tasks**.

7. Automatic Reasoning and Tool-use (ART)

Automatic Reasoning and Tool-use (ART) encourages the model to generalize from demonstrations to decompose a new task and use tools in appropriate places, in a zero-shot fashion. In addition, ART is extensible as it also enables humans to fix mistakes in the reasoning steps or add new tools by simply updating the task and tool libraries. The process is demonstrated below:

ART works as follows:

- given a new task, it select demonstrations of multi-step reasoning and tool use from a task library
- at test time, it pauses generation whenever external tools are called, and integrate their output before resuming generation

Best practices in prompt engineering

To be a better prompt engineer, you may follow the best practices listed below:

1. **Clarity and precision.** Always be clear and precise in your instructions. Ambiguities can lead to varied interpretations and outputs, which may not meet your needs.
2. **Iterative refinement.** Start with a basic prompt and refine it based on the responses you get. This process helps in fine-tuning the AI's outputs to your specific requirements.
3. **Use of keywords.** Incorporate relevant keywords and specific details that can guide the AI more effectively towards the desired output.
4. **Understanding the model's limitations.** Be aware of what the AI can and cannot do. This understanding will help you craft prompts that are within the capabilities of the model, avoiding overly complex requests that lead to poor responses.
5. **Feedback loop.** Utilize feedback to continuously improve the prompts. Feedback from users or the outputs themselves can provide valuable insights into how prompts can be adjusted for better results.
6. **Prompt length.** Mind the length of your prompts. If the instructions are too lengthy they can confuse the AI. It will also lead to higher token consumption and higher costs if you are deploying an AI-powered solution for your users and customers.
7. **Ethical considerations.** Ensure that the prompts do not encourage the AI to generate harmful or biased content. Being ethical in your prompt engineering is crucial for responsible AI use.

2. Assignment One of Our Course

Assignment 1

You will be asked to employ **prompt engineering** techniques to complete a task using LLMs. We offer six optional tasks, or you can choose one that personally interests you. We encourage you to pick a task that you find both challenging and enjoyable.

We will provide a detailed PDF file with assignment instructions:

https://llm-course.github.io/Assignments/Assignment1/Assignment_1_Prompt_Engineering.pdf

Assignment 1 Deadline

- **Deadline:** 2024. 10. 18

Take it easy, but don't forget to submit on time.

3. Colab Practice

Practice of Prompt Engineering

If you're new to using LLMs, don't worry; we've prepared a [Colab notebook](#) for you:

https://colab.research.google.com/drive/1JFtkSnT_Sik8vlqvAXB8iXDKwMiQrHyf?usp=sharing

This will guide you on

1. How to efficiently call the GPT-4 API.
2. Useful Prompt Engineering techniques.
3. How to get started with Task 1 and Task 2.

Thanks

That's all for today's class, and you are now free to leave!