Prompt Engineering for LLMs

In the realm of natural language processing, the art of crafting effective prompts plays a pivotal role in harnessing the full potential of LLMs. This tutorial offers a practical guide for data scientists and machine learning practitioners to strategically design prompts that optimize the performance of LLMs. The focus of this tutorial will be on ChatGPT and Gemini, demonstrating best practices and methods for key use-cases in prompt engineering.

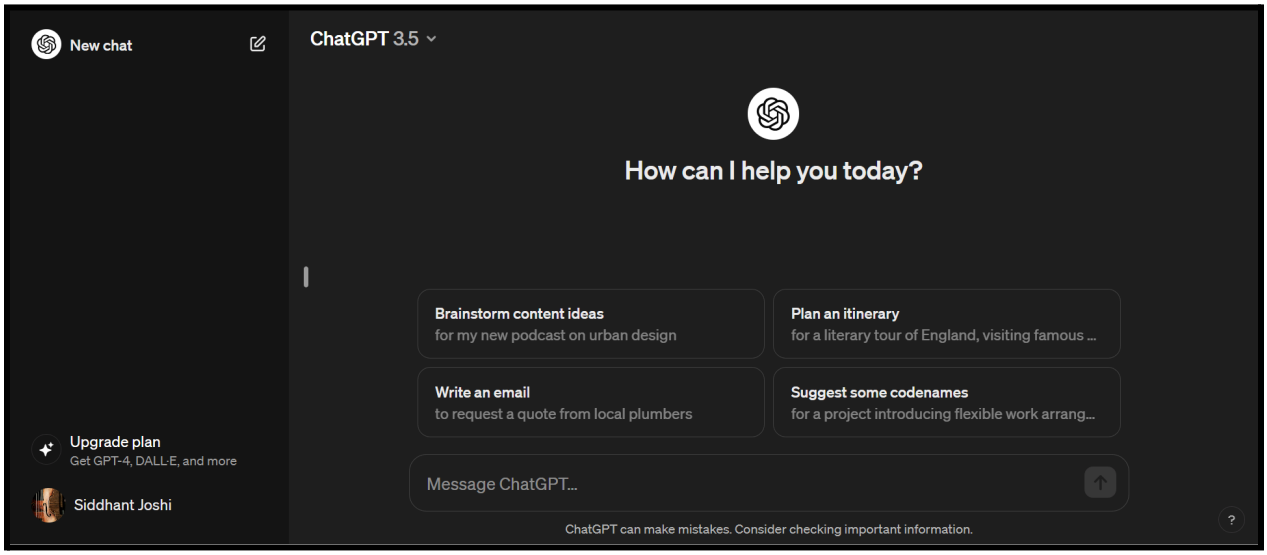
Table of Contents:

Setting up the Environment

OpenAI's ChatGPT

ChatGPT is an AI chatbot developed by OpenAI based on the GPT (Generative Pre-trained Transformer) architecture. It is designed to engage in natural language conversations with users, providing responses that are contextually relevant and coherent. The model is capable of engaging in conversations with user-based prompts, answer questions/provide information, and is a general purpose LLM. It also has a subscription version, called ChatGPT-4, which includes all of the original features in addition to image generation (via DALL-E) and web browsing.

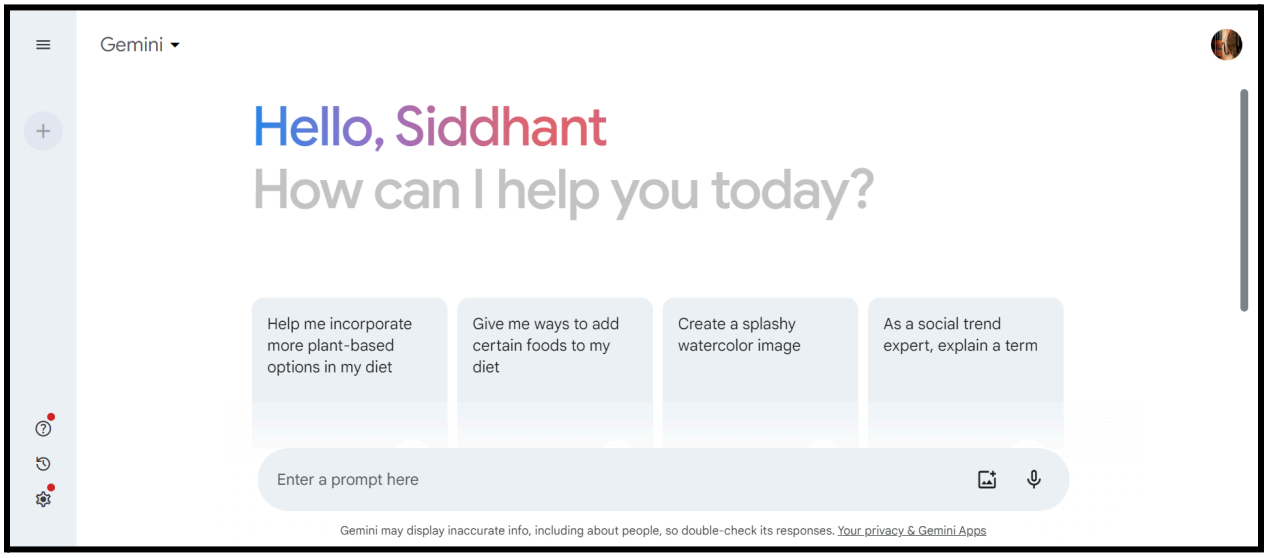
1. Navigate to https://chat.openai.com/auth/login and select *Sign Up* to create an account.
2. After creating an account, navigate back to the above link and login to your account.
3. The screen should look like the image below:



Google's Gemini

Formerly known as Bard, Gemini is an LLM developed and deployed by Google. It functions similarly to MS Copilot, however, it lacks the ability to tune conversation styles and domain-specific models. It runs on software managed by Google and is designed to engage in conversations with user-based prompts, answer questions, and put together responses based on information it gathers from the Internet.

1. Navigate to https://gemini.google.com/app and select or create a Google account (you may need to use a personal gmail account if your organization email does not work).
2. After creating an account, navigate back to the above link and you should see the following:



Prompt Refining

Prompt refining in the context of prompt engineering refers to the process of reformulating and rephrasing prompts to achieve more accurate and relevant outputs from AI models. This involves adjusting the wording, structure, and specificity of the prompts to guide the AI effectively. By refining prompts, users can enhance the clarity and precision of their queries, leading to better performance and results from the AI.

For best practices on how to effectively engineer prompts, refer to this guide from OpenAI.

Exercise 1:

PROMPT:
Tell me about data visualization.
REFINED PROMPT:
Insert your response here…

Exercise 2:

PROMPT:
Find a nice dataset that contains and includes petal lengths. Explore the data and tell me what results you get from exploring the data.
REFINED PROMPT:
Insert your response here…

Prompt Chaining

Prompt chaining in the context of prompt engineering involves linking multiple prompts together to facilitate more complex and coherent dialogues. This technique allows for building upon previous responses, enabling the AI to maintain context and continuity throughout the conversation. By chaining prompts, users can create sophisticated interactions that mimic natural, multi-turn conversations, enhancing the depth and relevance of the AI's responses.

Exercise 1:

PROMPT: How do transformers work?
CHATGPT: *Insert your response here…*
GEMINI: *Insert your response here…*

Exercise 2:

PROMPT: Give me a more detailed explanation of the self-attention mechanism.
CHATGPT: *Insert your response here…*
GEMINI: *Insert your response here…*

Exercise 3:

PROMPT: Give me a conceptual explanation of the Q, K, and V matrices.
CHATGPT: *Insert your response here…*
GEMINI: *Insert your response here…*

Shot Prompting and Chain-of-Thought Prompting

Chain-of-thought prompting is a technique where the prompt asks the AI to provide step-by-step reasoning, allowing it to break down complex problems into manageable parts. This approach helps in understanding the logic and sequence behind the AI's responses, ensuring more accurate and transparent outputs. On the other hand, shot prompting involves providing examples within the prompt to guide the AI's responses. By including these examples, users can set clear expectations and demonstrate the desired format or style, improving the relevance and quality of the outputs. Together, these methods enhance the AI's ability to handle intricate queries and deliver precise, context-aware results.

In this section, we want the answer to the following problem:

> "John's ship can travel at 7 miles per hour. He is sailing from 11 AM to 3 PM. He then travels back at a rate of 8 mph. How long does it take him to get back?"

The first step is to write the problem as a **question-answer pair**. This approach is commonly used for training or querying language models, especially for mathematical reasoning or word problems. In this format, you present a clear question followed by an answer. Current LLMs are trained in a way that encourages detailed and comprehensive responses, which can increase the likelihood of providing accurate information. Therefore, it is likely that you will receive accurate answers even without explicitly implementing chain-of-thought prompting. However, for more complex problems where explicit chain-of-thought prompting is necessary, the strategies learned from this example will still be applicable.

⏳ Using question-answer pair format:

PROMPT:
John's ship can travel at 7 miles per hour. He is sailing from 11 AM to 3 PM. He then travels back at a rate of 8 mph. How long does it take him to get back?
REVISED PROMPT:
Insert your response here…

Now, create a similar problem to the original and find its answer (just the number of hours, not the solution). Place this new question-answer pair before the original question.

This is known as shot prompting, where examples are provided within the prompt to guide the AI's responses. Generally, there are three different levels: zero-shot, one-shot, and few-shot prompting. Zero-shot prompting involves asking the AI to respond to a prompt without any examples. One-shot prompting provides one example to illustrate the desired response. Few-shot prompting includes a few examples to give the AI a better understanding of the task.

⏳ Adding another question:

PROMPT:

Question: Tom's ship can travel at 10 miles per hour. He is sailing from 1 to 4 PM. He then travels back at a rate of 6 mph. How long does it take him to get back?
Answer: 5 hours.

Question: John's ship can travel at 7 miles per hour. He is sailing from 11 AM to 3 PM. He then travels back at a rate of 8 mph. How long does it take him to get back?
Answer:

CHATGPT:

Insert your response here…

GEMINI:

Insert your response here…

Now, instead of providing just a brief answer,, we will include detailed reasoning and steps required to reach the answer. Expand your answer into a full solution. Prompt the LLMs again and observe the differences. This approach is known as chain-of-thought prompting. This works because it encourages the LLMs to get answers to multiple small answers (which it is good at) instead of getting an answer to one large question (which it is not great at).

✅ Using chain-of-thought prompting:

PROMPT:

Question: Tom's ship can travel at 10 miles per hour. He is sailing from 1 to 4 PM. He then travels back at a rate of 6 mph. How long does it take him to get back?
Answer:
1. Outbound Journey: Calculate the distance traveled by multiplying the speed (10 mph) by the time (3 hours). Tom traveled 30 miles.
2. Return Journey: Divide the outbound distance (30 miles) by the return speed (6 mph) to find the return time. Tom returned in 5 hours.

Question: John's ship can travel at 7 miles per hour. He is sailing from 11 AM to 3 PM. He then travels back at a rate of 8 mph. How long does it take him to get back?
Answer:

CHATGPT:

Insert your response here…

GEMINI:

Insert your response here…

Summarization

Summarization in the context of LLM prompt engineering involves creating concise and coherent summaries of longer texts. By crafting prompts that ask the AI to distill the main points or key information from a document, users can efficiently generate summaries that capture the essence of the content, making it easier to understand and digest large volumes of information quickly.

Exercise:

PROMPT: Explain the main findings of “Attention is All You Need” by Vaswani et al.
CHATGPT: *Insert your response here…*
GEMINI: *Insert your response here…*

Content Generation

Content generation leverages LLM prompt engineering to create various forms of content, such as articles, stories, or social media posts. By designing prompts that specify the topic, tone, and style, users can guide the AI to produce high-quality and contextually appropriate content. This application is particularly useful for automating content creation processes and enhancing creative workflows.

Exercise:

PROMPT: Create a resume for an undergraduate CS student at UCSD looking for jobs in machine learning and data science.
CHATGPT: *Insert your response here…*
GEMINI: *Insert your response here…*

Translation

Translation in LLM prompt engineering involves converting text from one language to another while maintaining the original meaning and context. By structuring prompts to indicate the source and target languages, users can utilize the AI's capabilities to perform accurate and nuanced translations, facilitating communication and understanding across different languages and cultures.

Exercise 1:

PROMPT: Can you translate this from French to English: Je ne parle Francais.
CHATGPT: *Insert your response here…*
GEMINI: *Insert your response here…*

Exercise 2:

PROMPT: Can you translate to English: Toi la nguoi Viet.
CHATGPT: *Insert your response here…*
GEMINI: *Insert your response here…*

Exercise 3:

PROMPT: How do you say this in Japanese: Sorry, I don't speak Japanese
CHATGPT: *Insert your response here…*
GEMINI: *Insert your response here…*

Exercise 4:

PROMPT: What does this mean: Hella fit; at first I was triggered and shook, but then looked closer, saw it was popping and legit, in fact it is totally lit!
CHATGPT: *Insert your response here…*
GEMINI: *Insert your response here…*

Code Generation

Code generation uses LLM prompt engineering to create snippets of code based on specific requirements or descriptions. By formulating prompts that outline the desired functionality or logic, users can have the AI generate code in various programming languages. This application is valuable for accelerating development processes and assisting with coding tasks.

Exercise 1:

PROMPT:
Can you provide Python code to read in a dataframe with Name, Dept, remove missing values, then group by Dept? Write results to a json file.
CHATGPT:
Insert your response here…
GEMINI:
Insert your response here…

Exercise 2 (follow-up from Exercise 1):

PROMPT:
Can you provide the equivalent PySpark code?
CHATGPT:
Insert your response here…
GEMINI:
Insert your response here…

Exercise 3:

PROMPT: Write Python code that uses Pandas to remove outliers from the ./titanic.csv dataset by limiting the 'Age' and 'Fare' columns to values within the 1st and 99th percentiles. Report the number of outliers before and after handling.
CHATGPT: *Insert your response here…*
GEMINI: *Insert your response here…*

Code Interpretation

With programming as the key method with which to implement our ideas, it is important to have a system to produce and accurately interpret code. Python is a prominent programming language in the data scientist's toolbox and LLMs can be leveraged to interpret segments of code accurately and efficiently. Below we offer some examples of how to formulate prompts to get the most effective interpretations of code.

Exercise[1]:

PROMPT:

Explain the following code:

```
def find_max(list):
    max_val = list[0]
    for i in range(len(list)):
        for j in range(i, len(list)):
            if list[j] > max_val:
                max_val = list[j]
    return max_val
```

CHATGPT:

Insert your response here…

GEMINI:

Insert your response here…

[1] The code in the exercise prompt is from Konfuzio: https://konfuzio.com/en/python-tutorial-complexity/