Prompting Techniques > Prompt Chaining

Prompt Chaining

Introduction to Prompt Chaining

To improve the reliability and performance of LLMs, one of the important prompt engineering techniques is to break tasks into its subtasks. Once those subtasks have been identified, the LLM is prompted with a subtask and then its response is used as input to another prompt. This is what's referred to as prompt chaining, where a task is split into subtasks with the idea to create a chain of prompt operations.

Prompt chaining is useful to accomplish complex tasks which an LLM might struggle to address if prompted with a very detailed prompt. In prompt chaining, chain prompts perform transformations or additional processes on the generated responses before reaching a final desired state.

Besides achieving better performance, prompt chaining helps to boost the transparency of your LLM application, increases controllability, and reliability. This means that you can debug problems with model responses much more easily and analyze and improve performance in the different stages that need improvement.

Prompt chaining is particularly useful when building LLM-powered conversational assistants and improving the personalization and user experience of your applications.

🎓 Learn more about prompt chaining and advanced prompting methods in our new AI courses. [Join now!](#)

Use Cases for Prompt Chaining

Prompt Chaining for Document QA

Prompt chaining can be used in different scenarios that could involve several operations or transformations. For instance, one common use case of LLMs involves answering questions about a large text document. It helps if you design two different prompts where the first prompt is responsible for extracting relevant quotes to answer a question and a second prompt takes as input the quotes and original document to answer a given question. In other words, you will be creating two different prompts to perform the task of answering a question given in a document.

The first prompt below extracts the relevant quotes from the document given the question. Note that for simplicity, we have added a placeholder for the document `{{document}}`. To test the prompt you can copy
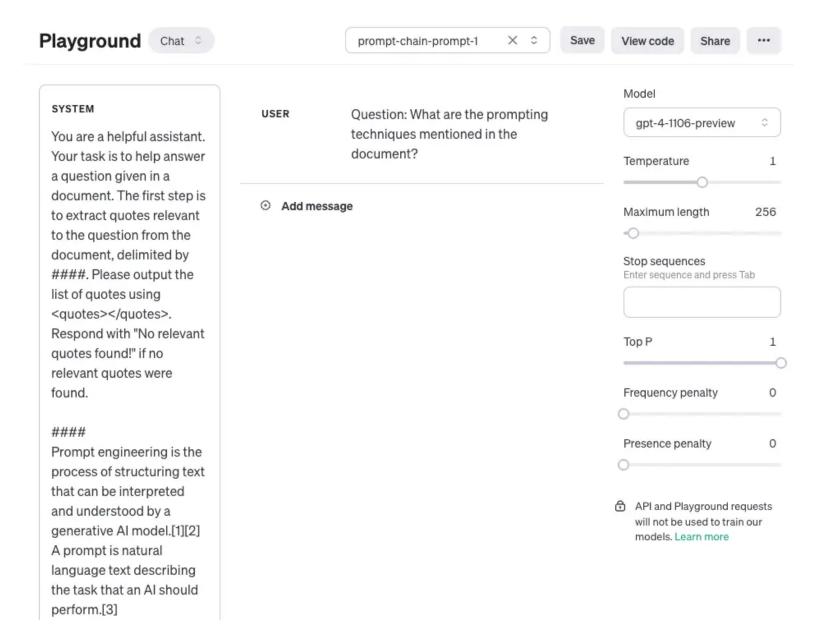
and paste an article from Wikipedia such as this page for [prompt engineering](). Due to larger context used for this task, we are using the `gpt-4-1106-preview` model from OpenAI. You can use the prompt with other long-context LLMs like Claude.

Prompt 1:

```
You are a helpful assistant. Your task is to help answer a question given in a document. The first step is
to extract quotes relevant to the question from the document, delimited by ####. Please output the list of
quotes using <quotes></quotes>. Respond with "No relevant quotes found!" if no relevant quotes were found.
####
{{document}}
####
```

Here is a screenshot of the entire prompt including the question which is passed using the `user` role.

Playground

SYSTEM

You are a helpful assistant. Your task is to help answer a question given in a document. The first step is to extract quotes relevant to the question from the document, delimited by ####. Please output the list of quotes using <quotes></quotes>. Respond with "No relevant quotes found!" if no relevant quotes were found.

####
Prompt engineering is the process of structuring text that can be interpreted and understood by a generative AI model.[1][2] A prompt is natural language text describing the task that an AI should perform.[3]

USER   Question: What are the prompting techniques mentioned in the document?

⊙ **Add message**

Model

gpt-4-1106-preview ⇕

Temperature   1

Maximum length   256

Stop sequences
Enter sequence and press Tab

Top P   1

Frequency penalty   0

Presence penalty   0

🔒 API and Playground requests will not be used to train our models. Learn more

Output of Prompt 1:

```
<quotes>
- Chain-of-thought (CoT) prompting[27]
- Generated knowledge prompting[37]
- Least-to-most prompting[38]
```

```
- Self-consistency decoding[39]
- Complexity-based prompting[41]
- Self-refine[42]
- Tree-of-thought prompting[43]
- Maieutic prompting[45]
- Directional-stimulus prompting[46]
- Textual inversion and embeddings[59]
- Using gradient descent to search for prompts[61][62][63][64]
- Prompt injection[65][66][67]
</quotes>
```

The quotes that were returned in the first prompt can now be used as input to the second prompt below. Note that you can clean up the quotes a bit more, i.e., remove the citations. Those citations could be removed or utilized as part of another prompt in the chain but you can ignore this for now. The second prompt then takes the relevant quotes extracted by prompt 1 and prepares a helpful response to the question given in the document and those extracted quotes. The second prompt can be the following:

Prompt 2:

```
Given a set of relevant quotes (delimited by <quotes></quotes>) extracted from a document and the original
document (delimited by ####), please compose an answer to the question. Ensure that the answer is
accurate, has a friendly tone, and sounds helpful.
####
{{document}}
####
<quotes>
- Chain-of-thought (CoT) prompting[27]
- Generated knowledge prompting[37]
- Least-to-most prompting[38]
- Self-consistency decoding[39]
- Complexity-based prompting[41]
- Self-refine[42]
- Tree-of-thought prompting[43]
- Maieutic prompting[45]
- Directional-stimulus prompting[46]
- Textual inversion and embeddings[59]
- Using gradient descent to search for prompts[61][62][63][64]
```

```
- Prompt injection[65][66][67]
</quotes>
```

Output of Prompt 2:

```
The prompting techniques mentioned in the document include:
1. Chain-of-thought (CoT) prompting[27]
2. Generated knowledge prompting[37]
3. Least-to-most prompting[38]
4. Self-consistency decoding[39]
5. Complexity-based prompting[41]
6. Self-refine[42]
7. Tree-of-thought prompting[43]
8. Maieutic prompting[45]
9. Directional-stimulus prompting[46]
10. Textual inversion and embeddings[59]
11. Using gradient descent to search for prompts[61][62][63][64]
12. Prompt injection[65][66][67]
Each of these techniques employs unique strategies to enhance or specify the interactions with large
language models to produce the desired outcomes.
```

As you can see, simplifying and creating prompt chains is a useful prompting approach where the responses need to undergo several operations or transformations. As an exercise, feel free to design a prompt that removes the citations (e.g., [27]) from the response before sending this as a final response to the user of your application.

You can also find more examples of prompt chaining in this documentation that leverages the Claude LLM. Our example is inspired and adapted from their examples.

Last updated on January 7, 2025