

Model I/O: Loading Quantized Models with LangChain

Using LangChain to load, run, and interact with quantized (compressed) language models that use fewer bits (like 8-bit, 4-bit) so they run faster and use less memory.

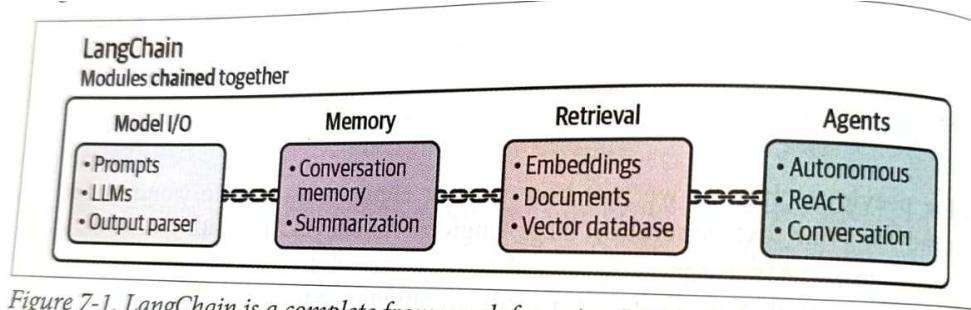


Figure 7-1. LangChain is a complete framework for using LLMs. It has modular components that can be chained together to allow for complex LLM systems.

The most basic form of a chain in LangChain is a single chain. Although a chain can take many forms, each with a different complexity, it generally connects an LLM with some additional tool, prompt, or feature. This idea of connecting a component to an LLM is illustrated in Figure 7-3.

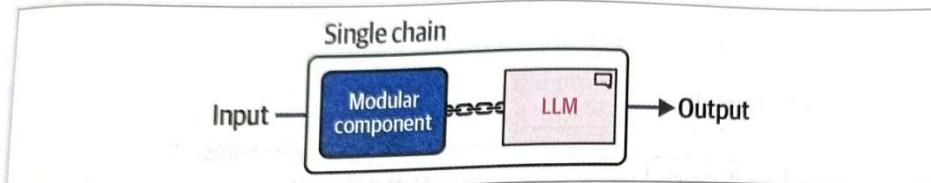


Figure 7-3. A single chain connects some modular component, like a prompt template or external memory, to the LLM.

In practice, chains can become complex quite quickly. We can extend the prompt template however we want and we can even combine several separate chains together to create intricate systems. In order to thoroughly understand what is happening in a chain, let's explore how we can add Phi-3's prompt template to the LLM.

A Single Link in the Chain: Prompt Template

We start with creating our first chain, namely the prompt template that Phi-3 expects. In the previous chapter, we explored how `transformers.pipeline` applies the chat template automatically. This is not always the case with other packages and they might need the prompt template to be explicitly defined. With LangChain, we will use chains to create and use a default prompt template. It also serves as a nice hands-on experience with using prompt templates.

The idea, as illustrated in Figure 7-4, is that we chain the prompt template together with the LLM to get the output we are looking for. Instead of having to copy-paste the prompt template each time we use the LLM, we would only need to define the user and system prompts.

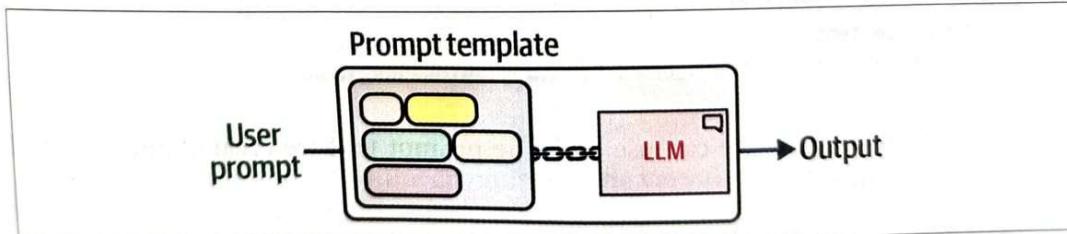


Figure 7-4. By chaining a prompt template with an LLM, we only need to define the input prompts. The template will be constructed for you.

A Chain with Multiple Prompts

In our previous example, we created a single chain consisting of a prompt template and an LLM. Since our example was quite straightforward, the LLM had no issues dealing with the prompt. However, some applications are more involved and require lengthy or complex prompts to generate a response that captures those intricate details.

Instead, we could break this complex prompt into smaller subtasks that can be run sequentially. This would require multiple calls to the LLM but with smaller prompts and intermediate outputs as shown in Figure 7-7.

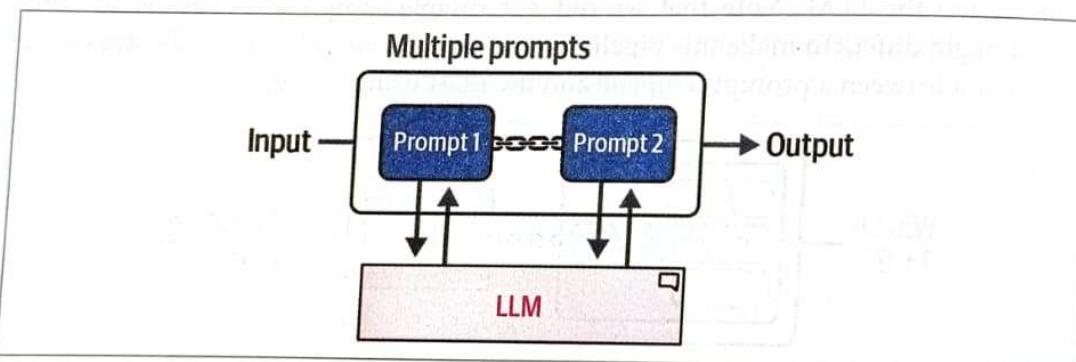


Figure 7-7. With sequential chains, the output of a prompt is used as the input for the next prompt.

This process of using multiple prompts is an extension of our previous example. Instead of using a single chain, we link chains where each link deals with a specific subtask.

For instance, consider the process of generating a story. We could ask the LLM to generate a story along with complex details like the title, a summary, a description of the characters, etc. Instead of trying to put all of that information into a single prompt, we could dissect this prompt into manageable smaller tasks instead.

Let's illustrate with an example. Assume that we want to generate a story that has three components:

- A title
- A description of the main character
- A summary of the story

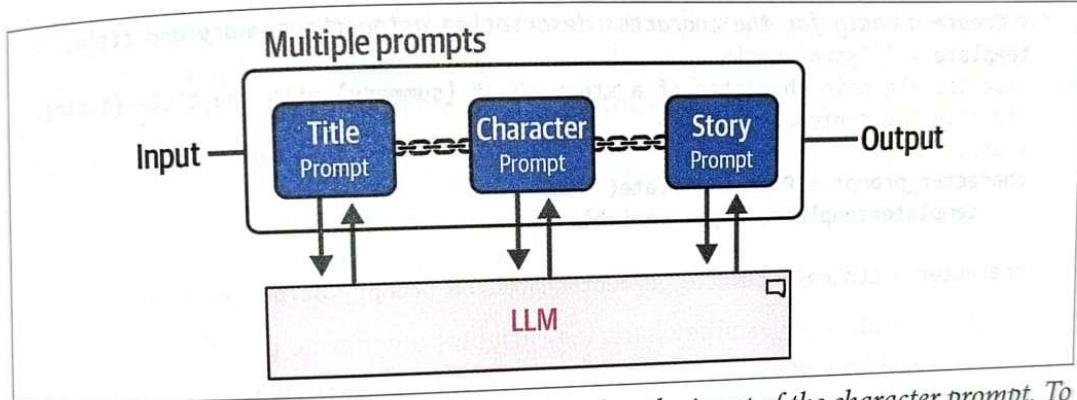


Figure 7-8. The output of the title prompt is used as the input of the character prompt. To generate the story, the output of all previous prompts is used.

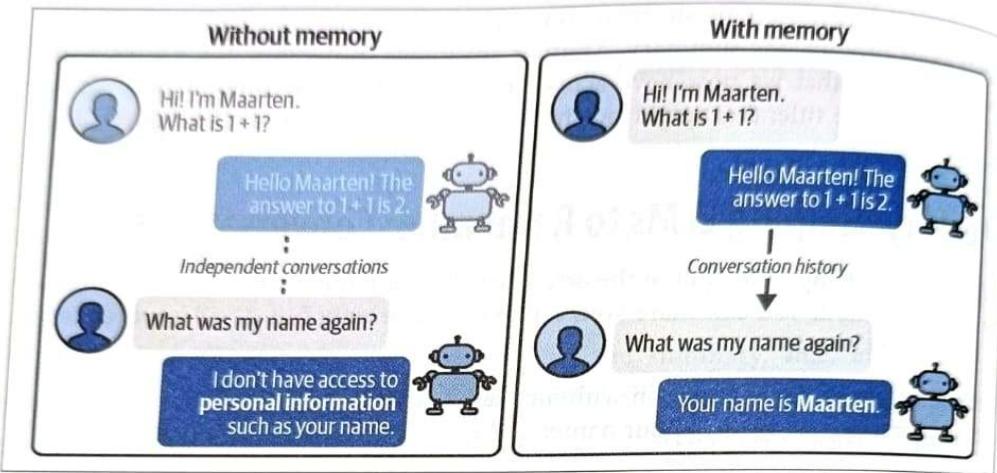


Figure 7-9. An example of a conversation between an LLM with memory and without memory.

Conversation Buffer

One of the most intuitive forms of giving LLMs memory is simply reminding them exactly what has happened in the past. As illustrated in Figure 7-10, we can achieve this by copying the full conversation history and pasting that into our prompt.

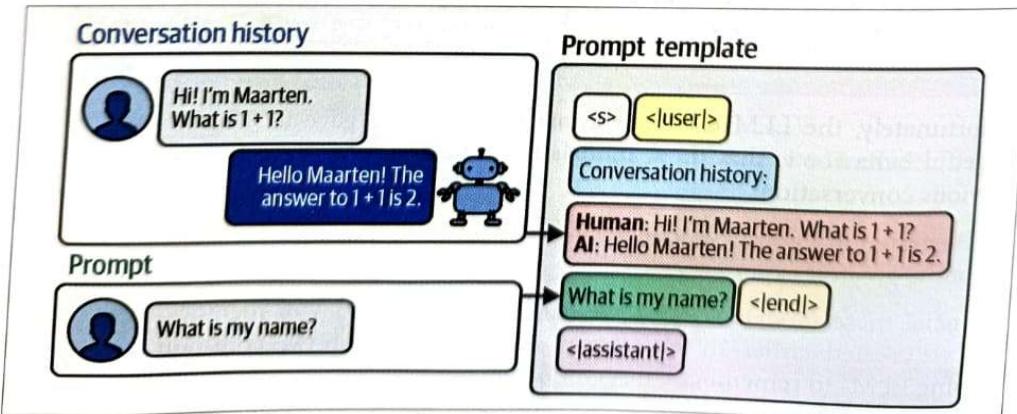


Figure 7-10. We can remind an LLM of what previously happened by simply appending the entire conversation history to the input prompt.

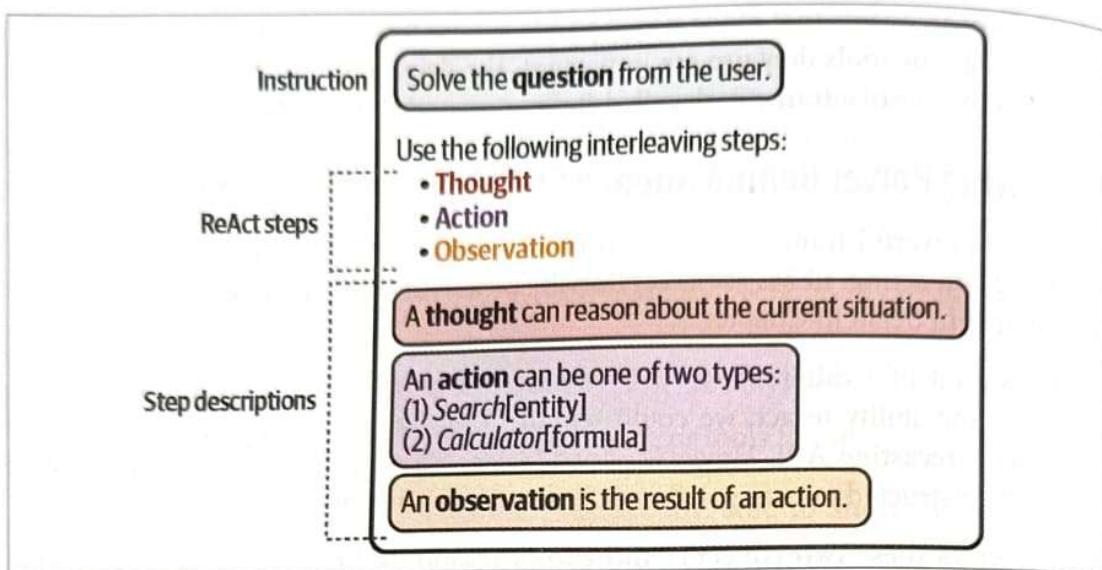


Figure 7-15. An example of a ReAct prompt template.

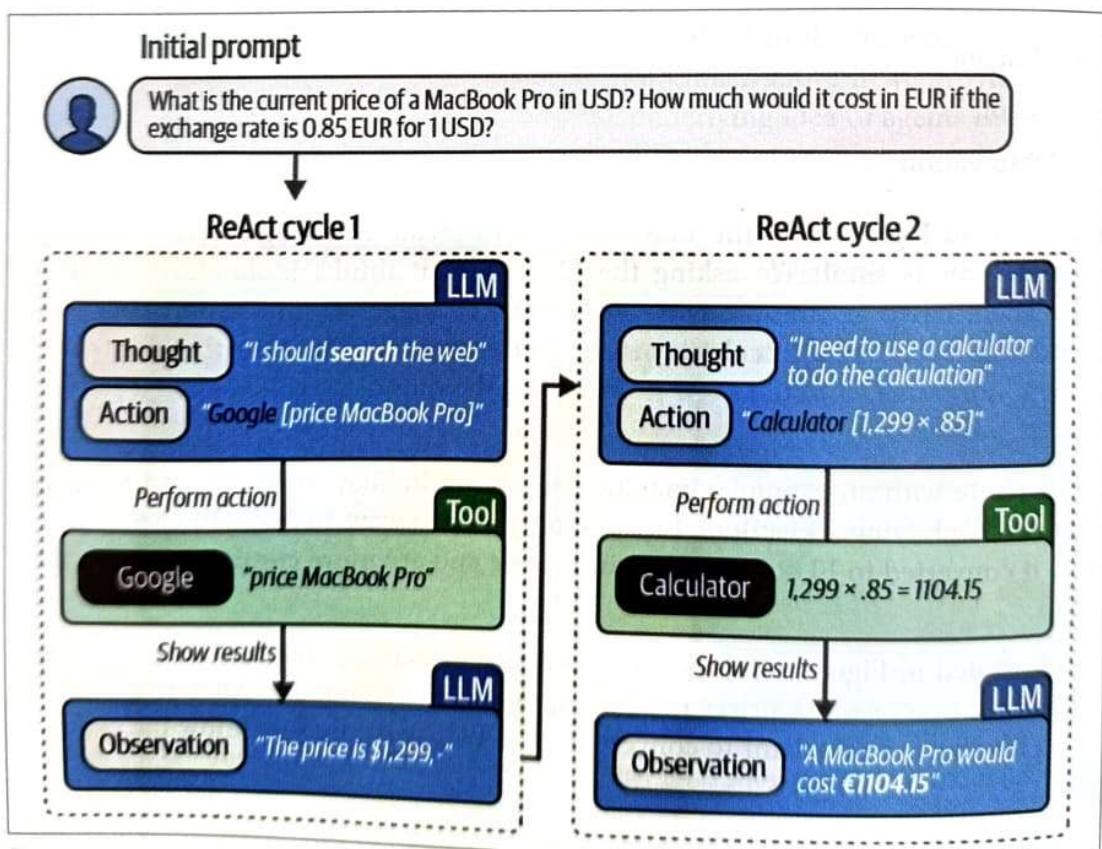


Figure 7-16. An example of two cycles in a ReAct pipeline.