



EECS E6893 Big Data Analytics Generative AI Part 2

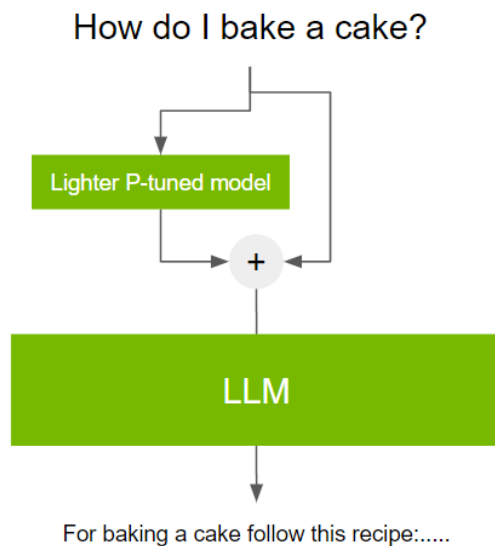
Qingcheng Yu, qy2281@columbia.edu

Agenda

- Prompt
- LangChain
- Chatbot

What is Prompt?

- interact with LLMs to accomplish a task
- a user-provided input to which the model is meant to respond
- include specific constraints or requirements like tone, style, or even desired length of the response



Simple Prompt	Complicated Prompt
Q: How many colors are in the rainbow? A: There are seven colors in the rainbow.	Company Name Number of Employees Year Established IPO Date Share Price Providence Inc. 250 1990 25th August 1990 \$0.90 Grant Corporation 2000 1890 21st September 1920 \$115.90 Rusty Metalworks 12459 1946 12th September 1986 \$15.23
Instruction Prompt Compose a motivational poem to make me happy: Life is full of ups and downs, But don't let it get you down. You can't always get what you want, But if you try sometimes, you might find, You get what you need. If you want to make a difference, Start by doing what's right.	Q: Which company has the most employees? A: Rusty Metalworks Q: What is the share price and IPO date of Grant Corporation? A: \$115.90 and 21st September 1920

Why Prompts are important?

- The quality and relevance of the response generated by the LLM is heavily dependent on the quality of the prompt
- Prompt can solve “hallucination” problem

Was King Renoit real?

King Renoit is not a real historical figure. He is a fictional character from the French epic poem The Song of Roland, written in the 11th century.

Is King Renoit mentioned in the Song of Roland, yes or no?

No. King Renoit is not mentioned in the Song of Roland.

If you ask AI an open-ended question, you received a hallucination.

In fact, King Renoit was never mentioned in the Song of Roland.

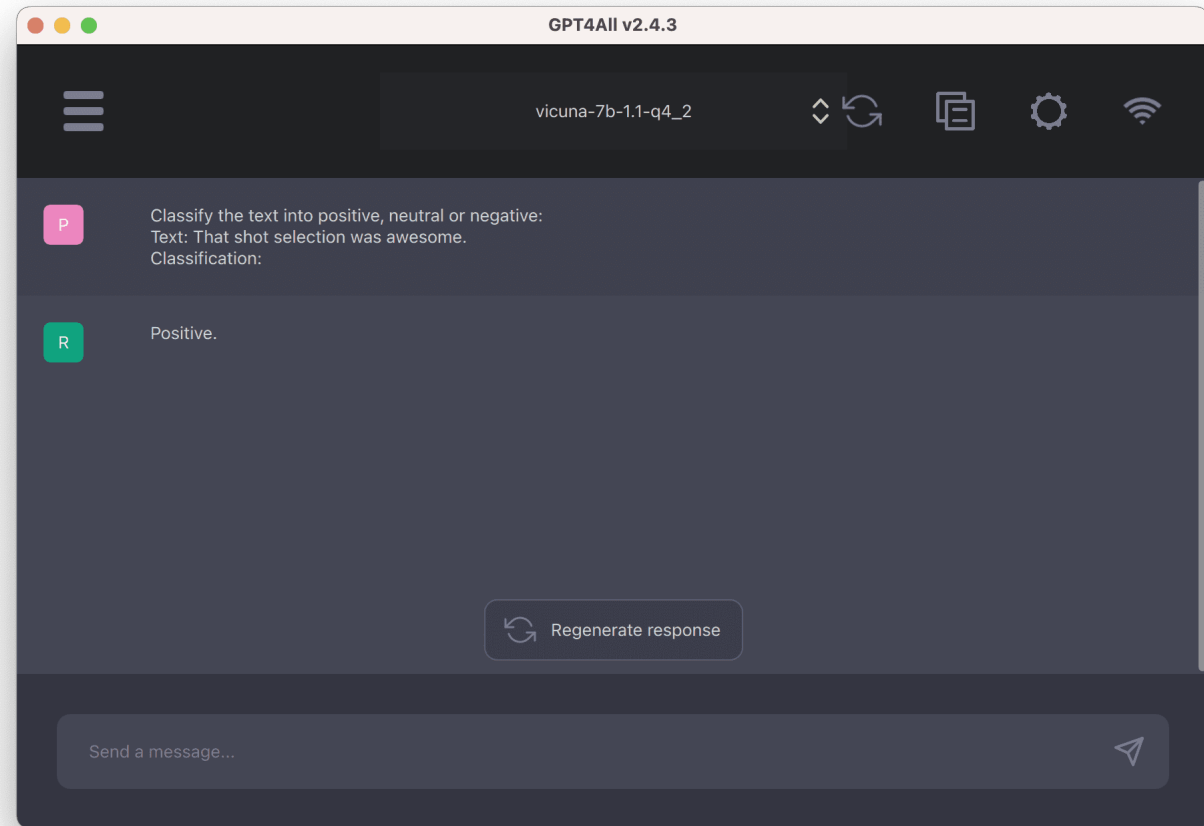
But when you asked it to respond with only "yes" or "no," it corrected itself.

Three ways of prompting

- Zero-Shot Prompting
- Few-Shot Prompting
- Chain-of-thought Prompting

Zero-Shot Prompting

- zero-shot prompting means providing a prompt that is not part of the training data to the model
- the model can generate a result that you desire



Few-Shot Prompting

- give some examples in the prompt
- few-shot prompt enables the model to learn without training
- model generates response based on format of given examples

Prompt:

```
The odd numbers in this group add up to an even number: 4, 8, 9, 15, 12, 2, 1.  
A: The answer is False.  
The odd numbers in this group add up to an even number: 17, 10, 19, 4, 8, 12, 24.  
A: The answer is True.  
The odd numbers in this group add up to an even number: 16, 11, 14, 4, 8, 13, 24.  
A: The answer is True.  
The odd numbers in this group add up to an even number: 17, 9, 10, 12, 13, 4, 2.  
A: 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:

```
The answer is True.
```

Chain-of-thought Prompting

- enable complex reasoning capabilities through intermediate reasoning steps

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. ✅

Chains, Agents, Agent Executors

COMMON APPLICATION LOGIC

Model I/O

MODEL

OUTPUT PARSER

PROMPT

EXAMPLE SELECTOR

Retrieval

DOCUMENT LOADER

RETRIEVER

EMBEDDING MODEL

VECTOR STORE

TEXT SPLITTER

LCEL

PARALLELIZATION

FALLBACKS

TRACING

BATCHING

STREAMING

ASYNC

COMPOSITION

What is LangChain?

- a framework for developing applications powered by language models
- connect a language model to sources of context (prompt instructions, , few shot examples)
- simplifies the process of creating generative AI application interfaces

LangChain prompt templates

- LangChain provides a set of default prompt templates
- Create custom prompt is also available
- To create a custom string prompt template, there are two requirements:
 - It has an input variables attribute
 - It defines a format method that takes in keyword arguments corresponding to the expected input variables and returns the formatted prompt

Custom Prompt

```
from langchain.prompts import StringPromptTemplate
from pydantic import BaseModel, validator

PROMPT = """
Given the function name and source code, generate an English language explanation of the function.
Function Name: {function_name}
Source Code:
{source_code}
Explanation:
"""

class FunctionExplainerPromptTemplate(StringPromptTemplate, BaseModel):
    """A custom prompt template that takes in the function name as input, and formats the prompt template to provide the function name and source code as input variables.

    @validator("input_variables")
    def validate_input_variables(cls, v):
        """Validate that the input variables are correct."""
        if len(v) != 1 or "function_name" not in v:
            raise ValueError("function_name must be the only input_variable.")
        return v

    def format(self, **kwargs) -> str:
        # Get the source code of the function
        source_code = get_source_code(kwargs["function_name"])

        # Generate the prompt to be sent to the language model
        prompt = PROMPT.format(
            function_name=kwargs["function_name"].__name__, source_code=source_code
        )
        return prompt

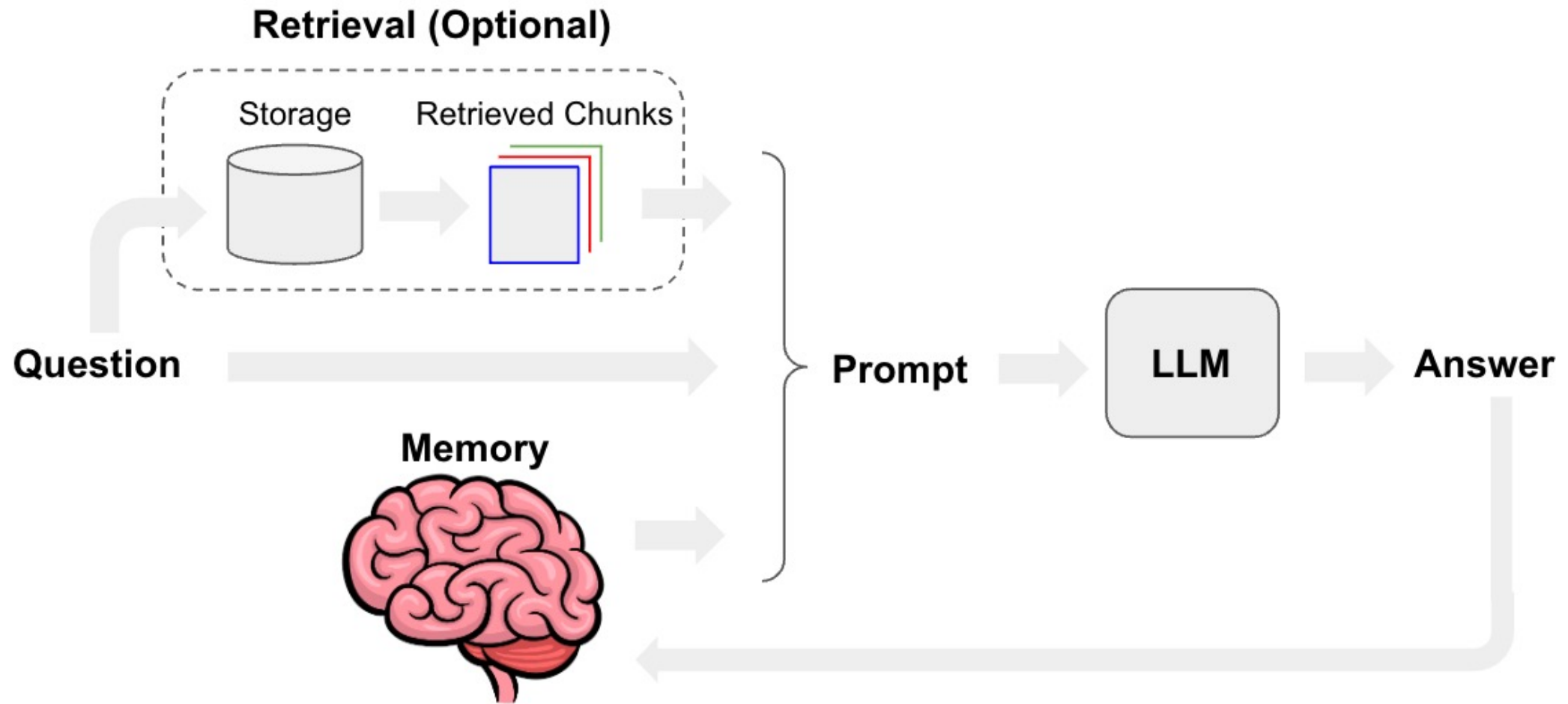
    def _prompt_type(self):
        return "function-explainer"
```

Template Prompt

```
from langchain.prompts import PromptTemplate

prompt_template = PromptTemplate.from_template(
    "Tell me a {adjective} joke about {content}."
)
prompt_template.format(adjective="funny", content="chickens")
```

Chatbot



Libraries for building chatbot

- `!pip install -Uqqq pip --progress-bar off`
- `!pip install -qqq torch==2.0.1 --progress-bar off`
- `!pip install -qqq transformers==4.33.2 --progress-bar off`
- `!pip install -qqq langchain==0.0.299 --progress-bar off`
- `!pip install -qqq chromadb==0.4.10 --progress-bar off`
- `!pip install -qqq xformers==0.0.21 --progress-bar off`
- `!pip install -qqq sentence_transformers==2.2.2 --progress-bar off`
- `!pip install -qqq tokenizers==0.14.0 --progress-bar off`
- `!pip install -qqq optimum==1.13.1 --progress-bar off`
- `!pip install -qqq auto-gptq==0.4.2 --extra-index-url https://huggingface.github.io/autogptq-index/whl/cu118/ --progress-bar off`
- `!pip install -qqq unstructured==0.10.16 --progress-bar off`

- Download model from your HuggingFace hub

```
import torch
from langchain import HuggingFacePipeline
from transformers import AutoModelForCausalLM, AutoTokenizer, GenerationConfig, pipeline

MODEL_NAME = "JackyYu/Llama-2-7b-chat-finetune"

tokenizer = AutoTokenizer.from_pretrained(MODEL_NAME, use_fast=True)

model = AutoModelForCausalLM.from_pretrained(
    MODEL_NAME, torch_dtype=torch.float16, trust_remote_code=True, device_map="cuda"
)

Loading checkpoint shards: 100% ██████████ 2/2 [01:01<00:00, 28.29s/it]

(...)tune/resolve/main/generation_config.json: 100% ██████████ 174/174 [00:00<00:00, 11.9kB/s]

/usr/local/lib/python3.10/dist-packages/transformers/generation/configuration_utils.py:362: UserWarning: `do_sample` is set to `False`. However, `temperature` is set to `0.9` -- this flag is
warnings.warn(
/usr/local/lib/python3.10/dist-packages/transformers/generation/configuration_utils.py:367: UserWarning: `do_sample` is set to `False`. However, `top_p` is set to `0.6` -- this flag is
warnings.warn(
```

- Change the configuration of model and build the llm pipeline

```
generation_config = GenerationConfig.from_pretrained(MODEL_NAME)
generation_config.max_new_tokens = 1024
generation_config.temperature = 0.0001
generation_config.top_p = 0.95
generation_config.do_sample = True
generation_config.repetition_penalty = 1.15

text_pipeline = pipeline(
    "text-generation",
    model=model,
    tokenizer=tokenizer,
    generation_config=generation_config,
)

llm = HuggingFacePipeline(pipeline=text_pipeline, model_kwargs={"temperature": 0})
```

- Test llm pipeline

```
result = llm(  
    "Explain the difference between ChatGPT and open source LLMs in a couple of lines."  
)  
print(result)
```

ChatGPT is an AI-powered chatbot developed by Meta that uses natural language processing (NLP) to generate human-like responses to user queries, while open source LLMs are machine learning models that are trained on large amounts of text data and can generate human-like text.

- Example of prompt template

```
✓ 0s ▶ from langchain import PromptTemplate  
  
template = """  
<s>[INST] <<SYS>>  
Act as a Machine Learning engineer who is teaching high school students.  
<</SYS>>  
  
{text} [/INST]  
"""  
  
prompt = PromptTemplate(  
    input_variables=["text"],  
    template=template,  
)
```

Functions you need for chatbot

```
from langchain.prompts.chat import (  
    ChatPromptTemplate,  
    HumanMessagePromptTemplate,  
    SystemMessagePromptTemplate,  
)  
  
from langchain.schema import AIMessage, HumanMessage
```

- Chatbot looks like:

```
AI: The most powerful AI model is the Transformer, which was introduced in 2017 by Google researchers. It has been used for a wide range of applications such as language translation, t  
Human: Can you explain it to me like I'm five years old?  
AI: Sure thing! Imagine you have a big box full of toys. You want to find one specific toy inside the box, but you don't know where it is or what color it is. That's kinda like how com  
Human: Wow, thank you for explaining it to me like that!  
AI: Of course! I am always here to help you learn about artificial intelligence. Do you have any questions about anything else related to this topic?
```


References

- <https://www.promptingguide.ai/techniques/fewshot>
- <https://lh6.googleusercontent.com/aKjm1ycrNyqvETuDGvbNS0M1aINN-m3ZHs3RVz--INy-zxwbycf6MuB-P4JDb8yDx-urlJ1LbUkJpZQrLH4LGHeN0cy2o0IDsl9qqPMMYQi5Wg7tRYXVhGdZ-z1xRh-Ln1R9fVDHxsEmqMUfa4eSjCo>
- <https://zapier.com/blog/ai-hallucinations/>
- <https://machinelearningmastery.com/what-are-zero-shot-prompting-and-few-shot-prompting/>
- <https://www.promptingguide.ai/techniques/cot>
- <https://www.langchain.com/>