Learn and Grow

Generative AI LangChain

ASHIMA MALIK

ashimamalik58@gmail.com

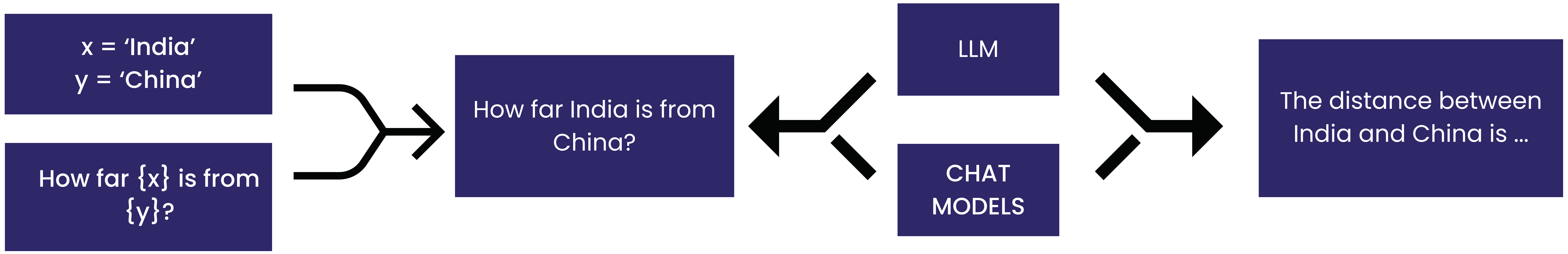
Model I/O

In this part of the course, we'll start learning about LangChain. We'll learn how to ask questions to models and understand their answers.

LangChain mainly sends messages to language models and deals with what they say back.

LangChain gives the building blocks to interface with any language model.

MODEL I/O


x = 'India'
y = 'China'
How far {x} is from {y}?
How far India is from China?
LLM
CHAT MODELS
The distance between India and China is ...

INPUT (PROMPTS)

instructions or input that set the context and parameters for the model's operation

MODELS

LLM takes a text string as input \
and
returns a text string

Chat model that takes a list of messages
as input
and returns a message

OUTPUT (PARSER)

LLM response to the Prompt

MODELS

ASHIMA MALIK

Large Language Models

LLMs

LLMs are powerful AI models trained on massive amounts of text data. They can generate text, translate languages, write different kinds of creative content, and answer your questions in an informative way.

They can handle various tasks like summarization, question answering, and text generation based on the prompt or input provided.

CHAT Models

Chat models are specialized for conversational interactions. They take a sequence of messages (like a chat history) as input and generate a response that fits the conversation context.

They are designed to understand the flow of conversation, consider previous messages, and respond accordingly. They are ideal for building chatbots or virtual assistants that can engage in natural and engaging dialogues.

?

When to Use Which?

ASHIMA MALIK

MESSAGES

ASHIMA MALIK

Messages are fundamental units that represent the information exchanged within a conversation or workflow.

ChatModels take a list of messages as input and return a message.

SystemMessage

This type is used for internal communication within the Langchain workflow. It doesn't directly appear in the conversation with the user but provides instructions or configuration details for the agent's behavior.

HumanMessage

This type represents a message originating from the human user interacting with your conversational AI agent. It typically contains the user's query, request, or response within the conversation.

AIMessage

This type represents a message generated by the AI agent itself. It can be a response to a user's query, the output from a Langchain tool, or any information the agent provides during the interaction.

You are a witty and intelligent teenager.

Tell me about stars.

Ah, stars, those sparkling celestial beauties!
They're like the glitter in the universe's cosmic confetti.

ASHIMA MALIK

PROMPTS

ASHIMA MALIK

Prompts

Prompts are crucial elements that guide Large Language Models (LLMs) and other tools towards generating the desired output. They act as instructions or inputs that set the context and parameters for the model's operation.

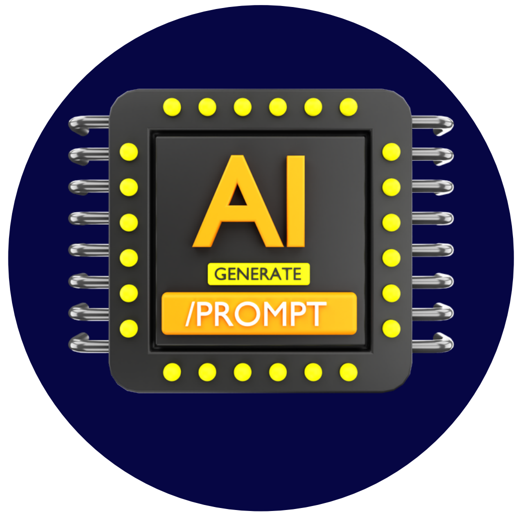
The objects that take user input and transform it into the final string or messages are known as "Prompt Templates".

PromptTemplate

create a template for a string prompt

```
from langchain.prompts import PromptTemplate

prompt = PromptTemplate.from_template("Suggest me a good company name for {agency}?")
prompt.format(agency="digital marketinng service agency")
```



ChatPromptTemplate

The prompt to chat models is a list of chat messages

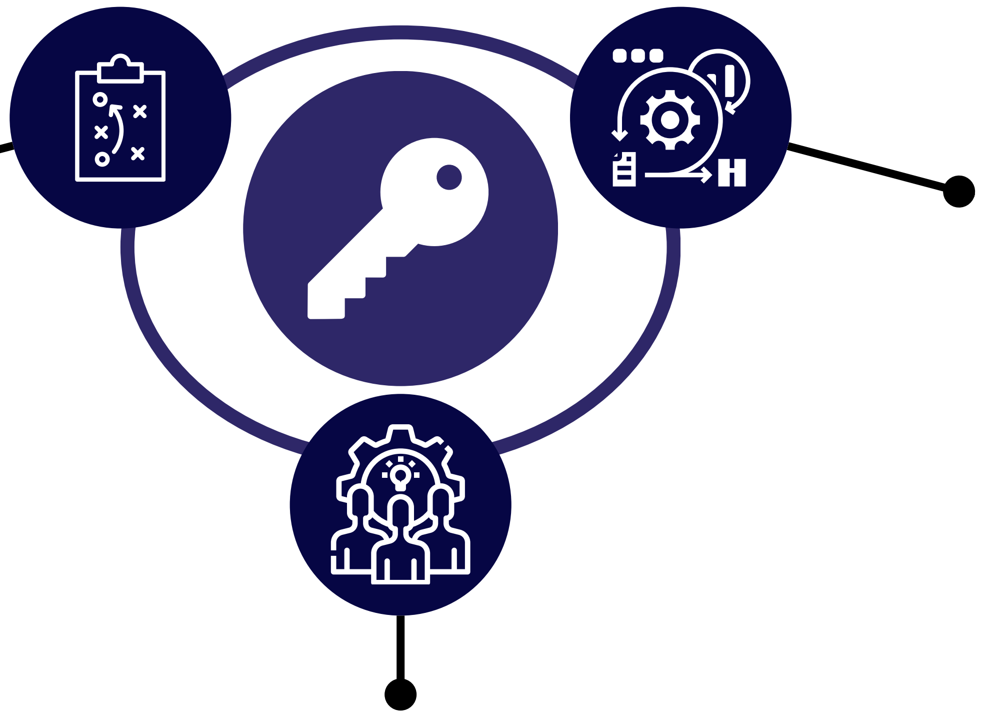
```
from langchain_core.prompts import ChatPromptTemplate

chat_template = ChatPromptTemplate.from_messages(
    [
        ("system", "You are a helpful AI assistant bot. Your name is {name}."),
        ("human", "Hi, how are you?"),
        ("ai", "I am doing good"),
        ("human", "{user_input}"),
    ]
)

messages = chat_template.format_messages(name="Lily", user_input="What is your name?")
```

ASHIMA MALIK

BENEFITS OF GOOD PROMPTS



Improved Output Quality: Clear and well-designed prompts lead to more accurate, relevant, and informative responses from LLMs and tools.

Flexibility: Prompts allow you to adapt your workflows to handle various tasks and user interactions effectively.

Enhanced Control: By carefully crafting prompts, you can exercise greater control over the direction and style of the generated text.

SELECTORS

- Imagine you have a large number of examples. Then you need to select which one to include in the prompt.
- Selectors act like filters, analyzing the prompts and choosing the one that best fits your needs.

Similarity	This selects examples based on cosine similarity to the inputs.
MMR (Maximal Marginal Relevance)	This approach balances similarity and diversity. It selects examples based on a combination of which examples have the most similarity to the inputs, but also maintain the diversity by penalizing them for their closeness.
Length	This selector chooses examples based on a desired length (e.g., selecting the shortest or longest response).
N-gram Overlap	This method focuses on the overlap of n-grams (sequences of n words) between the examples and input. The selector picks the response with the highest n-gram overlap.

Improved Output Quality

Reduced Manual Work

Fine-Tuning Output

Few-shot prompt templates

“

- In FewShotPromptTemplate, we provide the examples to the LLM model. It will tell the LLM about the context and how we want the output to look like.

```
# Examples of a pretend task of creating antonyms.
examples = [
{"input": "parrot", "output": "bird"},
{"input": "cow", "output": "animal"},
{"input": "snake", "output": "reptile"},
{"input": "frog", "output": "amphibian"},
{"input": "dog", "output": "mammal"},
]

example_prompt = PromptTemplate(
input_variables=["input", "output"],
template="Input: {input}\nOutput: {output}",
```

PARSERS

ASHIMA MALIK

OUTPUT PARSER

In LangChain, output parsers are specialized tools that take the raw text response generated by a large language model (LLM) and transform it into a more structured and usable format.

Structured Data

Not all tasks require raw text. Sometimes you need the information in a specific format, like JSON, a Python data class, or a Pandas DataFrame. Output parsers can convert the LLM's response into these formats for easier manipulation and use in your workflows.

Extracting Key Information

For tasks like summarization or question answering, you might only be interested in the most important points. Output parsers can help extract these key details from the LLM's potentially lengthy response.

Data Cleaning & Formatting

LLM responses might contain inconsistencies or irrelevant details. Output parsers can help clean and format the data to make it consistent and usable for further processing.

Types of Output Parsers

Pydantic Parser	Converts the LLM response into a user-defined Pydantic data class, allowing for structured data manipulation.
JSON Parser	Transforms the response into a JSON object for easy integration with other tools and workflows.
List Parser	Parses the output into a list of strings, useful for scenarios where the LLM generates multiple items.
Datetime Parser	Extracts and formats dates or times mentioned in the response.
Structured Parser (Basic)	This is a simpler option that parses the response into a dictionary structure, suitable for less complex data.

output parsers are a powerful tool in LangChain that bridges the gap between the raw LLM output and your desired data format. They allow you to effectively leverage the capabilities of LLMs for various tasks by structuring and extracting the information you need.