The LangChain ecosystem

DEVELOPING LLM APPLICATIONS WITH LANGCHAIN

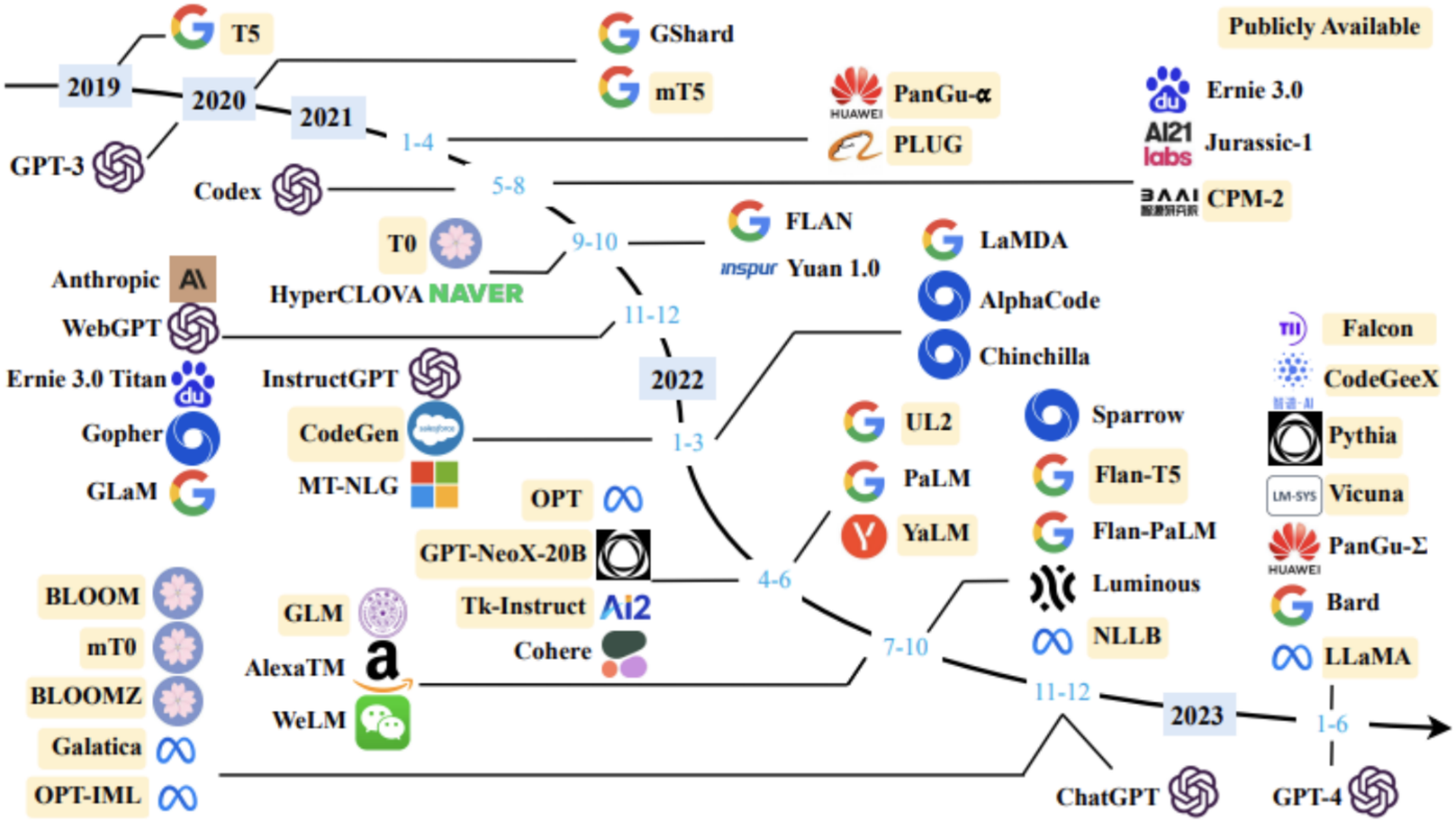
Jonathan Bennion
AI Engineer & LangChain Contributor

datacamp

Meet your instructor...

- Jonathan Bennion, **AI Engineer**
- ML & AI at Facebook, Google, Amazon, Disney, EA
- Created *Logical Fallacy chain* in LangChain
- Contributor to **DeepEval**

The state of Large Language Models (LLMs)

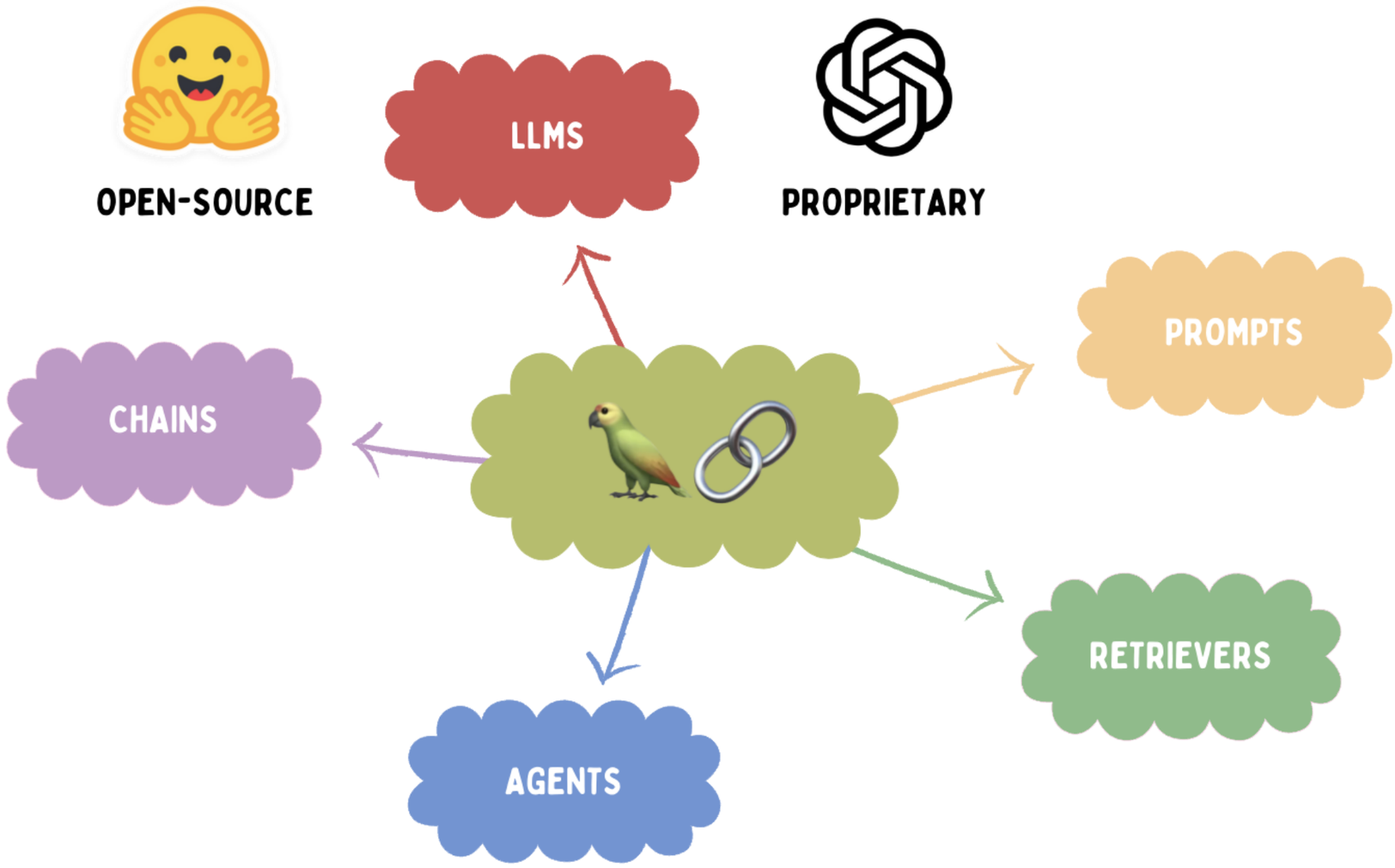


[1] https://arxiv.org/pdf/2303.18223.pdf

What is LangChain?

- **Open-source** framework for connecting:
  - Large Language Models (LLMs)
  - Data sources
  - Other functionality under a **unified syntax**
- Allows for scalability
- Contains modular components
- Supports **Python** and **JavaScript**

Core components of LangChain



Hugging Face

- **Open-source** repository of models, datasets, and tools

Creating a Hugging Face API key:

1. Sign up for a Hugging Face account
2. Navigate to `https://huggingface.co/settings/tokens`
3. Select New token and copy the key

Hugging Face (*Falcon-7b*):

```python
from langchain_huggingface import HuggingFaceEndpoint

llm = HuggingFaceEndpoint(
    repo_id='tiiuae/falcon-7b-instruct',
    huggingfacehub_api_token=huggingfacehub_api_token
)

question = 'Can you still have fun'
output = llm.invoke(question)

print(output)
```

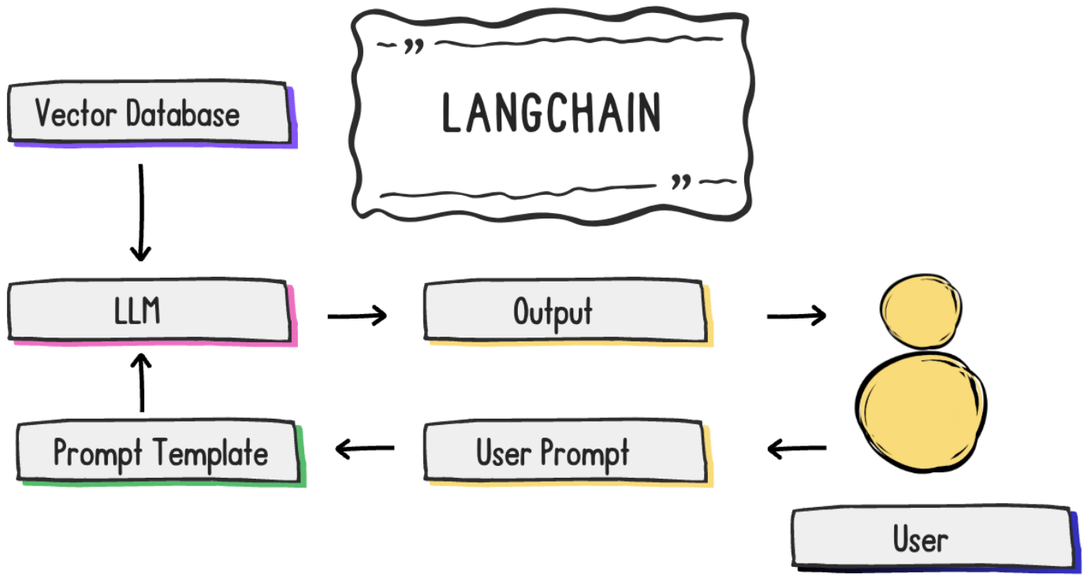
```
 in the rain?
Yes, you can still have fun in the
rain! There are plenty of
```

OpenAI (`gpt-3.5-turbo-instruct`):

```python
from langchain_openai import OpenAI

llm = OpenAI(
    openai_api_key=openai_api_key
)


question = 'Can you still have fun'
output = llm.invoke(question)

print(output)
```

```
 without spending a lot of money?

Yes, you can still have fun without
spending a lot of money. You could do
activities like hiking, biking, playing
sports, going to the beach, camping...
```

Real-world usage



Examples:

- Natural language conversations with documents
- Automate tasks
- Data analysis

Note: course uses `langchain==0.2.3`

Let's practice!

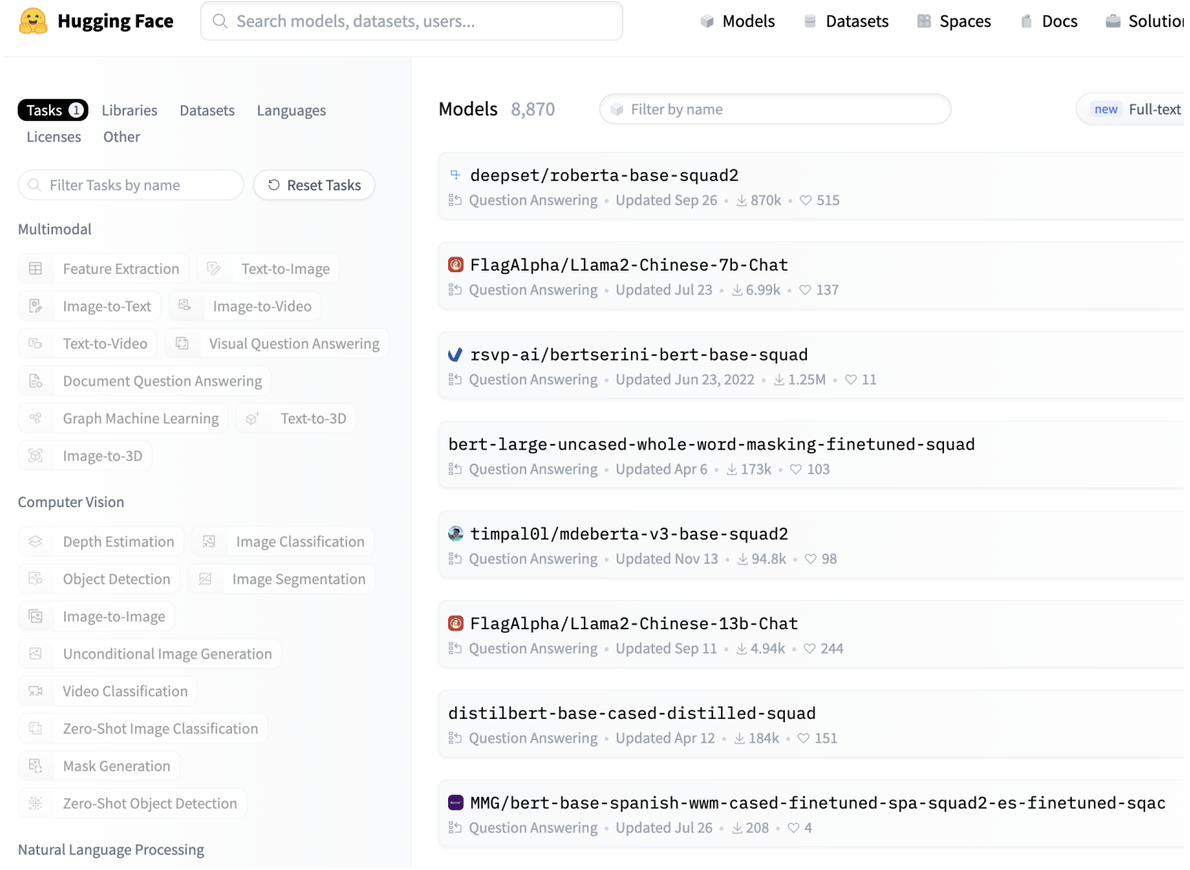
DEVELOPING LLM APPLICATIONS WITH LANGCHAIN

Prompting strategies for chatbots

DEVELOPING LLM APPLICATIONS WITH LANGCHAIN

Jonathan Bennion
AI Engineer & LangChain Contributor

Finding the right model



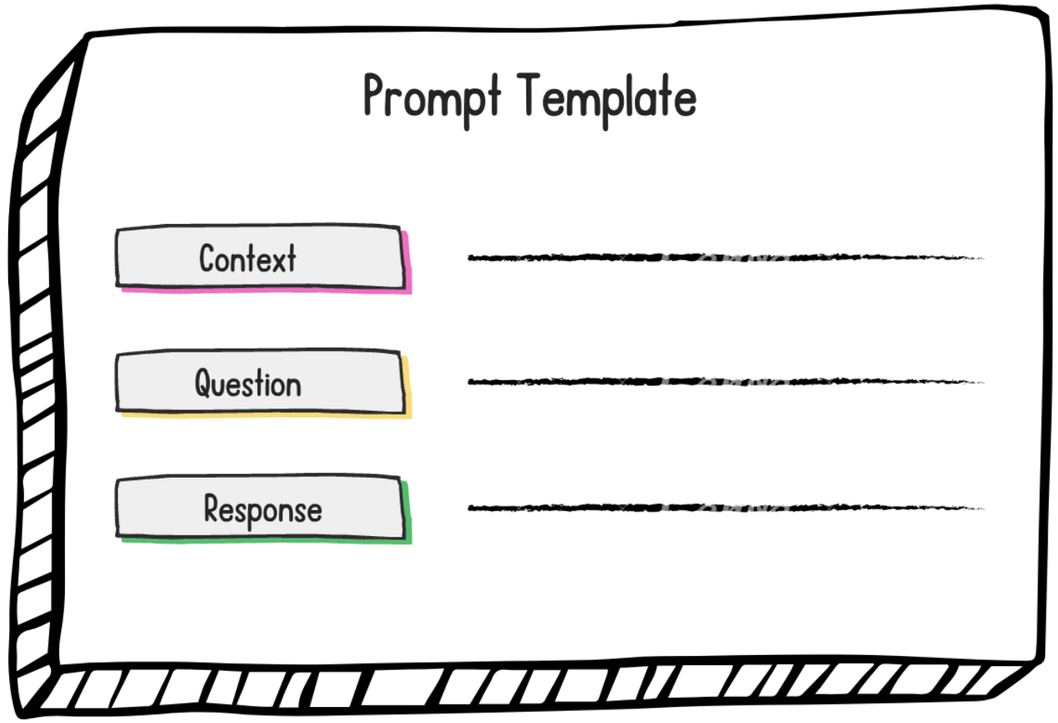
Hugging Face is a searchable hub for chat models

- *Fine-tuned models* for more domain-specific use cases

[1] https://huggingface.co/models?pipeline_tag=question-answering&sort=trending

Prompt templates

- **Recipes** for generating prompts
- *Flexible* and *modular*
- Can contain: instructions, examples, and additional context



Prompt templates

```
from langchain_core.prompts import PromptTemplate

template = "You are an artificial intelligence assistant, answer the question. {question}"
prompt_template = PromptTemplate(template=template, input_variables=["question"])

print(prompt_template.invoke({"question": "What is LangChain?"}))
```

```
text='You are an artificial intelligence assistant, answer the question. What is LangChain?'
```

Integrating PromptTemplate with LLMs

```
from langchain_community.llms import HuggingFaceHub

llm = HuggingFaceHub(repo_id='tiiuae/falcon-7b-instruct', huggingfacehub_api_token=huggingfacehub_api_token)
llm_chain = prompt_template | llm

question = "What is LangChain?"
print(llm_chain.invoke({"question": question}))
```

```
LangChain is an artificial intelligence language model that uses a neural network to generate human-like text
```

- LangChain Expression Language (**LCEL**)
- **Chain:** connect calls to different components

Chat models

```
from langchain_core.prompts import ChatPromptTemplate


prompt_template = ChatPromptTemplate.from_messages(
    [
        ("system", "You are soto zen master Roshi."),
        ("human", "What is the essence of Zen?"),
        ("ai", "When you are hungry, eat. When you are tired, sleep.")
        ("human", "Respond to the question: {question}")
    ]
)
```

Integrating ChatPromptTemplate

```
from langchain_openai import ChatOpenAI

llm = ChatOpenAI(openai_api_key=openai_api_key)

llm_chain = prompt_template | llm
question='What is the sound of one hand clapping?'

response = llm_chain.invoke({"question": question})
print(response.content)
```

```
The sound of one hand clapping is not something that can be easily explained or
understood through words alone. It is a question that has been pondered by Zen
practitioners for centuries, and its purpose is to provoke a deeper inquiry into
the nature of reality and the self. In Zen practice, we often engage...
```

Let's practice!

DEVELOPING LLM APPLICATIONS WITH LANGCHAIN

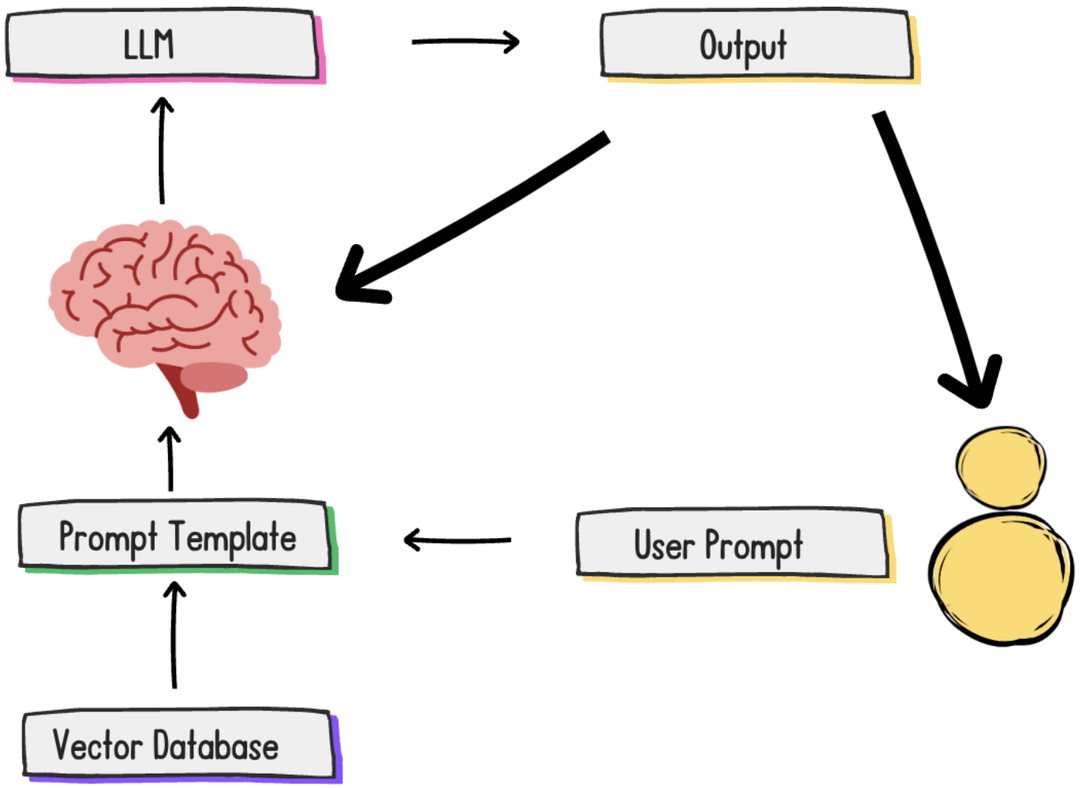
Managing chat model memory

DEVELOPING LLM APPLICATIONS WITH LANGCHAIN

Jonathan Bennion
AI Engineer & LangChain Contributor

datacamp

In-conversation memory



- Follow-up questions
- Response iteration and expansion
- Personalization

Context window: amount of input text a model can consider at once

- `ChatMessageHistory`
- `ConversationBufferMemory`
- `ConversationSummaryMemory`

ChatMessageHistory

- Stores **full message history**

```
from langchain.memory import ChatMessageHistory
from langchain_openai import ChatOpenAI
llm = ChatOpenAI(openai_api_key=openai_api_key)

history = ChatMessageHistory()
history.add_ai_message("Hi! Ask me anything about LangChain.")
history.add_user_message("Describe a metaphor for learning LangChain in one sentence.")

response = llm.invoke(history.messages)
print(response.content)
```

```
Learning LangChain is like unraveling a complex tapestry of interconnected languages, each thread revealing a new layer of linguistic understanding.
```

ChatMessageHistory

```
history.add_user_message("Summarize the preceding sentence in fewer words")

response = llm.invoke(history.messages)
print(response.content)
```

```
LangChain is a linguistic bridge that connects learners to a world of new languages.
```

ConversationBufferMemory

```
from langchain.memory import ConversationBufferMemory
from langchain.chains import ConversationChain

llm = ChatOpenAI(openai_api_key=openai_api_key)
memory = ConversationBufferMemory(size=4)

buffer_chain = ConversationChain(llm=llm, memory=memory)
```

ConversationBufferMemory

```
buffer_chain.invoke("Describe a language model in one sentence")
buffer_chain.invoke("Describe it again using less words")
buffer_chain.invoke("Describe it again fewer words but at least one word")
buffer_chain.invoke("What did I first ask you? I forgot.")
```

```
{'input': 'What did I first ask you? I forgot.',
 'history': 'Human: Describe a language model in one sentence\nAI:  A language model is a ...',
 'response': ' You asked me to describe a language model in one sentence.'}
```

ConversationSummaryMemory

- *Summarizes* the history to condense information

```
from langchain.memory import ConversationSummaryMemory

llm = ChatOpenAI(openai_api_key=openai_api_key)

memory = ConversationSummaryMemory(llm=ChatOpenAI(openai_api_key=openai_api_key))

summary_chain = ConversationChain(llm=llm, memory=memory, verbose=True)
```

ConversationSummaryMemory

```
summary_chain.invoke("Please summarize the future in 2 sentences.")
summary_chain.invoke("Why?")
summary_chain.invoke("What will I need to shape this?")
```

```
> Entering new ConversationChain chain...
...

> Finished chain.
{'input': 'What will I need to shape this?', 'history': 'The human asks the AI to summarize the future in
2 sentences. The AI predicts that rapid technological advancements will lead to increased automation and
interconnectedness, posing challenges related to ethics, privacy, and the impact of artificial intelligence
on jobs and daily life. The AI emphasizes that these advancements will shape the future, highlighting the
need to address ethical concerns and adapt to the changing landscape of technology.', 'response': 'To shape
the future, you will need to stay informed about emerging technologies, engage in discussions about ethical
considerations, and be open to adapting to new ways of living and working in a tech-driven world. It will
also be important to prioritize collaboration, innovation, and continuous learning to navigate the challenges
and opportunities that come with the evolving future.'}
```

Let's practice!

DEVELOPING LLM APPLICATIONS WITH LANGCHAIN