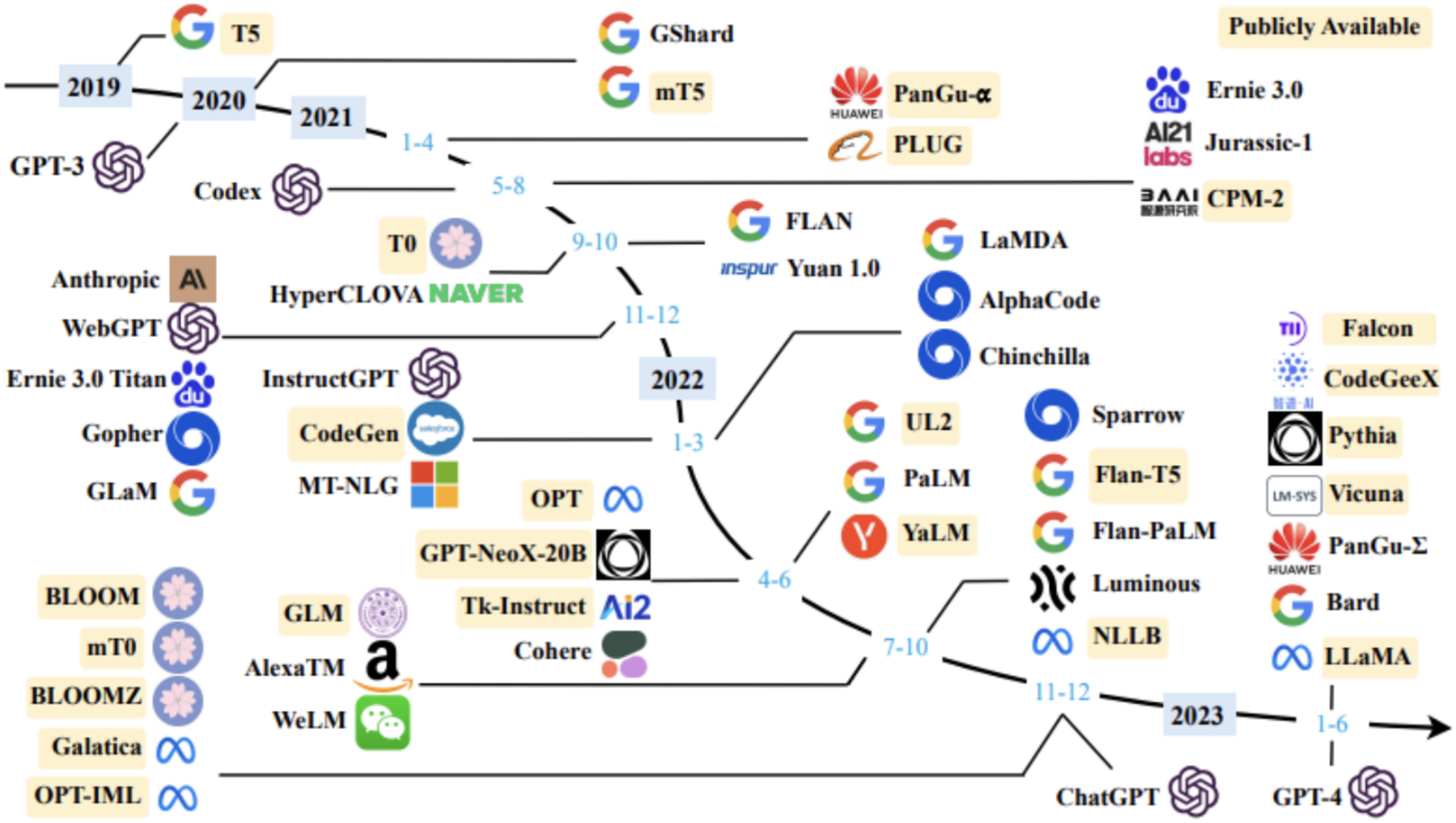
The LangChain ecosystem

DEVELOPING LLM APPLICATIONS WITH LANGCHAIN

Jonathan Bennion
AI Engineer & LangChain Contributor

datacamp

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

What will this course cover?

Overview:

- Key functionality of the `langchain` library
- **Chains** and **Tools** for optimizing LLM output
- Troubleshooting and evaluation techniques

Core components of LangChain

Open-Source Models

Closed-Source Models

Prompts

Parsers and Indexers

Chains and Agents

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 (using Falcon 7b):

```
from langchain_huggingface import HuggingFaceEndpoint

llm = HuggingFaceEndpoint(repo_id='tiiuae/falcon-7b-instru
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 (using default OpenAI model):

```
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==0.1.17`

[1] Wikimedia Commons license: https://creativecommons.org/licenses/by-sa/3.0/deed.en

Let's practice!

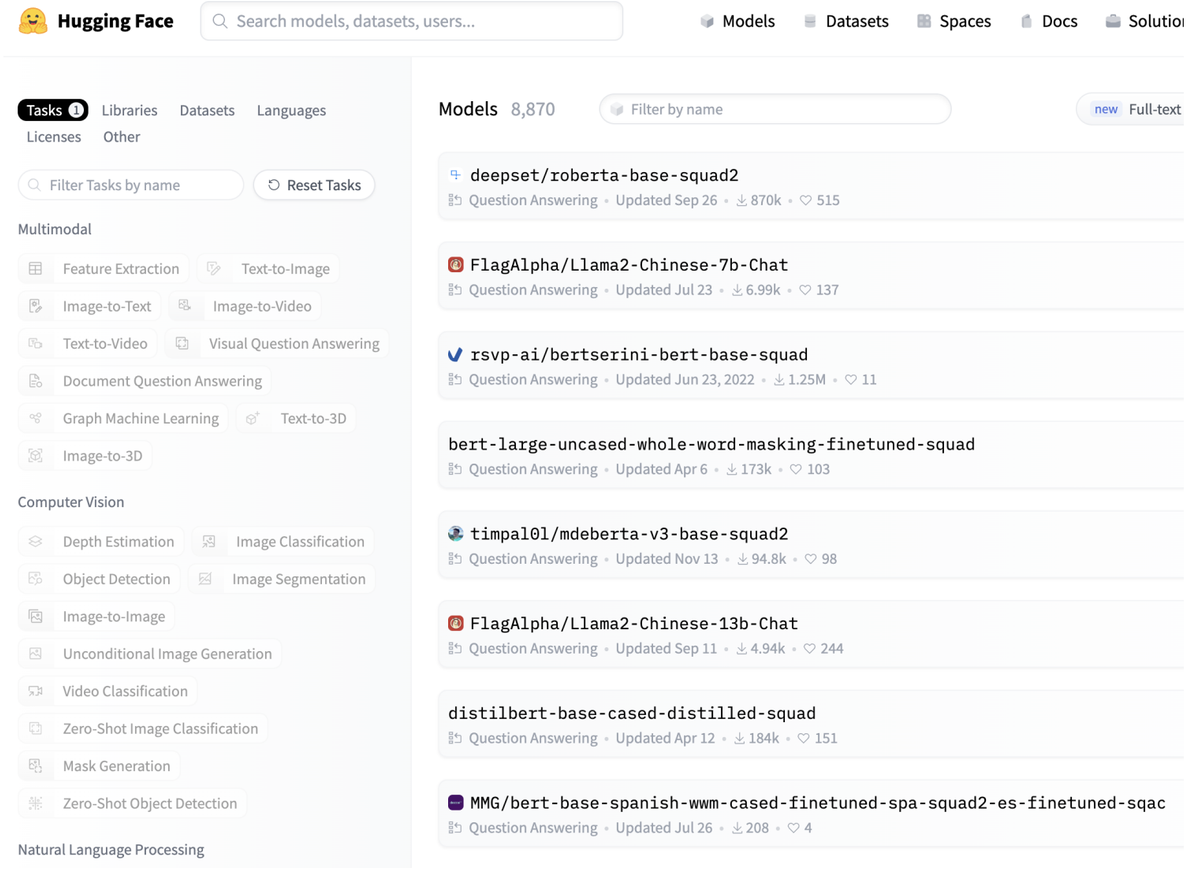
DEVELOPING LLM APPLICATIONS WITH LANGCHAIN

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

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

```
from langchain.prompts import PromptTemplate

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

Integrating PromptTemplate with LLMs: LLMChain

```
from langchain.chains import LLMChain
from langchain_community.llms import HuggingFaceHub

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

question = "What is LangChain?"
print(llm_chain.run(question))
```

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

- LangChain Expression Language (**LCEL**) → **Chapter 3**

Chat models using LangChain with OpenAI

```
from langchain_openai import ChatOpenAI
from langchain_core.prompts import ChatPromptTemplate

llm = ChatOpenAI(temperature=0, openai_api_key=openai_api_key)

prompt_template = ChatPromptTemplate.from_messages(
    [
        ("system", "You are soto zen master Roshi."),
        ("human", "Respond to the question: {question}")
    ]
)

full_prompt = prompt_template.format_messages(question='What is the sound of one hand clapping?')
llm(full_prompt)
```

```
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

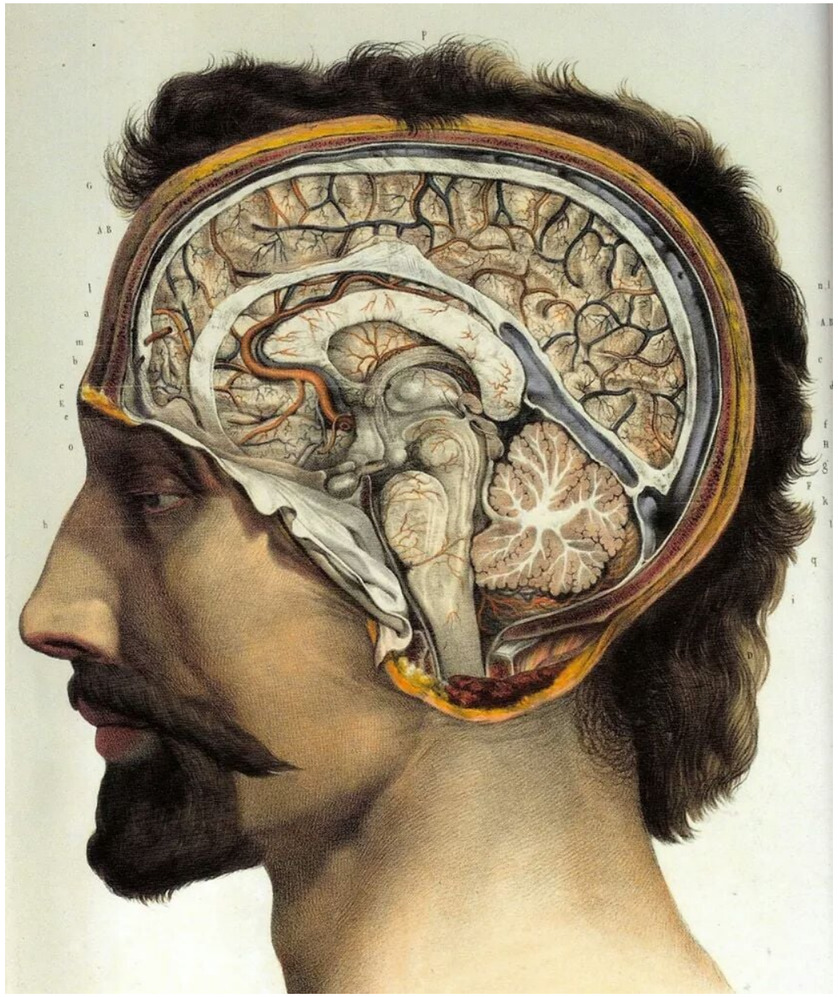
Managing chat model memory

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`

[1] Wikimedia Commons license https://creativecommons.org/licenses/by-sa/3.0/deed.en

ChatMessageHistory

- Stores **full message history**

```
from langchain.memory import ChatMessageHistory
from langchain.chat_models import ChatOpenAI
chat = ChatOpenAI(temperature=0, 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.")
chat(history.messages)
```

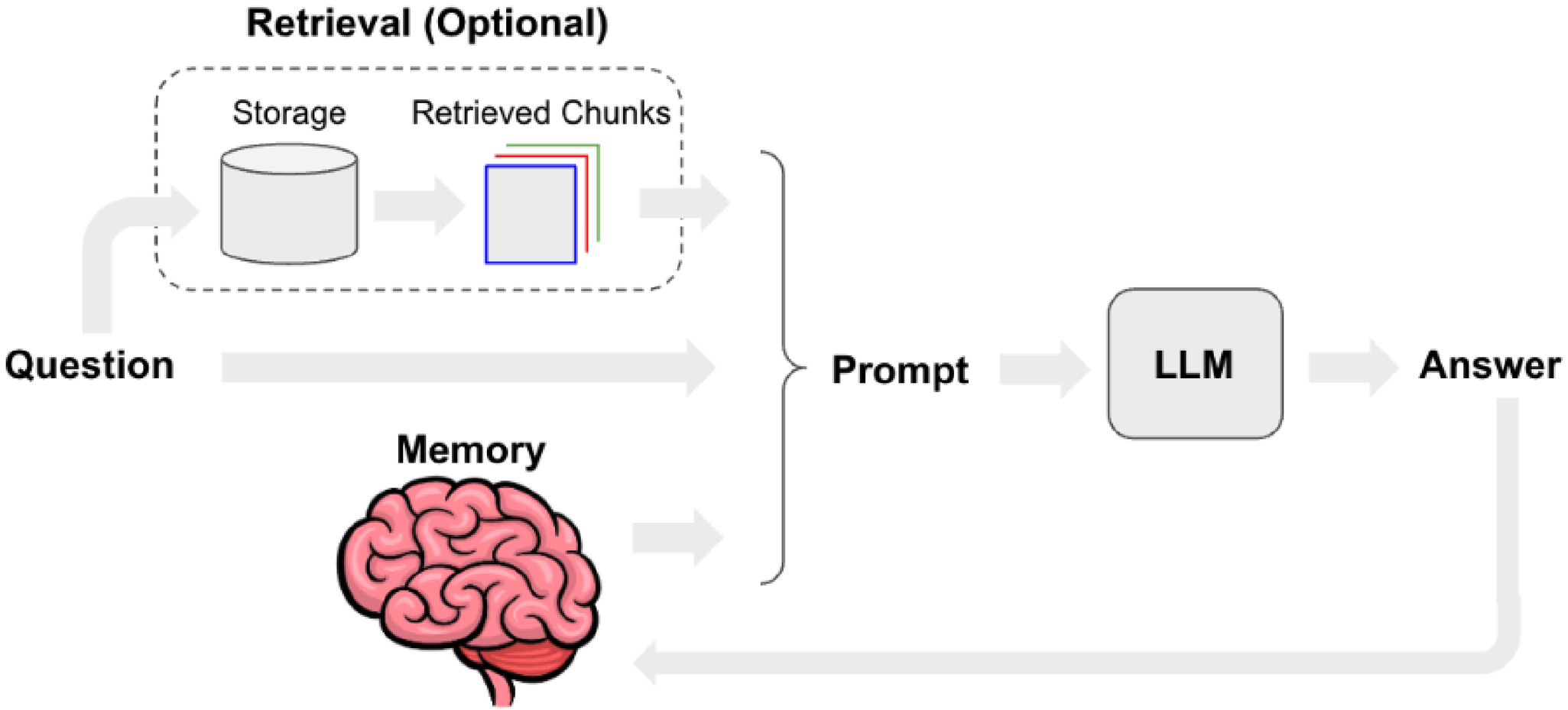
```
Learning LangChain is like unraveling a complex tapestry of interconnected languages, each thread revealing a new layer of linguistic understanding.
```

ChatMessageHistory

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

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

How does chat memory scale with each response?



[1] LangChain

ConversationBufferMemory

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

chat = OpenAI(model_name="gpt-3.5-turbo-instruct", temperature=0, openai_api_key=openai_api_key)
memory = ConversationBufferMemory(size=4)

buffer_chain = ConversationChain(llm=chat, memory=memory, verbose=True)
```

ConversationBufferMemory Output

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

```
> Entering new ConversationChain chain...
Current Conversation:
Human: Describe a language model in one sentence
AI:  A language model is a probabilistic model that assigns a probability to a sequence of words
in a given language.
Human: Describe it again using less words
AI:  A language model assigns probabilities to word sequences.
Human: Describe it again fewer words but at least one word
AI:  Probabilistic.
Human: What did I first ask you? I forgot.
> Finished chain.
```

ConversationSummaryMemory in LangChain

```
from langchain.memory import ConversationSummaryMemory

chat = OpenAI(model_name="gpt-3.5-turbo-instruct", temperature=0, openai_api_key=openai_api_key)

memory = ConversationSummaryMemory(llm=OpenAI(model_name="gpt-3.5-turbo-instruct",
                                              openai_api_key=openai_api_key))

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

ConversationSummaryMemory Output Example

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

```
> Entering new ConversationChain chain...
...
Current conversation:
The human asks the AI to summarize the future
in two sentences. The AI responds that the future is uncertain and full of possibilities,
and that it is up to us to shape it and make it a better place by taking action and making positive
changes. To do this, the AI suggests advocating for policies that promote equity and sustainability,
investing in renewable energy sources, and supporting initiatives that promote economic growth and
opportunity, and to be mindful of the impact of decisions and actions on the environment and society.
Human: What will I need to shape this?
AI:
> Finished chain.
```

Let's practice!

DEVELOPING LLM APPLICATIONS WITH LANGCHAIN