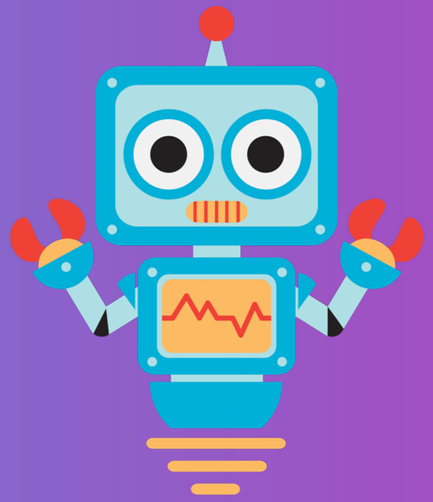
Dr. Abulkarim Albanna

Retrieval/Knowledge-Augmented Generation (RAG)

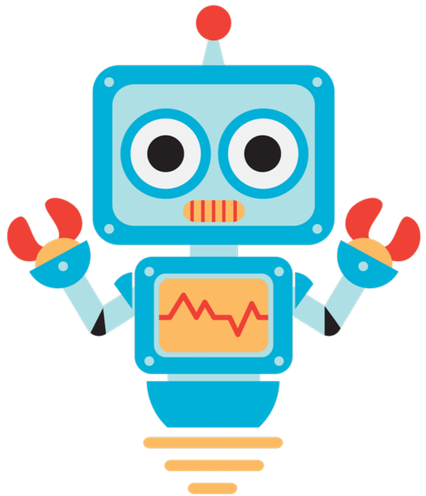


Day 4

January 21th 2024

Content

- Introduction
- Langchain
- Langchain components
- Build Apps with LLMs using Langchain
- Prompts
- LLM Chains

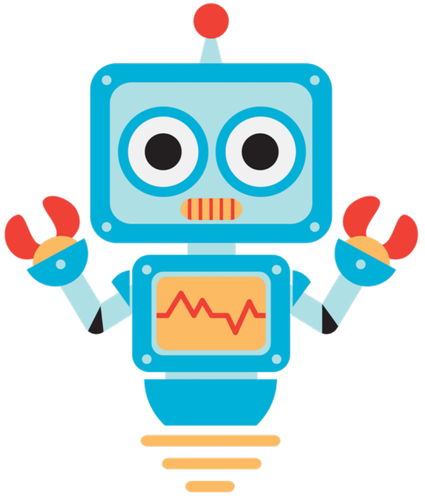


Introduction

RAG

- In industrial settings, companies, particularly startups, seek **cost-effective**, **privacy-compliant**, and **dependable solutions**.

- Recent studies and the release of new chatbots have demonstrated their ability to utilize knowledge and information beyond their initial training data. This approach is known as **Retrieval Augmented Generation** (RAG).

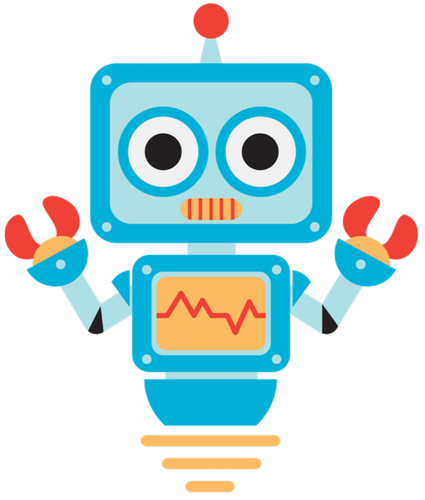


Introduction

RAG

- RAG enables **contextual learning** without expensive fine-tuning, making the use of Large Language Models (LLMs) more cost-efficient.

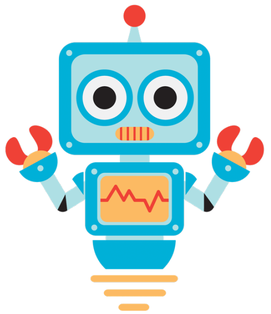
- Companies can use the same model to process and generate responses based on new data while customizing their solution and maintaining relevance.



Introduction

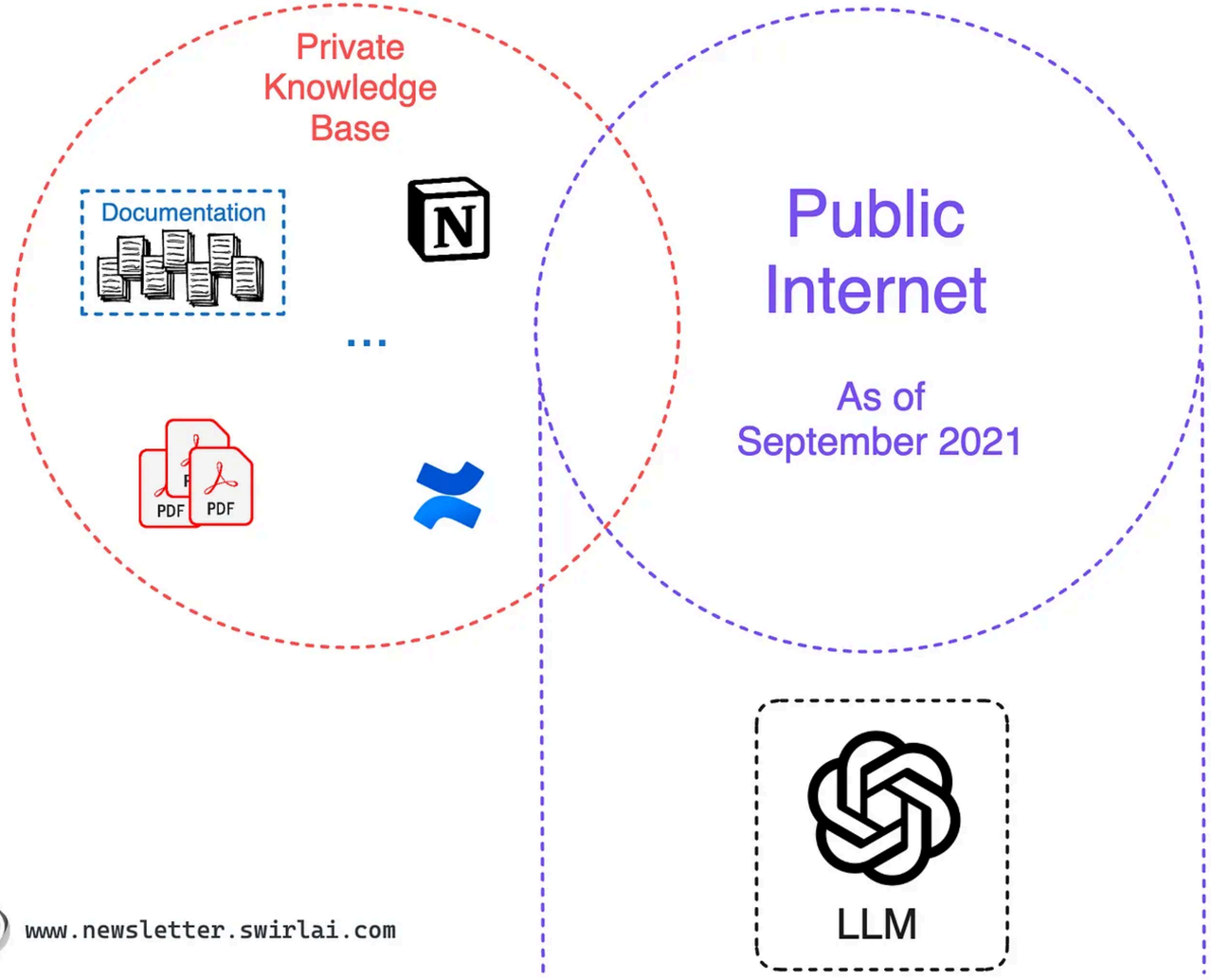
The next approach one could think off

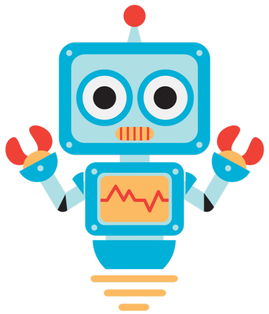
LLM API like ChatGPT



Introduction

Why can't we just use a pure commercial LLM API like ChatGPT to answer the question directly?





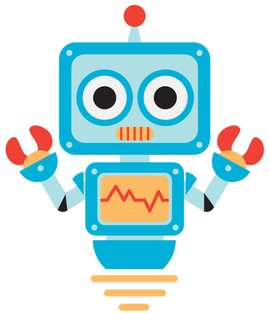
Introduction

Why can't we just use a pure commercial LLM API like ChatGPT to answer the question directly?

- Commercial LLMs have been trained on a large corpus of data available on the internet. The data has more irrelevant context about your question than you might like.

- The **data** contained in your **internal systems** of interest might have not been used when training the LLM - it might be too recent and not available when the model was trained. Or it could be private and not available publicly on the internet

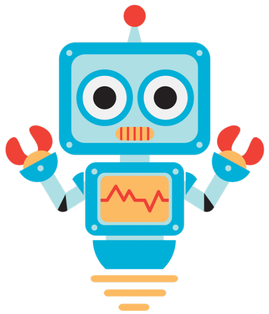
- Currently available LLMs have been shown to produce hallucinations inventing data that is not true



Introduction

The next approach one could think off

- Formulate a **Question/Query** in a form of a meta-prompt.
- Pass the **entire Knowledge Base** from **internal documents** together with the Question via a prompt.
- Via the same meta-prompt, instruct the LLM to only use the previously provided Knowledge Base to answer the question.

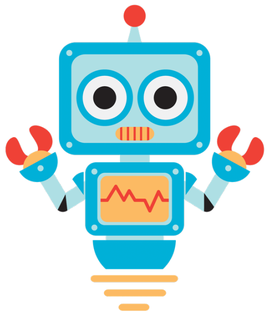


Introduction

What are the problems with this approach?

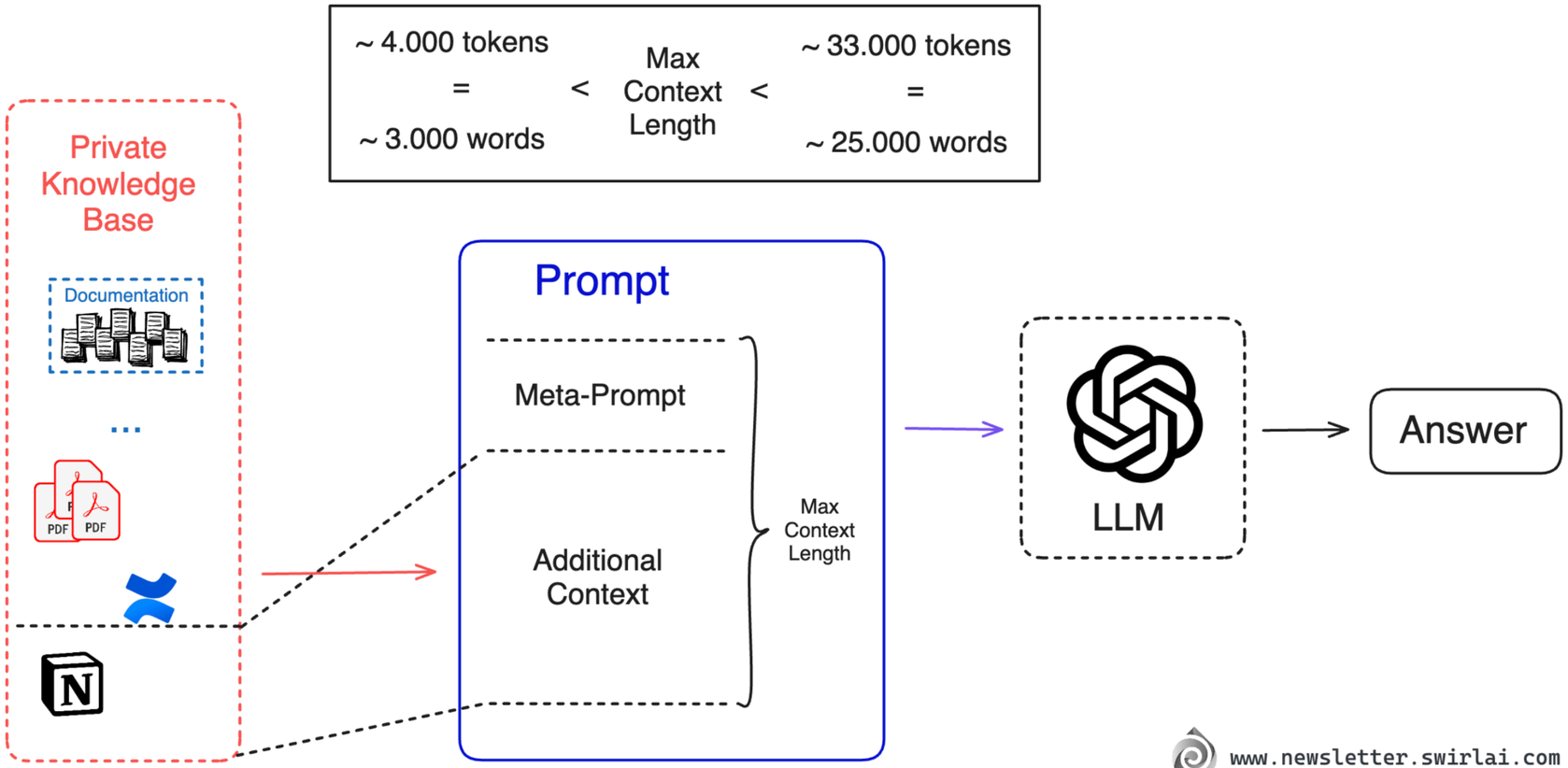
- Token Limit of how much information you can pass to LLM as a prompt. Input context length varies from ~4.000 to ~33.000. 1000 tokens map roughly to 750 words.

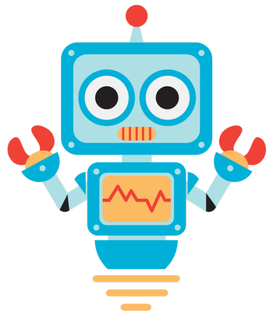
- The maximum prompt length is ~25.000 words.



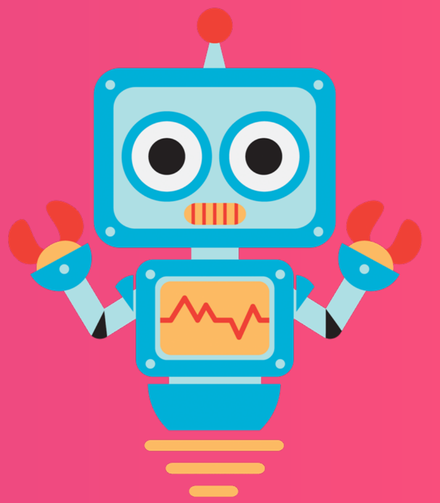
Introduction

Solution





Langchain



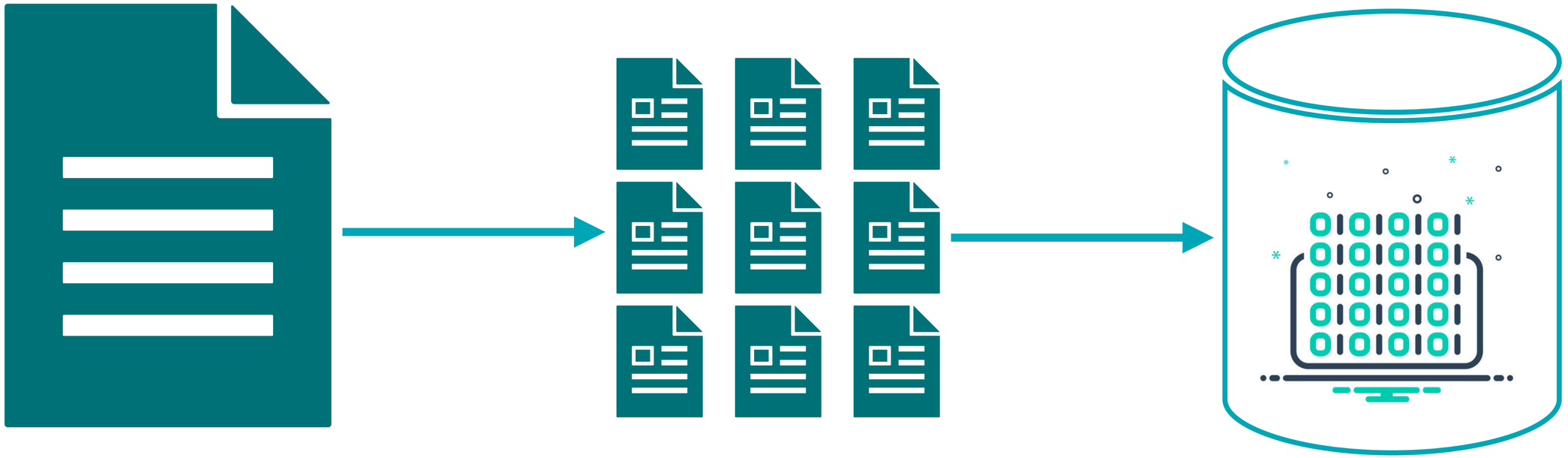
Langchain

Hugging Face

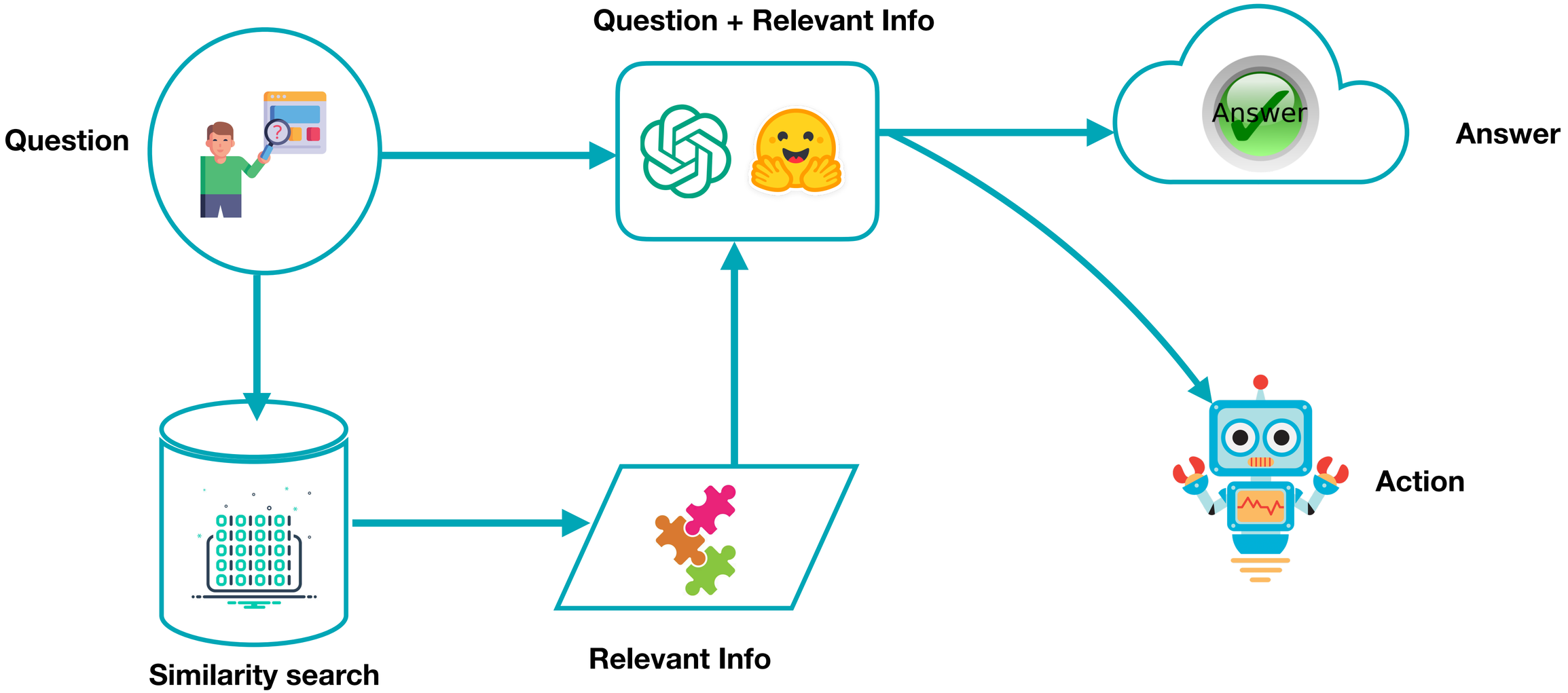
LangChain

documents

Langchain

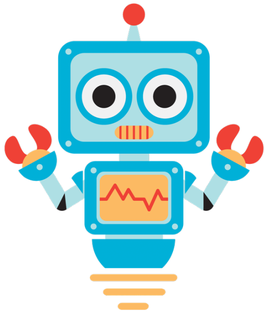


Langchain


Question + Relevant Info
Question
Answer
Answer
Action
Similarity search
Relevant Info

Langchain

🦜🔗 LangChain

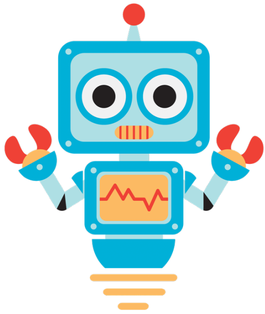


- LangChain is an **open-source** framework to build applications with LLMs.
- It enhances the capabilities of LLMs by providing a standard interface for prompt templates and integration of language models with different APIs and external databases.
- It has an API connection to **~40** of the public **LLMs, chat and embedding models**. It integrates with more than **30** different **tools** and **20** different **vector databases**.

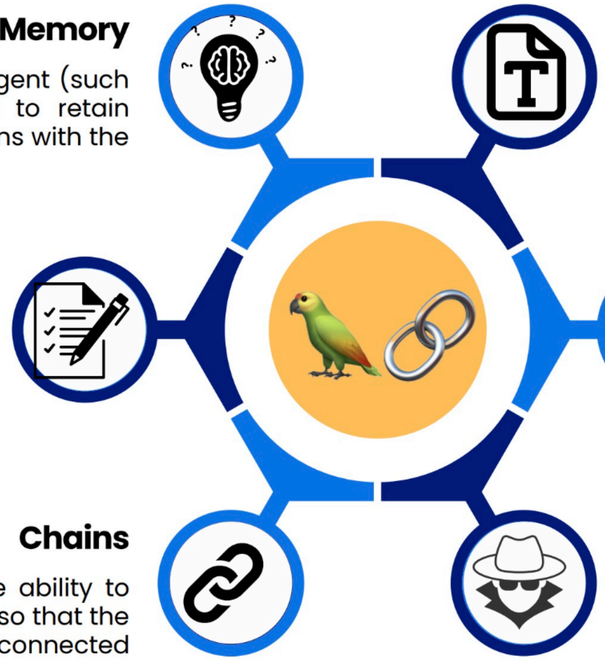
Langchain

LangChain

Components



LangChain Components



Memory

"Memory" refers to the ability of an agent (such as a chatbot or language model) to retain information about previous interactions with the user.

Prompt templates

Prompt templates are a way to generate prompts consistently and reproducibly. The template is a 'text string' that the user can customize in multiple ways. The prompt templates can consist of instructions to LLMs, a few shot examples, etc.

Indexes

Indexes are used to structure documents for interaction with LLMs. Indexes are commonly used for "retrieval" to find the most relevant documents for a user's query.

LLMs

LangChains provide a standard interface to connect to a number of LLMs(OpenAI, Hugging Face, Cohere, Anthropic, and many more) out there.

Chains

The 'main' idea of LangChain is the ability to chain different components together so that the user can build intricate and interconnected applications that leverage LLMs.

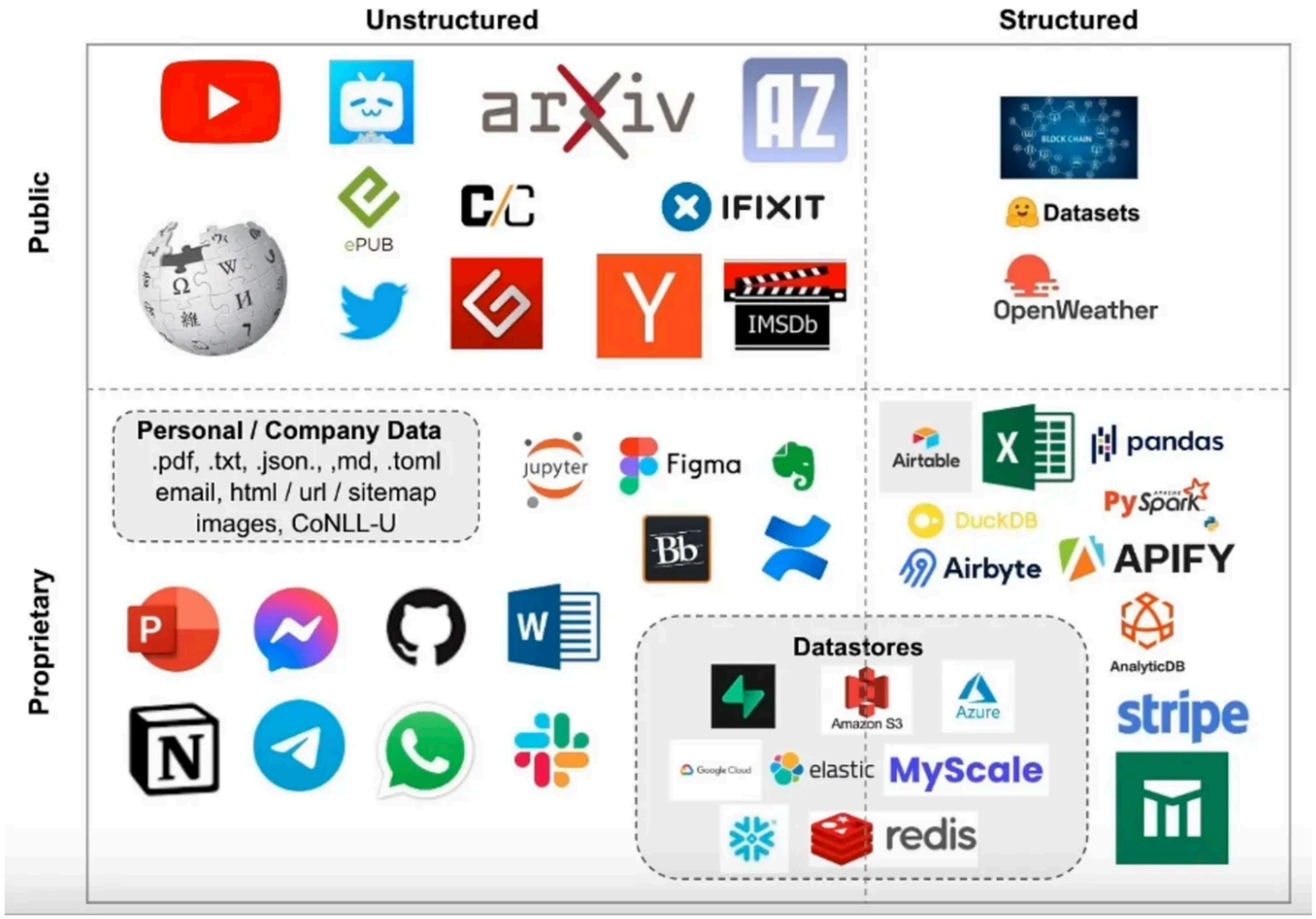
Agents

LangChain Agents employ an LLM as a reasoning mechanism to connect apps to the outside world based on user input. Instead of using a predetermined chain of calls to LLMs/other tools, it uses an LLM to decide a potentially new chain to use that depends on the user's input.

Langchain

Before start

Before we start we need to store the knowledge of your internal documents in a format that is suitable for querying.



Introduction

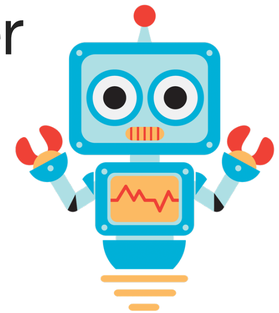
Steps

1. **Split** text corpus of the entire knowledge base into **chunks** - a chunk will represent a single piece of context available to be queried.

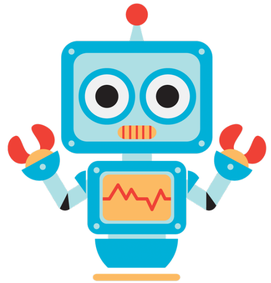
2. **Use** the **Embedding Model** to transform each of the chunks into a vector embedding.

3. **Store** all vector embeddings in a **Vector Database**.

4. **Save** text that represents each of the embeddings separately together with the pointer to the embedding.



Introduction



Steps

5- **Embed** a **question/query** you want to ask using the same Embedding Model that was used to embed the knowledge base itself.

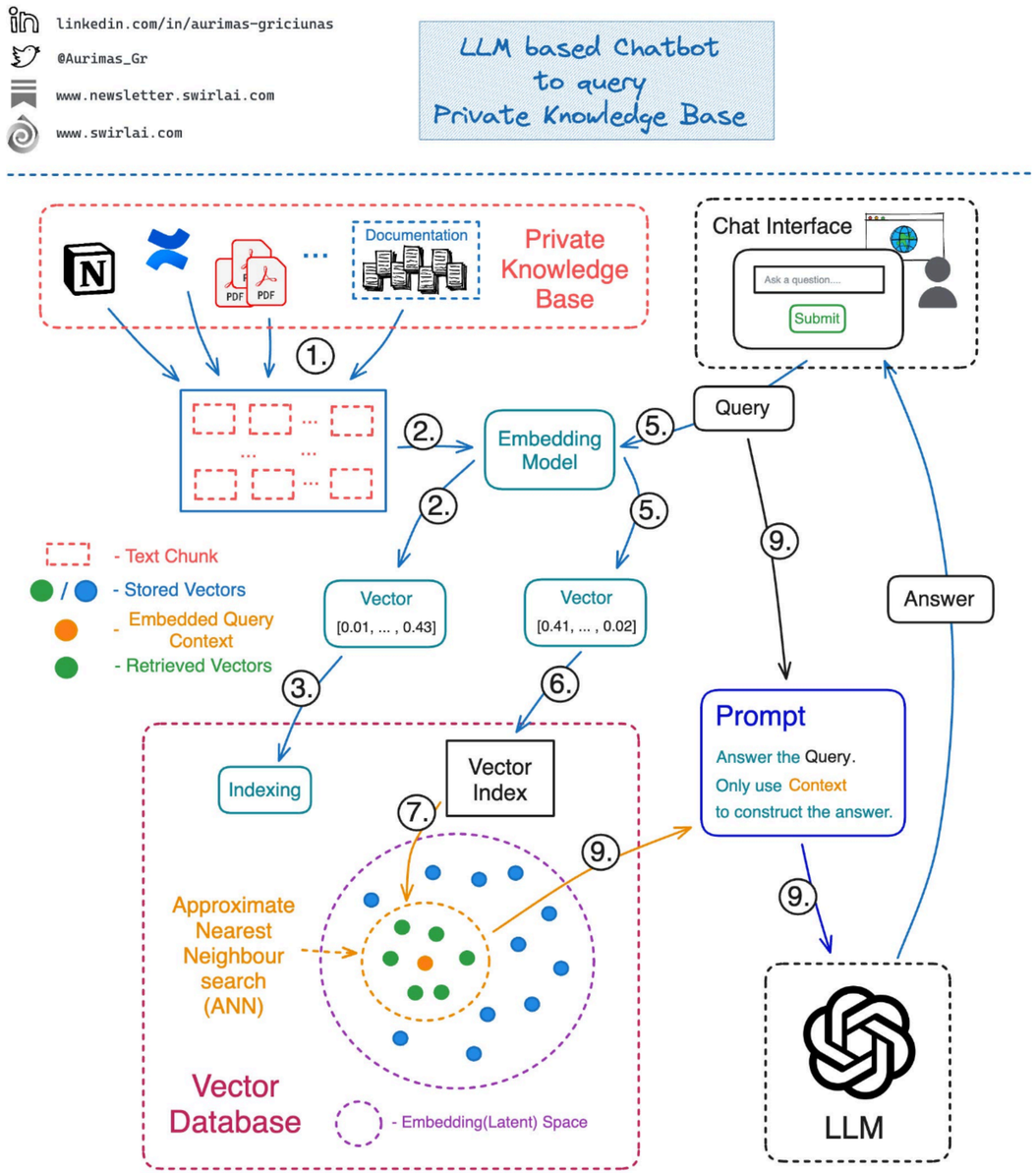
6- **Use** the **resulting Vector Embedding** to run a query against the index in Vector Database.

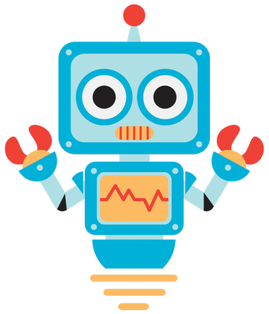
7- Vector DB performs an **Approximate Nearest Neighbour** (ANN) search

8- **Map** the **returned Vector Embeddings** to the **text chunks** that represent them.
9- **Pass** a question together with the **retrieved context text chunks** to the **LLM** via prompt.

Introduction

All Steps





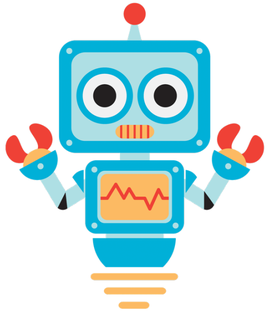
Langchain

Prompts

Prompts are the new way to program models. A prompt refers to the **style of creating inputs** to pass into the model.

Prompts are often constructed from multiple components. **Prompt templates** and **Example selectors** provide main classes and functions to construct and work with prompts easily.

```
(env) Abedelkarims-MacBook-Pro:chat_langchain abedelkarimalbanna$ pip isntall langchain
```



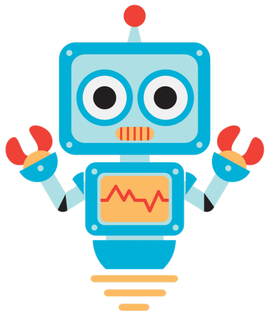
https://python.langchain.com/docs/modules/model_io/prompts/quick_start

Langchain

Prompts

Example Selector Types

Name	Description
Similarity	Uses semantic similarity between inputs and examples to decide which examples to choose.
MMR	Uses Max Marginal Relevance between inputs and examples to decide which examples to choose.
Length	Selects examples based on how many can fit within a certain length
Ngram	Uses ngram overlap between inputs and examples to decide which examples to choose.



https://python.langchain.com/docs/modules/model_io/prompts/quick_start

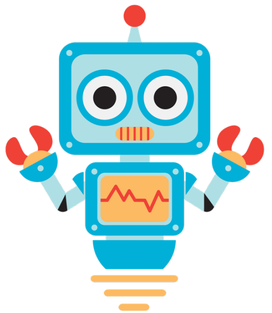
Langchain

Prompts

```python
from langchain.prompts import PromptTemplate

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

```
'Tell me a funny joke about chickens.'
```

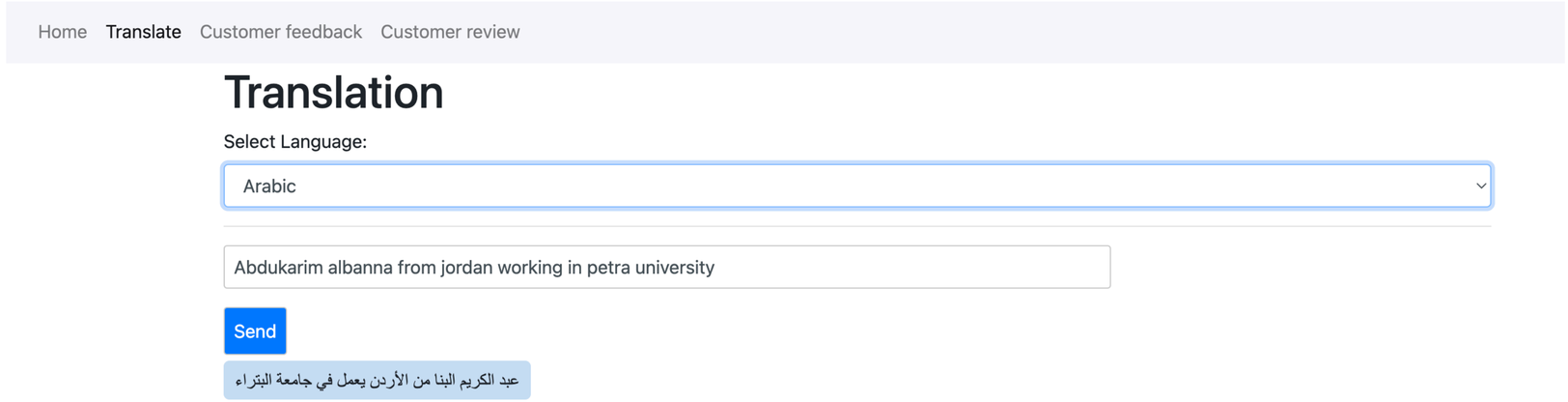
```python
from langchain.prompts import ChatPromptTemplate
```



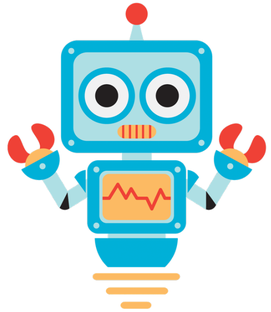
https://python.langchain.com/docs/modules/model_io/prompts/quick_start

Langchain

Prompts for translation app




```
# Define a template string
template_string = """Translate  the text that is delimited by triple backticks \
                into a language that is {language}. text: ```{text}```
                """
prompt_template = ChatPromptTemplate.from_template(template_string)

user_message = prompt_template.format_messages(
    language=chat_language,
    text=chat_text)
```



Langchain

Prompts for email feedback app

Customer Email response

Email

اسمي عبدالكريم البنا اود ان اسأل عن خدمات شركة جوجل للمطورين

Response

عزيزي العميل،

شكرًا لتواصلك مع خدمة العملاء. نحن نقدر استفسارك بشأن خدمات شركة جوجل للمطورين.

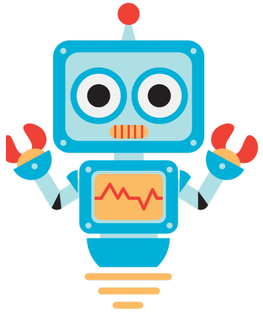
نحن نسعد بتقديم المساعدة وتوفير المعلومات المطلوبة. يُرجى ملاحظة أننا نقدم خدماتنا باللغة العربية، ونحن هنا للإجابة على أي أسئلة قد تكون لديك.

شركة جوجل للمطورين توفر مجموعة واسعة من الخدمات والأدوات للمطورين، بما في ذلك منصات تطوير البرمجيات والتطبيقات وخدمات السحابة والتحليلات والتسويق وغيرها الكثير. يمكنك زيارة موقع جوجل للمطورين للحصول على مزيد من المعلومات والتفاصيل حول الخدمات المتاحة.

إذا كان لديك أي أسئلة أخرى أو إذا كنت بحاجة إلى مساعدة إضافية، فلا تتردد في طرحها. نحن هنا لخدمتك في أي وقت.

شكرًا لاختيارك شركة جوجل للمطورين.

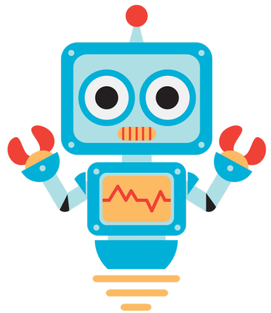
مع خالص التحية،
فريق خدمة العملاء



Langchain

Prompts for email feedback app

```
def customer_feedback(customer_langauge,customer_email):

    template_string = """write response  email from customer service to  the email that is delimited by triple backticks \
                      into a language that is {language} text: ```{email}```
                      """
    prompt_template = ChatPromptTemplate.from_template(template_string)
    customer_messages = prompt_template.format_messages(
        language=customer_langauge,
        email=customer_email)
```



Langchain

Prompts for review app(Output Parser)

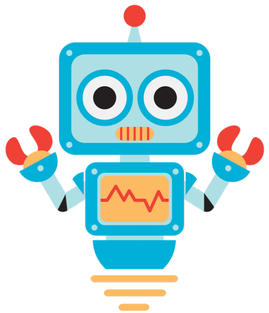
Customer review

Review

اشتريت هدية لصديقي احمد بسعر ٢٠ دينار. و استغرق التوصيل اكثرمن ٥ ايام

Structure response

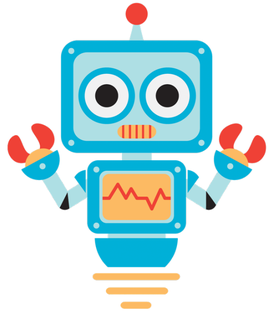

```
{
  "gift": true,
  "delivery_days": 5,
  "price_value": ["اشتريت هدية لصديقي احمد بسعر ٢٠ دينار"]
}
```

Langchain

Prompts for review app

```
review_template = """\
For the following text, extract the following information:

gift: Was the item purchased as a gift for someone else? \
Answer True if yes, False if not or unknown.

delivery_days: How many days did it take for the product \
to arrive? If this information is not found, output -1.

price_value: Extract any sentences about the value or price,\
and output them as a comma separated Python list.

Format the output as JSON with the following keys:
gift
delivery_days
price_value

text: {text}
"""
prompt_template = ChatPromptTemplate.from_template(review_template)
customer_messages = prompt_template.format_messages(text=customer_review)
```



Langchain

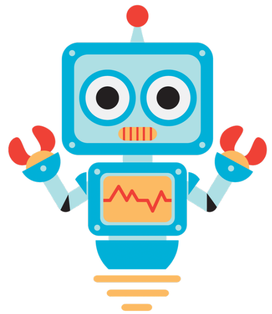
LLM Chains

An LLMChain is a basic but the most commonly used type of chain.

It consists of a PromptTemplate, an Open AI model (an LLM or a ChatModel), and an optional output parser.

LLM chain takes multiple input variables and uses the PromptTemplate to format them into a prompt.

It passes the prompt to the model.
Finally, it uses the OutputParser (if provided) to parse the output of the LLM into a final format.



Langchain

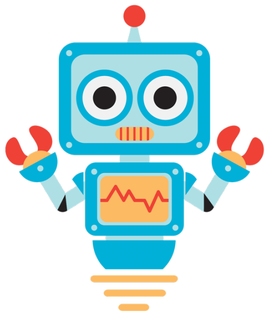
LLM Chains

An LLMChain is a basic but the most commonly used type of chain.

It consists of a PromptTemplate, an Open AI model (an LLM or a ChatModel), and an optional output parser.

LLM chain takes multiple input variables and uses the PromptTemplate to format them into a prompt.

It passes the prompt to the model.
Finally, it uses the OutputParser (if provided) to parse the output of the LLM into a final format.



Langchain

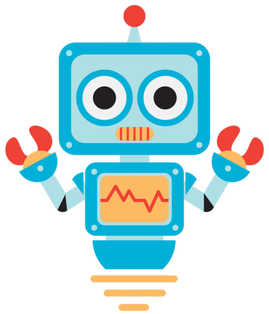
LLM Chains

SequentialChain

A sequential chain combines multiple chains where the output of one chain is the input of the next chain. It runs a sequence of chains one after another.

There are 2 types of sequential chains

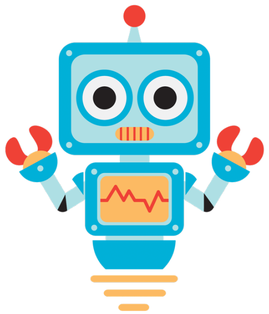
1. SimpleSequentialChain — single input/output
2. SequentialChain — multiple inputs/outputs



Langchain

SequentialChain

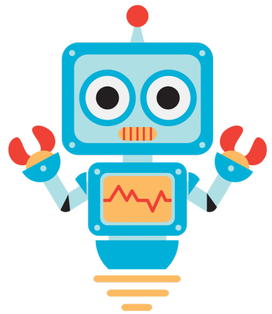

```python
from langchain.llms import OpenAI
from langchain.chains import LLMChain
from langchain.prompts import PromptTemplate
from langchain.chains import SimpleSequentialChain
```



Langchain

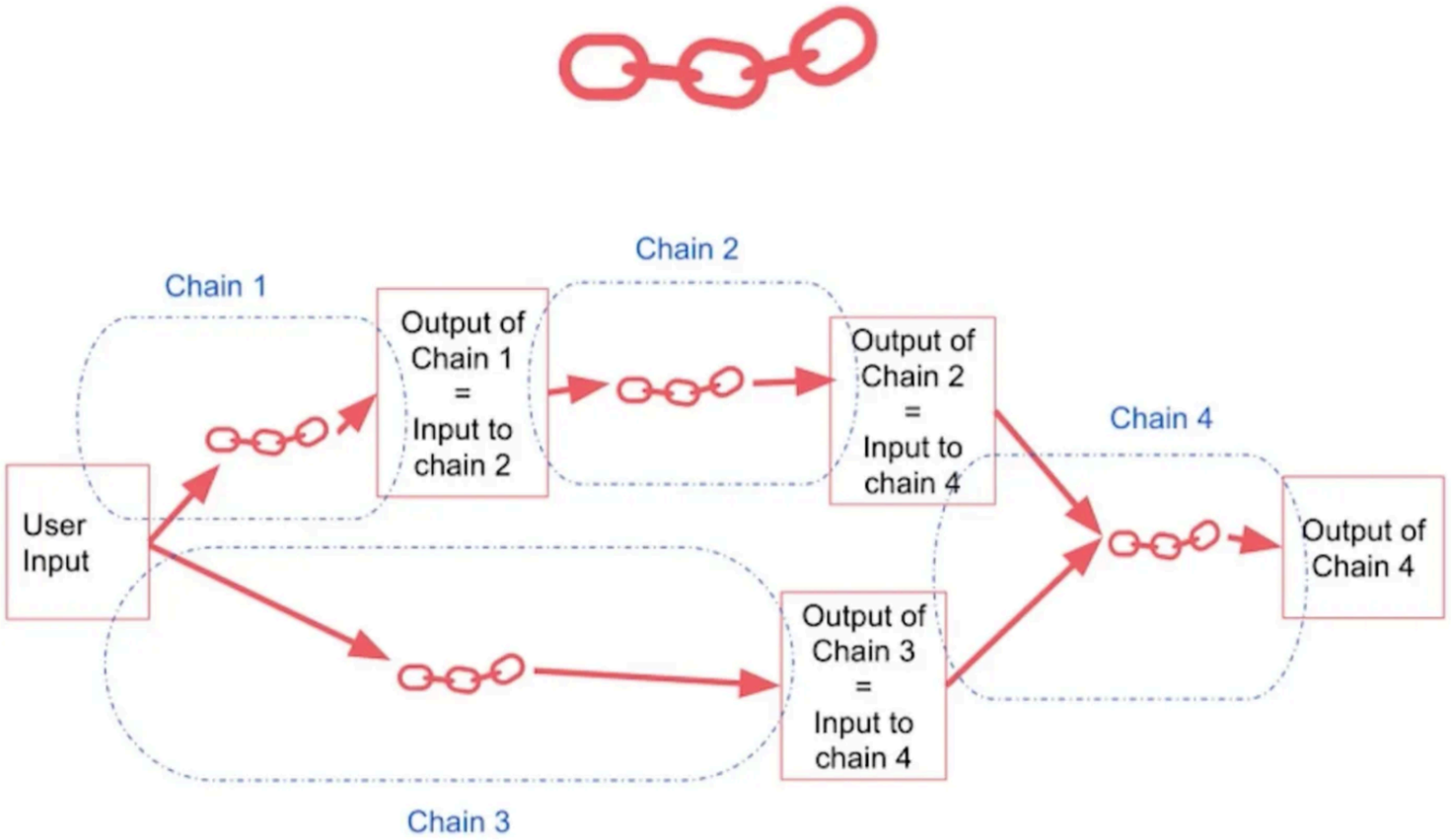
SequentialChain

```
os.environ["OPENAI_API_KEY"] = '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'
# This is an LLMChain to write a synopsis given a title of a play.
llm = OpenAI(temperature=.7)
template = """You are a web developer. given a program name to learn kids programming, it is your job to write a html code of that example.

name: {name}
code: This is html code for a program name:"""
prompt_template = PromptTemplate(input_variables=["name"], template=template)
html_chain = LLMChain(llm=llm, prompt=prompt_template)

# This is an LLMChain to write a review of a play given a synopsis.
llm = OpenAI(temperature=.7)
template = """You are python developer. Given {code}, it is your job to convert this code into python and explain it."""
prompt_template = PromptTemplate(input_variables=["code"], template=template)
python_chain = LLMChain(llm=llm, prompt=prompt_template)

# This is the overall chain where we run these two chains in sequence.
overall_chain = SimpleSequentialChain(chains=[html_chain, python_chain], verbose=True)

review = overall_chain.run("add two numbers")
```



Langchain

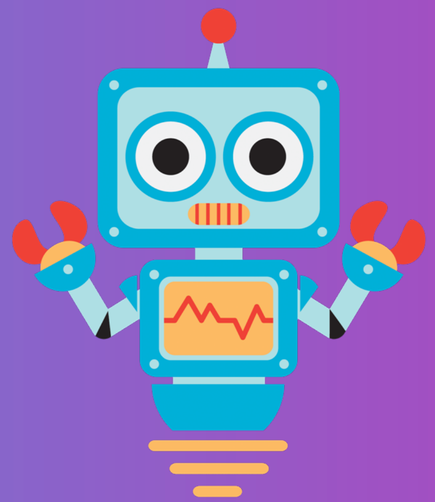
multiple inputs/outputs



Thank you for your attention

Day 4

January 18th 2024



GAME Khoot