Building generative AI applications using LangChain and OpenAI APIs

[[](https://medium.com/@avikumart_?source=post_page-----ee3212400630--------------------------------)](https://medium.com/@avikumart_?source=post_page-----ee3212400630--------------------------------)

[[The Modern Scientist](https://medium.com/the-modern-scientist?source=post_page-----ee3212400630--------------------------------)](https://medium.com/the-modern-scientist?source=post_page-----ee3212400630--------------------------------)

[Avikumar Talaviya](https://medium.com/@avikumart_?source=post_page-----ee3212400630--------------------------------)

·

[Follow](https://medium.com/m/signin?actionUrl=https%3A%2F%2Fmedium.com%2F_%2Fsubscribe%2Fuser%2F13fcacf52e4f&operation=register&redirect=https%3A%2F%2Fmedium.com%2Fthe-modern-scientist%2Fbuilding-generative-ai-applications-using-langchain-and-openai-apis-ee3212400630&user=Avikumar+Talaviya&userId=13fcacf52e4f&source=post_page-13fcacf52e4f----ee3212400630---------------------post_header-----------)

Published in

[The Modern Scientist](https://medium.com/the-modern-scientist?source=post_page-----ee3212400630--------------------------------)

·

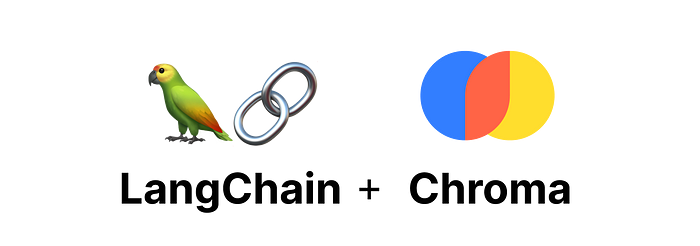
8 min read

·

Nov 7

168

Learn to use LangChain, ChromaDB, and OpenAI API to build a semantic search application pipeline.



Official logos of langchain and Chromadb (source: LangChain docs)

**Introduction**

Generative AI is leading the latest tech wave in the industry. Applications like image generation, text generation, summarization, and question-and-answer bots, to name a few, are booming. As OpenAI led the large language model wave in the recent past, many startups came up with a wide range of tools and frameworks to allow developers to build innovative applications using these LLMs.

One such tool is LangChain, a framework to develop applications powered by LLMs with composability and reliability. LangChain has become the go-to tool for AI developers around the world to build generative AI applications. LangChain also allows for the connection of external data sources and integration with many LLMs available on the market. Apart from this, LLM-powered apps require a vector storage database to store the data they will retrieve later on. In this blog, we will learn about LangChain and Its functions by building an application pipeline with OpenAI APIs and ChromaDB. So let’s get started:

**Table of contents:**

1. Overview of a LangChain
2. Environment set up and loading documents
3. Text Embedding using LangChain and Open Source model
4. Semantic Search Using OpenAI APIs, ChromaDB, and LangChain
5. Conclusion
6. FAQs

**Overview of a LangChain**

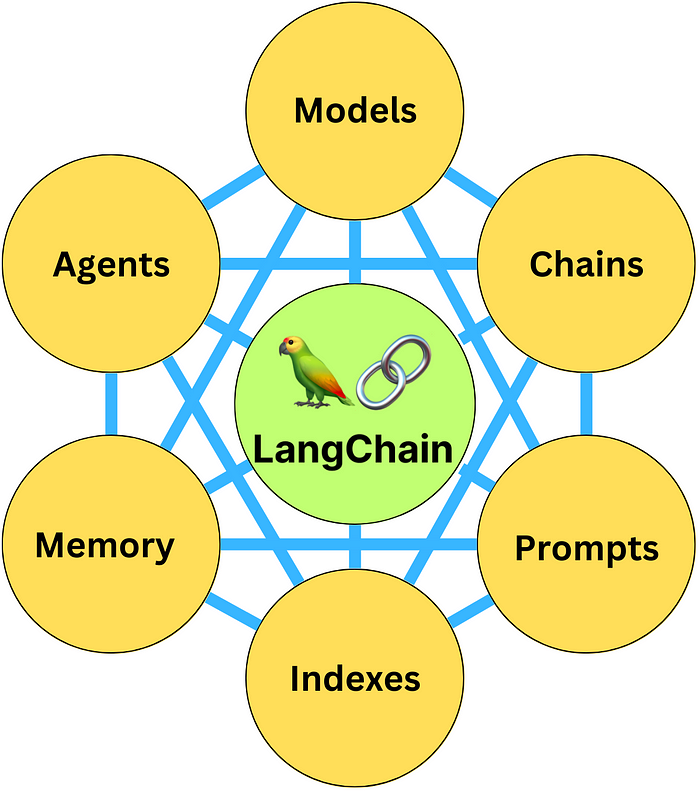
LangChain has become a popular framework for large language model applications in recent times. LangChain provides a sophisticated framework to interact with LLMs, external data sources, prompts, and User Interfaces. The main value propositions of the LangChain are:

1. **Components:** These are the set of abstractions needed to work with language models. Components are modular and easy to use for a wide range of LLM use cases.
2. **Off-the-shelf chains:**A structured assembly of various components and modules to accomplish a specific task such as summarization, Q&A, etc.

LangChain is an Open source project, and since its launch, the project has garnered over 54K+ Github stars, which shows the popularity and acceptability of the project. The project readme file describes the framework in something like this (Source: [project repo](https://github.com/hwchase17/langchain)):

*Large language models (LLMs) are emerging as a transformative technology, enabling developers to build applications that they previously could not. However, using these LLMs in isolation is often insufficient for creating a truly powerful app — the real power comes when you can combine them with other sources of computation or knowledge.*

Clearly, It defines the purpose of the framework with its aims to assist in developing such applications where user knowledge is leveraged.



LangChain Components (source: ByteByteGo NewsLetter)

LangChain has six main components to build LLM applications: model I/O, Data connections, Chains, Memory, Agents, and Callbacks. The framework also allows integration with many tools to develop full-stack applications, such as OpenAI, Huggingface Transformers, and Vectors stores like Pinecone and chromadb, among others. Let’s now look at some of the use cases for LangChain.

1. **Question answering or chat over specific documents**
2. **Chatbots**
3. **Summarization**
4. **Agents**
5. **Interacting with APIs**

These are a few of the many use cases. We will learn and develop a semantic search application for question answering on specific documents using OpenAI APIs and ChromaDB, an open-source vector database. To learn more about the LangChain framework, I highly recommend reading the official documentation. (Link: [here](https://python.langchain.com/docs/get_started/introduction.html))

**Environment set up and loading documents**

Now, we will set up an environment for our semantic search application using OpenAI’s LLM APIs to answer users’ questions over a set of documents. We are using sample documents in this article, but you can use your document to build a question-answering application. First, we need to install the following libraries:

**Installing the project dependencies**

# install openai, langchain, sentense transfoers and other dependencies  
!pip install openai langchain sentence\_transformers -q  
!pip install unstructured -q  
  
# install the environment dependencies  
!pip install pydantic==1.10.8  
!pip install typing-inspect==0.8.0 typing\_extensions==4.5.  
!pip install chromadb==0.3.26

LangChain requires some of the environment dependencies with a specific version, such as pedantic, typing extensions, and chromadb. Once the installation is complete, you will be able to run the following code in your colab or any other notebook environment.

**LangChain document loader**

LangChain provides document loader classes to load documents from user input or any database. It supports a variety of file formats, such as HTML, JSON, CSV, etc. We have a few text files that we will use in our use case. You can find the files in the GitHub repo. (GitHub repo — [Link](https://github.com/avikumart/LLM-GenAI-Transformers-Notebooks/tree/main/ChromaDB_semantic_search/pets))

# import langchain dir loader from document loaders  
from langchain.document\_loaders import DirectoryLoader  
  
# directory path  
directory = '/content/pets'  
  
# function to load the text docs  
def load\_docs(directory):  
 loader = DirectoryLoader(directory)  
 documents = loader.load()  
 return documents  
  
documents = load\_docs(directory)  
len(documents)  
  
---------------------------[Output]----------------------------------------  
[nltk\_data] Downloading package punkt to /root/nltk\_data...  
[nltk\_data] Unzipping tokenizers/punkt.zip.  
[nltk\_data] Downloading package averaged\_perceptron\_tagger to  
[nltk\_data] /root/nltk\_data...  
[nltk\_data] Unzipping taggers/averaged\_perceptron\_tagger.zip.  
5

Once the data is loaded, we will use a *text splitter* to split the text documents into the *fixed size of chunks* to store them in the vector database. LangChain offers multiple text splitters such as *split by character*, *split by code*, etc.

# use text splitter to split text in chunks  
from langchain.text\_splitter import RecursiveCharacterTextSplitter  
  
# split the docs into chunks using recursive character splitter  
def split\_docs(documents,chunk\_size=1000,chunk\_overlap=20):  
 text\_splitter = RecursiveCharacterTextSplitter(chunk\_size=chunk\_size, chunk\_overlap=chunk\_overlap)  
 docs = text\_splitter.split\_documents(documents)  
 return docs  
  
# store the splitte documnets in docs variable  
docs = split\_docs(documents)

Once the documents are converted into chunks, we will embed them into a vector using open-source embedding models in the next section.

**Text Embedding using LangChain and Open Source model**

Embedding texts is the most important concept in the LLM application development pipeline. All the text documents need to be vectorized before they can be processed for certain tasks like semantic search, summarization, etc. We will use the open-source sentence-transformer model *“all-MiniLM-L6-v2”*for text embeddings. Once the documents are embedded, we can store them in the open-source vector database ChromaDB to perform a semantic search. Let’s look at the hands-on code example

# embeddings using langchain  
from langchain.embeddings import SentenceTransformerEmbeddings  
embeddings = SentenceTransformerEmbeddings(model\_name="all-MiniLM-L6-v2")  
  
# using chromadb as a vectorstore and storing the docs in it  
from langchain.vectorstores import Chroma  
db = Chroma.from\_documents(docs, embeddings)  
  
# Doing similarity search using query  
query = "What are the different kinds of pets people commonly own?"  
matching\_docs = db.similarity\_search(query)  
  
matching\_docs[0]  
  
--------------------------[output]----------------------------------------  
Document(page\_content='Pet animals come in all shapes and sizes,   
each suited to different lifestyles and home environments.   
Dogs and cats are the most common, known for their companionship   
and unique personalities. Small mammals like hamsters, guinea pigs,   
and rabbits are often chosen for their low maintenance needs.  
 Birds offer beauty and song, and reptiles like turtles and   
lizards can make intriguing pets. Even fish, with their calming presence,   
can be wonderful pets.',   
metadata={'source': '/content/pets/Different Types of Pet Animals.txt'})

In the above code, we use embeddings to store in ChromaDB, which supports in-memory storage. so we can query the database to get the answer from our text documents. We asked to know about different kinds of pets commonly owned by people, and it resulted in a correct answer with the source of the answer.

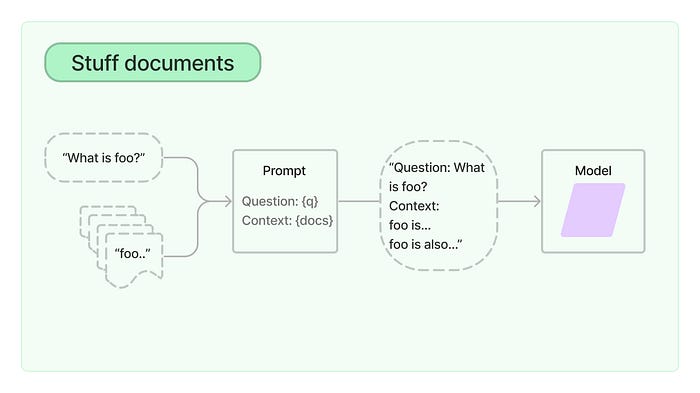
**Semantic Search Using OpenAI APIs, ChromaDB, and LangChain**



Open AI Logo (source: Wikimedia)

Semantic search application pipelines entail the process of interpreting the intent and context of search terms and documents to produce more precise search results. Semantic search pipelines can increase search accuracy by comprehending user intent, examining the connections between words and concepts, and producing context-aware search results by utilizing methods like attention mechanisms in the field of natural language processing (NLP).

LangChain offers an OpenAI chat interface to call the model APIs into your application and create a question/answer pipeline that answers users’ queries based on given context or input documents. It basically performs a vectorized search to find the most similar answer to the question. (Refer to the below flow chart.)



Context-based search pipeline (source: LangChain Docs)

**Semantic search Q&A using LangChain and OpenAI APIs**

# insert an openai key below parameter  
import os  
os.environ["OPENAI\_API\_KEY"] = "YOUR-OPENAI-KEY"  
  
# load the LLM model  
from langchain.chat\_models import ChatOpenAI  
model\_name = "gpt-3.5-turbo"  
llm = ChatOpenAI(model\_name=model\_name)  
  
  
# Using q&a chain to get the answer for our query  
from langchain.chains.question\_answering import load\_qa\_chain  
chain = load\_qa\_chain(llm, chain\_type="stuff",verbose=True)  
  
# write your query and perform similarity search to generate an answer  
query = "What are the emotional benefits of owning a pet?"  
matching\_docs = db.similarity\_search(query)  
answer = chain.run(input\_documents=matching\_docs, question=query)  
answer  
  
-----------------------------------[Results]---------------------------------  
'Owning a pet can provide numerous emotional benefits. Pets offer  
 companionship and can help reduce feelings of loneliness and isolation.   
They provide unconditional love and support, which can boost mood and overall  
 well-being. Interacting with pets, such as petting or playing with them,   
has been shown to decrease levels of stress hormones and increase the  
 release of oxytocin, a hormone associated with bonding and relaxation.   
Pets also offer a sense of purpose and responsibility, as taking care of   
them can give a sense of fulfillment and provide a distraction from daily   
stressors. Additionally, the bond between pets and their owners can provide  
 a sense of stability and consistency during times of personal or societal   
stress.'

The above code, calls the*“gpt-3.5-turbo”* model API using LangChain’s ChatOpenAI() function and creates a q&a chain for answering our query. for more detailed information on code, you can visit LangChain’s official documentation ([here](https://python.langchain.com/docs/modules/model_io/models/chat/)) and Github code notebook ([here](https://github.com/avikumart/LLM-GenAI-Transformers-Notebooks/blob/main/Langchain_with_chromaDB.ipynb))

**Conclusion**

In conclusion, the blog post explored the exciting domain of building generative AI applications using LangChain and OpenAI APIs. We saw an overview of LangChain and its various components, along with use cases for LLM applications. Generative AI has revolutionized various domains, allowing us to generate realistic text, images, videos, and more. Semantic search is one such application used to build question-and-answer applications using OpenAI LLMs like GPT-3.5 and GPT-4. Let’s look at the key takeaways from this blog:

1. We learned about a brief overview of LangChain — An open source framework to build LLM-powered applications.
2. We learned to use LangChain and ChromaDB — A vector database to store embeddings for similarity search applications.
3. Finally, we learned about OpenAI LLM APIs to build a semantic search pipeline using LangChain

**FAQs**

**Q1: What is a generative AI?**

**A:** Generative AI is a type of machine-learning technology that learns from a large amount of unstructured data to generate new text, images, music, and even videos.

**Q2: What is a semantic search?**

**A:** Semantic search applications entail the process of interpreting the intent and context of search terms and documents to produce precise search results given a certain query by users.

**Q3: What is a LangChain and what are its key features?**

**A:** LangChain is a popular framework in large language model application development. LangChain provides a framework to interact with LLMs, external data sources, prompts, and User Interfaces.

**Q4: What are the benefits of LangChain?**

**A:** LangChain enables connection with a wide range of external sources of data. It also integrates LLM in various use cases such as a chatbot, summarization, and code generation.

**Q5: What databases are supported by LangChain?**

**A:**LangChain supports vector databases like ChromaDB, and Pinecone as well as structured databases like MS SQL, MySQL, MariaDB, PostgreSQL, Oracle SQL, etc.