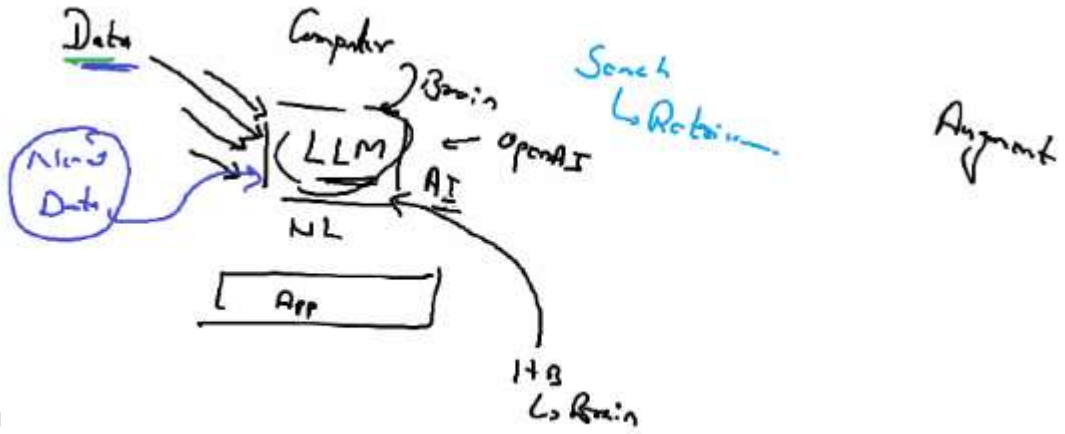
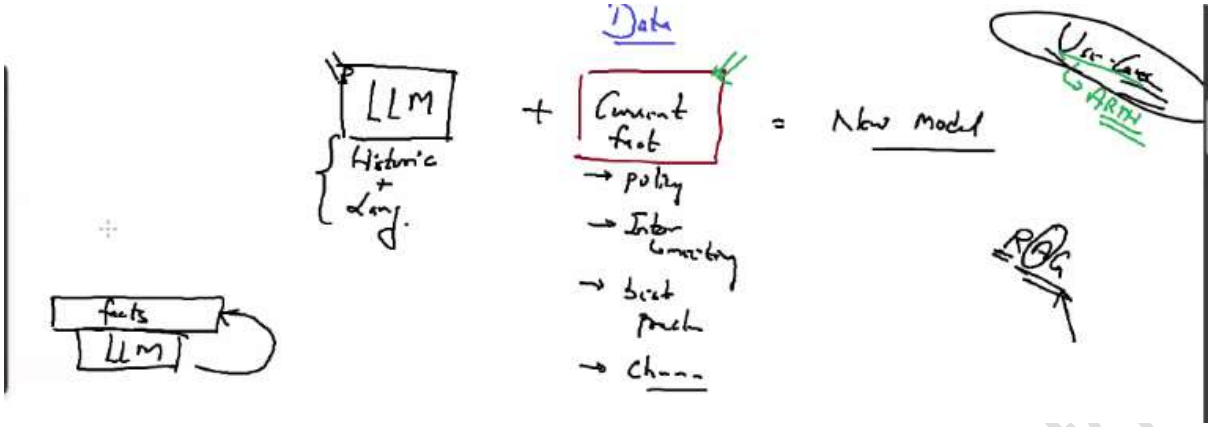
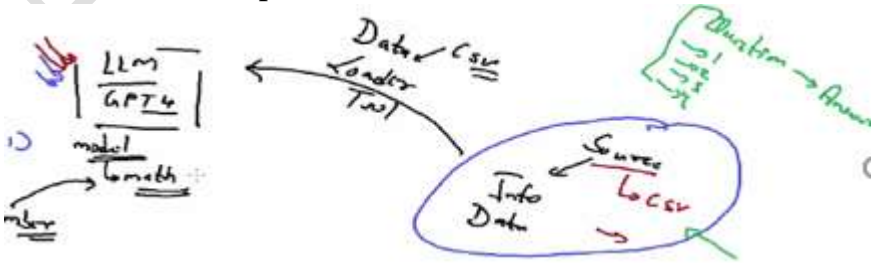
- Today we created an AI Application
- The concepts behind Large Language Models (LLMs), particularly how they can be augmented and retrained to solve real-world problems
- LLMs are not databases or search engines; they are trained on a specific dataset. The knowledge they can retrieve is limited to what they were trained on.
- large language models (LLMs), like GPT models, are not search engines or databases. They are only capable of providing information based on the data they were trained on and cannot retrieve new or up-to-date facts.
- LLMs are designed to generate text based on language understanding but aren't meant for retrieving new facts or up-to-date information from external databases or search engines.
- LLMs may generate incorrect or outdated information if asked about topics they were not trained on or if the information has changed since their training.
- For instance, the example of a fee structure in a specific course illustrates how an LLM could provide outdated information if it hasn’t been updated or retrained.



- Large language models like GPT-4 are trained on publicly available data from the internet, but they do not have access to internal or proprietary knowledge that may be specific to a company or use case.
- To make these models more effective for specific business use cases, companies can augment them with private, internal data that isn’t available online.



- The process of enhancing LLMs with additional data is often referred to as Retrieval-Augmented Generation (RAG). This method involves retrieving specific pieces of information from a knowledge base (e.g., databases, PDFs, CSVs) and then using that information to generate contextually accurate responses
- By adding company-specific knowledge, the model can answer queries in a way that reflects the company's unique processes or information. For instance, internal company policies, customer support practices, or sales pitches can be fed into the model to create customized answers.
- The model can be augmented using a variety of sources, such as internal databases, documents, PDFs, or spreadsheets. This process is not the same as fine-tuning, which involves training the model to behave or respond in a certain way.
- The model can be augmented using a variety of sources, such as internal databases, documents, PDFs, or spreadsheets. This process is not the same as fine-tuning, which involves training the model to behave or respond in a certain way.
- how the model, augmented with specific knowledge about their Arth program, can generate customized email responses. This kind of customization can be useful in real-world support scenarios where juniors can use the augmented AI to assist customers effectively by leveraging senior-level best practices

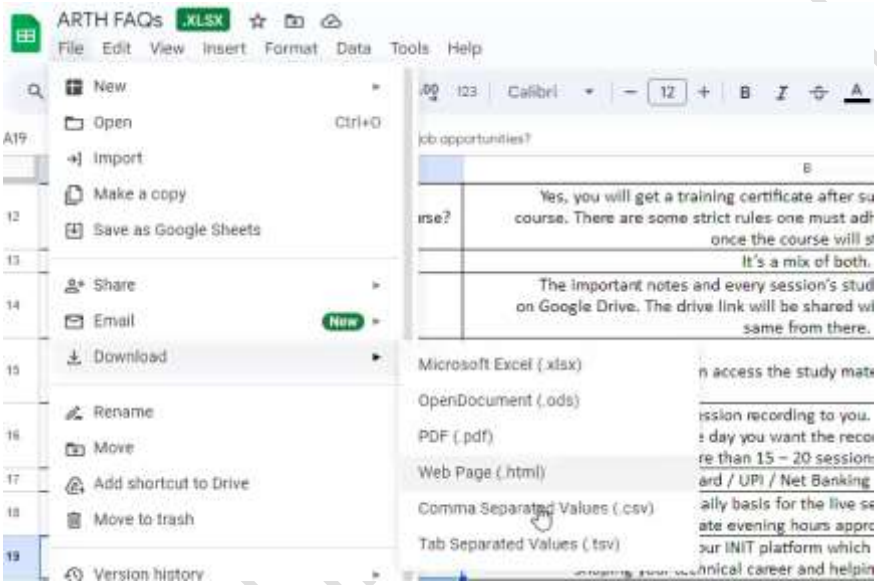


- RAG is especially useful in fields such as IT security, where challenges are often solved using company-specific methods. This approach can enable even junior employees to provide high-quality support by leveraging augmented AI models.
- how to set up a Retrieval-Augmented Generation (RAG) model. They explain that RAG involves adding external knowledge to an existing large language model (LLM), like GPT-4, from OpenAI. This added knowledge can come from various sources such as databases, CSV files, PDFs, or Excel sheets. RAG is different from fine-tuning because RAG augments the model with external data rather than modifying the model's internal behavior
- RAG involves using external data to augment a model's knowledge and provide better, more context-aware responses without changing the model itself
- For practice we create a FAQ data

	A	B	C
	Customer Question	**Response**	
	What is ARTH ?	ARTH is a Hindi language word अर्थ which means "meaning, significance. Intention, aim, substance". Talking about LinuxWorld's Biggest Initiative. "ARTH - The School of Technologies" which is a life-changing 12 Months Online Live Technical Program to mould and prepare the technical mindset and skills to get valued asidol in the IT industry. ARTH will include everything a pursuing technicalstudent needs from technically sounded mind to logically strong concepts. It is a rigorous program integrating 10+ technologies (realm of technologies)	
	Who should join ARTH Program?	ARTH is mainly focused on pursuing technical students With ARTH, you can : ⊡ Upgrade your technical skills to different level, Implementing most demanded technologies in the industry today and near future. ⊡ Convert yourself from just a basic engineer to a totally most-demanded and skilled engineer of the industry. ⊡ Get assured about your placements and technical skills about high-level technologies after getting trained under the World Record Holder Mr. Vimal Daga.	
	Are there any prerequisites or language requirements?	There are no prerequisites for ARTH. Everything will be starting from the basic as 0 and to the real depth of specific technology.	
	How do I contact technical support during my technical	Once you will get registered for ARTH, You will be added to a WhatsApp group which have some technical volunteers with different time slots to help you out completely. Technical volunteers will be there every time to sort	
10	How can ARTH boost my career path?	person's career into the most demanded idol engineer of the industry which will be high-level skilled personality. He/ She will be the One & Only One who will know the maximum number of technologies & Recruiter's No 1 Choice.	
11	I am a professional. Can I register for ARTH ?	No, you cannot. This course is specifically for ongoing college technical students only.	
12	Will I get a certificate after completion of this course?	Yes, you will get a training certificate after successful completion of this course. There are some strict rules one must adhere to which will be explained once the course will start.	
13	Do we build projects in groups or individual ?	It's a mix of both.	
14	How can I access the study material?	The important notes and every session's study material will be uploaded on Google Drive. The drive link will be shared with you and you can access the same from there.	
15	How long can I access the study material ? Will it be accessible even after the course ends?	Yes, You can access the study material for life time.	
16	What if I miss any session due to some reason?	We will provide the session recording to you. You will be required to fill a Google form on the same day you want the recording for. But yes we adhere to strict norms of not missing more than 15 – 20 sessions in the complete span of 12 months	
17	What are the accepted payment methods ?	Debit Card / UPI / Net Banking / Credit Card	
18	How many number of hours do I need to spend everyday?	Approx 3 - 4 hours on daily basis for the live sessions. The sessions will be scheduled in the late evening hours approx. 7:15 pm IST onwards	

12	Will I get a certificate after completion of this course?	Yes, you will get a training certificate after successful completion of this course. There are some strict rules one must adhere to which will be explained once the course will start.
13	Do we build projects in groups or individual ?	It's a mix of both.
14	How can I access the study material?	The important notes and every session's study material will be uploaded on Google Drive. The drive link will be shared with you and you can access the same from there.
15	How long can I access the study material ? Will it be accessible even after the course ends?	Yes, You can access the study material for life time.
16	What if I miss any session due to some reason?	We will provide the session recording to you. You will be required to fill a Google form on the same day you want the recording for. But yes we adhere to strict norms of not missing more than 15 – 20 sessions in the complete span of 12 months
17	What are the accepted payment methods ?	Debit Card / UPI / Net Banking / Credit Card
18	How many number of hours do I need to spend everyday?	Approx 3 - 4 hours on daily basis for the live sessions. The sessions will be scheduled in the late evening hours approx. 7:15 pm IST onwards
19	Will I be assisted for placement / any other job opportunities?	Yes definitely. Refer our INIT platform which is dedicatedly focusing on shaping your technical career and helping you shape the world

- To load this data we need to download this data into a CSV file



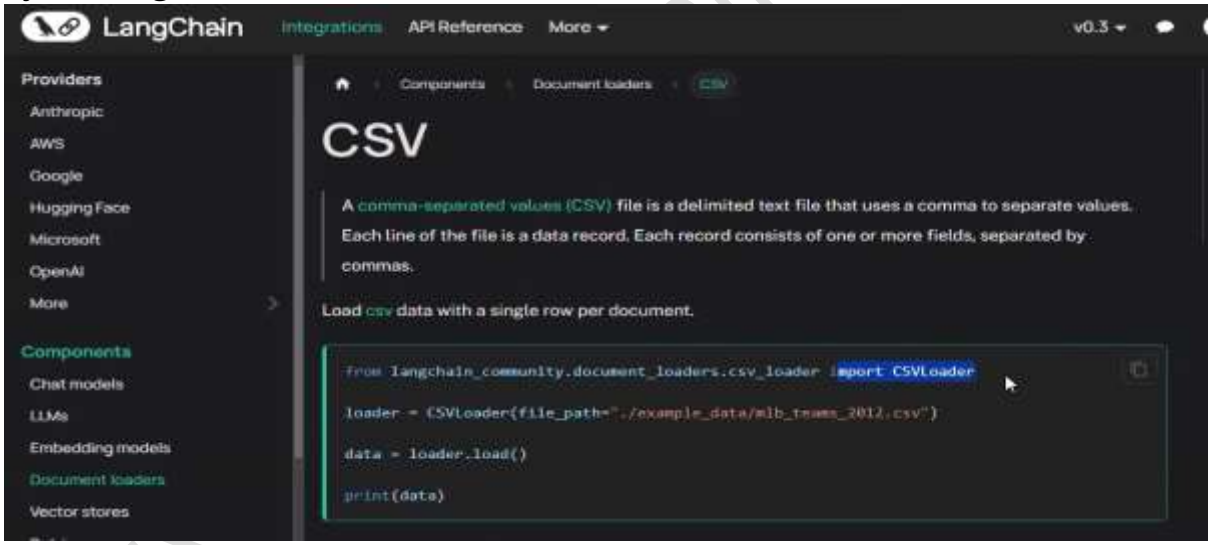
- The knowledge for the RAG model can come from CSVs, databases, or other structured/unstructured formats
- Using a CSV file of frequently asked questions (FAQs) from their marketing and customer support teams to load into the model. This enables the model to provide answers based on real customer inquiries and responses
- The use of tools, like LangChain, to load different data formats (such as CSVs) into the model to augment it
- After the CSV file download go to CMD and open the CSV file

```
Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~
$ cd Documents/

Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~/Documents
$ cd GenAIOps_Training_2024/

Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~/Documents/GenAIOps_Training_2024
$ ls
 ARTH.csv                  RAG.ipynb    Untitled.ipynb    untitled.txt
'Google LangChain.ipynb'   RAG.py       lw.csv

Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~/Documents/GenAIOps_Training_2024
$ vim ARTH.csv
```

- After that every record we need to use the "|" symbol

```
Customer Question|Response
What is ARTH ?|"ARTH is a Hindi language word अर्थ which means "meaning, signific
ance.
intention, aim, substance". Talking about LinuxWorld's Biggest Initiative,
"ARTH - The School of Technologies" which is a life-changing 12 Months Online Li
ve Technical Program to mould and prepare the technical mindset and skills to ge
t valued asidol in the IT industry. ARTH will include everything a pursuing tech
nicalstudent needs from technically sounded mind to logically strong concepts. I
t is a rigorous program integrating 10+ technologies (realm of technologies)"

Who should join ARTH Program?|"ARTH is mainly focused on pursuing technical stud
ents With ARTH, you can :
 Upgrade your technical skills to different level, implementing most
demanded technologies in the industry today and near future.
 Convert yourself from just a basic engineer to a totally most-demanded
and skilled engineer of the industry.
 Get assured about your placements and technical skills about high-level
technologies after getting trained under the World Record Holder Mr.
Vimal Daga."

Are there any prerequisites or language requirements?|There are no prerequisites
```

- Means the first column and the second column have separate
- manually processed the data (820 records) and created a CSV file using a pipe (|) as the separator rather than a comma.
- Now we create pipe separate file
- Now we need to load the data into the model and this load function is given by the langchain

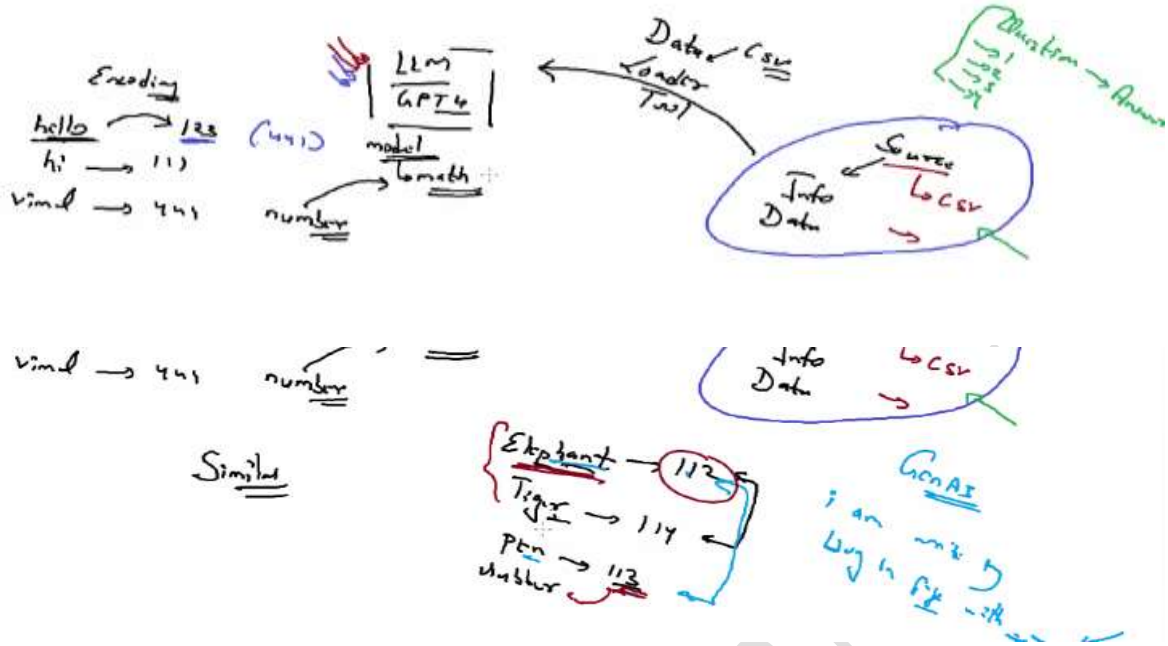


- handling a CSV file and loading it into a program, specifically using the CsvLoader function to load the data for training a machine learning model.
- They used the CsvLoader function from LangChain libraries to load the data.
- Now we start our practical
- First we Loading CSV Data with LangChain

```
In [1]:
from langchain_community.document_loaders.csv_loader import CSVLoader

In [4]:
loader = CSVLoader(file_path="ARTH.csv", encoding='utf-8',  csv_args={
            'delimiter': '|',

        })

In [5]:
loader.load()

Out[5]: [Document(metadata={'source': 'ARTH.csv', 'row': 0}, page_content='Customer Question|Response: W
hat is ARTH ?|"ARTH is a Hindi language word अर्थ which means "meaning\nNone: significance.'),
 Document(metadata={'source': 'ARTH.csv', 'row': 1}, page_content='Customer Question|Response: i
ntention\nNone: aim,substance". Talking about LinuxWorld's Biggest Initiative,'),
 Document(metadata={'source': 'ARTH.csv', 'row': 2}, page_content='Customer Question|Response:
"ARTH - The School of Technologies" which is a life-changing 12 Months Online Live Technical Pro
gram to mould and prepare the technical mindset and skills to get valued asidol in the IT indust
ry. ARTH will include everything a pursuing technicalstudent needs from technically sounded mind
to logically strong concepts. It is a rigorous program integrating 10+ technologies (realm of te
chnologies)"'),
 Document(metadata={'source': 'ARTH.csv', 'row': 3}, page_content='Customer Question|Response: W
```

```
In [19]:
dataset = loader.load()
```

- The CSVLoader is used to load a CSV file (called "ARTH.csv") that contains past customer interaction data or best practice responses
- **file_path="ARTH.csv"** specifies the location of the file
- **encoding="utf-8"** ensures the CSV is read using the proper encoding to avoid character errors
- An error occurred during the file loading process due to encoding issues because the file contained non-English characters (in this case, Hindi)
- fixed the issue by specifying an appropriate encoding (such as UTF-8 or UTF-16) to properly load non-English characters.
- Machine learning models, including language models, can only process numerical data, not words or characters from any language like Hindi or English. Hence, all input words must be encoded as numbers
- Each word or token is assigned a numerical representation. This could be done through encoding methods, where words like "Hello" or "Hi" are mapped to specific numbers. However, the encoding must align with the pre-trained language model's internal representations, as these models already have predefined numerical encodings for words from their training data.
- For generative AI models like GPT, words have already been mapped to specific numbers during training. If we provide new encodings that don't match the model's internal system, the model will become confused, and predictions or outputs may become inaccurate



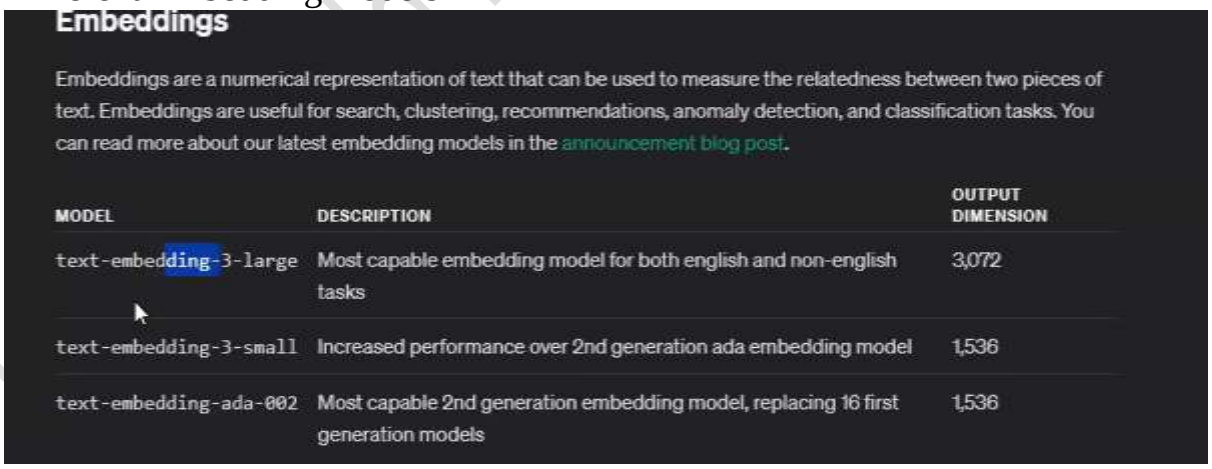
- While encoding, it's important to maintain the semantic relationships between words. For example, words like "Elephant" and "Tiger" should have closer numerical representations because they are conceptually related (both are animals). In contrast, words like "Pen" and "Elephant" should be encoded with numbers that reflect their lack of similarity
- Companies often provide separate models called "embedding models" that handle the encoding of words into numbers, maintaining both the model's original encodings and the necessary semantic similarities between words
- When introducing new data into pre-trained models like GPT, the encoding must match the format the model was trained with, or the model will misinterpret the data, leading to incorrect outputs
- **csv_args={'delimiter': '|'}**: The CSV file is expected to use a pipe symbol (|) as the delimiter between columns, not the typical comma
- They demonstrated adjusting the field separator from a comma to a pipe and verified that the data was loaded correctly
- The loader. load() function loads this CSV into a dataset that will be used later for generating responses based on similarity searches
- After that we need to set the Environment Variable Loading

```
In [14]:  1 from dotenv import load_dotenv

In [15]:  1 load_dotenv()
Out[15]: True

In [16]:  1 import os
          2 openAIApi_key = os.getenv("llm_api_key")
```

- load_dotenv() loads environment variables from a .env file. This file stores sensitive information (like API keys) to avoid hardcoding them in the code
- Retrieving the OpenAI API Key
- The os.getenv() method retrieves the value of the environment variable named "llm_api_key" which is your OpenAI API key
- The OpenAI API key is necessary to interact with OpenAI's language model (GPT-4) for generating responses
- Setting up Embeddings Model

```
In [11]: H  1 from langchain.embeddings.openai import OpenAIEmbeddings

In [17]: H  1 myembedModel = OpenAIEmbeddings( openai_api_key=openAIApi_key)
```

- This part imports and initializes the **OpenAIEmbeddings** model, which converts your text data into numerical vector embeddings.
- Embeddings are used to measure how similar two pieces of text are by representing them in vector space.
- **openai_api_key=openAIApi_key** uses the API key retrieved earlier to access OpenAI's API for embedding generation.
- embedding models are used to convert text data into numerical representations to measure similarities. These models are essential for various platforms, like OpenAI's ChatGPT, that rely on embeddings for processing and comparing data
- Different Embedding models



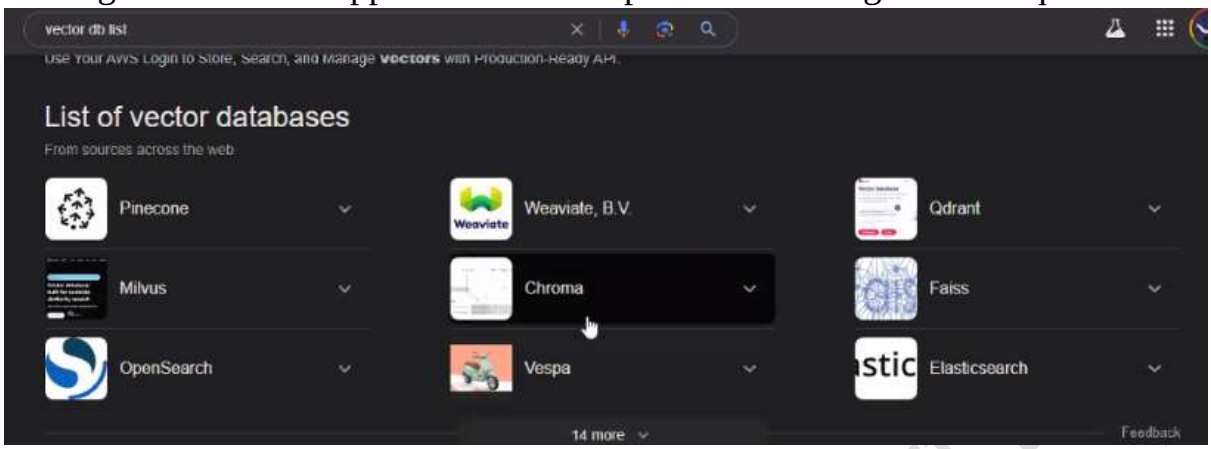
Embeddings

Embeddings are a numerical representation of text that can be used to measure the relatedness between two pieces of text. Embeddings are useful for search, clustering, recommendations, anomaly detection, and classification tasks. You can read more about our latest embedding models in the announcement blog post.

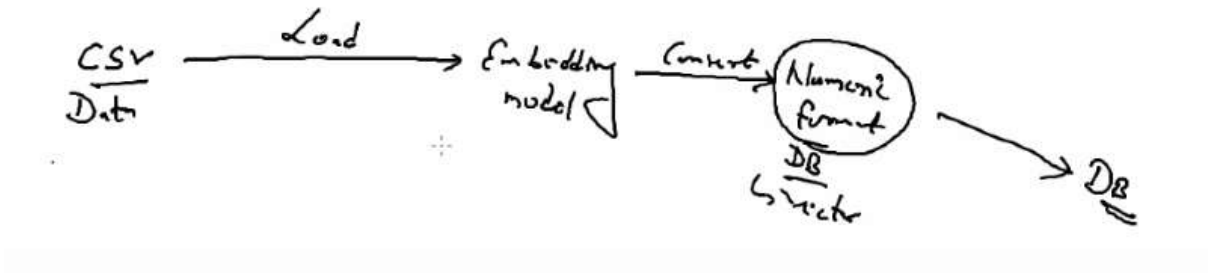
MODEL	DESCRIPTION	OUTPUT DIMENSION
text-embedding-3-large	Most capable embedding model for both english and non-english tasks	3,072
text-embedding-3-small	Increased performance over 2nd generation ada embedding model	1,536
text-embedding-ada-002	Most capable 2nd generation embedding model, replacing 16 first generation models	1,536

- **Embedding Models**: These are models provided by platforms like OpenAI, which take text data and transform it into a numerical format (vector representation) to measure similarity between different data points
- Since embedding models produce high-dimensional vector data, traditional databases like MySQL or MongoDB, which store 2D data (table-like structures), cannot handle such data effectively. Instead, specialized vector databases are used, such as **Pinecone**, **Chroma**, and **FAISS** (an open-source

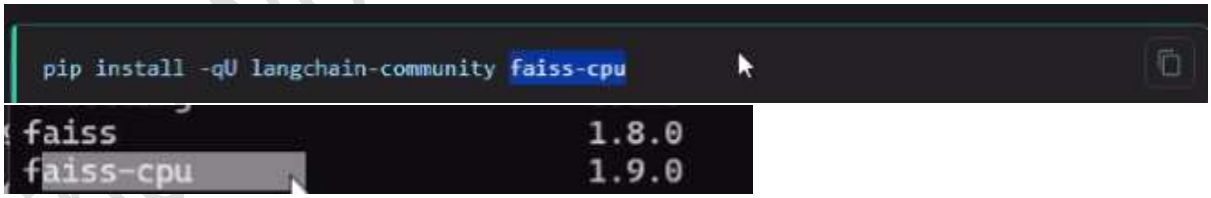
vector database from Facebook). FAISS works as an in-memory database, making it suitable for applications where permanent storage is not required.



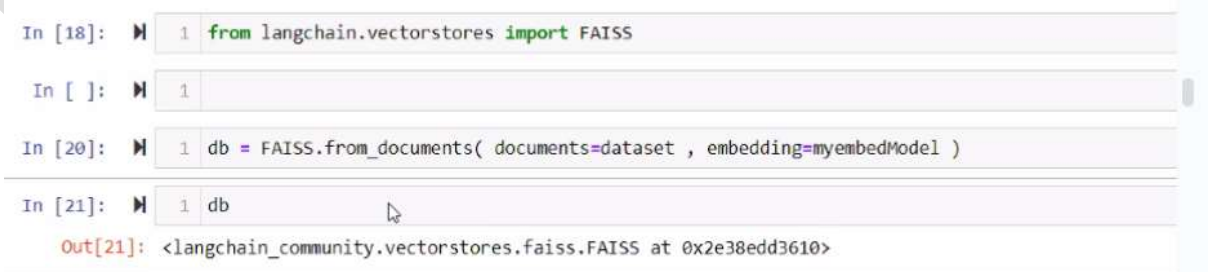
- load CSV data into the embedding model, convert it into vector format and store the resulting vector data in FAISS for further use. FAISS works in memory, meaning the data is not permanently stored but is sufficient for the current demo



- Storing Data in a FAISS Vector Store
- For that first we need to install the FAISS Vector Store



- After that import Faiss



- **FAISS** is a library for fast nearest-neighbor search, which allows searching for similar documents in large datasets.

- FAISS.from_documents() creates a searchable vector store from the dataset loaded earlier. It uses the embeddings model to convert each document into a vector and store them in FAISS.
- Later, you will use this FAISS vector store to search for the best practices that are most similar to the customer's message
- **db.similarity_search(query)**: This searches the FAISS store using the embeddings, returning documents that are close in vector space to the query

```
In [22]: db.similarity_search("what is the fees of ARTH")

Out[22]: [Document(metadata={'source': 'ARTH.csv', 'row': 20}, page_content='Customer Question: what is
fees of ARTH program\nResponse: it is around 4000 rupees per month'),
 Document(metadata={'source': 'ARTH.csv', 'row': 5}, page_content='Customer Question: How can S
one register for ARTH ?\nResponse: To register for ARTH, you need to get in touch with our ARTH
am on +91 9772201449 & they will help you with the process. Once the process is completed you w
be the lucky one to get mentored by World Record Holder Mr. Vimal Daga.'),
 Document(metadata={'source': 'ARTH.csv', 'row': 1}, page_content='Customer Question: Who shoul
oin ARTH Program?\nResponse: ARTH is mainly focused on pursuing technical students With ARTH, y
can :\r\n\uf0b7 Upgrade your technical skills to different level, implementing most\r\ndemanded
chnologies in the industry today and near future.\r\n\uf0b7 Convert yourself from just a basic
ineer to a totally most-demanded\r\nand skilled engineer of the industry.\r\n\uf0b7 Get assured
out your placements and technical skills about high-level\r\ntechnologies after getting trained
der the World Record Holder Mr.\r\nVimal Daga.'),
 Document(metadata={'source': 'ARTH.csv', 'row': 0}, page_content='Customer Question: What is A
?\nResponse: ARTH is a Hindi language word अर्थ which means "meaning, significance.\r\nintention,
im, substance". Talking about LinuxWorld's Biggest Initiative,\r\n"ARTH - The School of Technol
es" which is a life-changing 12 Months Online Live Technical Program to mould and prepare the t
nical mindset and skills to get valued asidol in the IT industry. ARTH will include everything
```

- To Retrieving Similar Responses Based on Customer Queries we create one Function

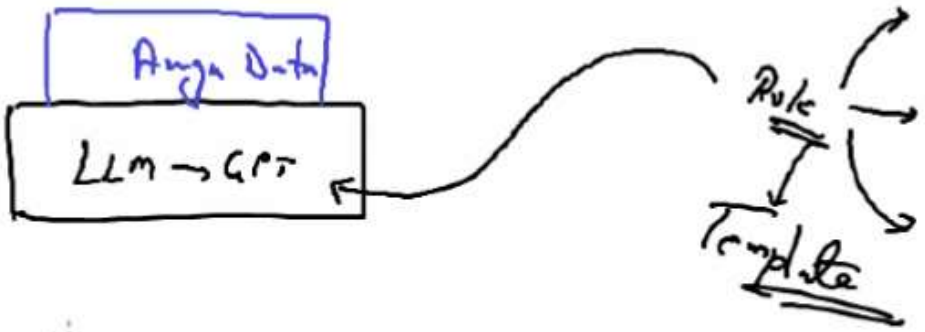
```
In [62]:
def retrieve_data(query):
    similar_response = db.similarity_search(query)
    entire_data = [ doc.page_content for doc in similar_response ]
    return entire_data

In [63]: retrieve_data("what is the fees of ARTH")
```

- The retrieve_data() function takes a customer's query (message) and performs a **similarity search** in the FAISS database.
- It returns a list of best practices that are similar to the customer's query.
- **similar_response = db.similarity_search(query)**: This searches the FAISS store using the embeddings, returning documents that are close in vector space to the query.
- **entire_data = [doc.page_content for doc in similar_response]**: The content of the retrieved documents is extracted and returned as a list.
- Now Setting up the GPT-4 Language Model

```
In [26]: from langchain.chat_models import ChatOpenAI

In [28]: llm = ChatOpenAI(model="gpt-4o" , openai_api_key=openAIApi_key , temperature=0)
```

- You are using **ChatOpenAI** to initialize the **GPT-4** model for generating text.

- **temperature=0** ensures that the model's responses are more deterministic, meaning it will give more consistent answers. A higher temperature would make the model more creative and variable in its responses.
- Now Creating a Custom Response Template
- the use of prompt templates and dynamic prompts in language models (LM), specifically within the context of creating a customized assistant or chatbot
- The user wants to create an assistant (LM) that behaves according to a specific role (e.g., engineer, math teacher, singer) and provides responses based on internal company standards or templates



- The approach includes creating a fixed structure (template) for the responses, ensuring consistency in tone and style, as well as embedding company-specific practices.
- The responses can be made dynamic by passing different variables to the template. For example, user queries or input can change, and the template adjusts based on these inputs, ensuring the assistant's responses remain relevant and aligned with best practices
- a framework that helps in handling prompts and language model outputs. In LangChain, the concept of prompt templates allows users to pass arguments (like message inputs and best practices) to generate responses dynamically. LangChain automates the process of creating and using dynamic prompt
- The template relies on augmented data (e.g., previous messages, historical customer interactions) to generate responses that mimic senior staff members' tone, argument structure, and overall approach
- We create a below template

```
template = """
You are a world class business development representative.
I will share a prospect's message with you and you will give me the best answer that
I should send to this prospect based on past best practies,
and you will follow ALL of the rules below:

1/ Response should be very similar or even identical to the past best practies,
in terms of length, ton of voice, logical arguments and other details

2/ If the best practice are irrelevant, then try to mimic the style of the best practice to pr

Below is a message I received from the prospect:
{message}

Here is a list of best practies of how we normally respond to prospect in similar scenarios:
{best_practice}

Please write the best response that I should send to this prospect:
"""
```

- This is a **prompt template** that will be used to generate customer responses
- The prompt tells the AI to act as a business development representative and respond to the customer in a manner similar to past best practices
- After that Initialize the Prompt with LangChain

```
In [30]: from langchain.prompts import PromptTemplate

In [32]: prompt = PromptTemplate( template=ARTHtemplate , input_variables=["message" , "best_practice"]

In [33]: prompt
```

- The **PromptTemplate** is used to format the ARTHtemplate and plug in the necessary input variables (message and best_practice).
- It ensures that the customer's message and the retrieved best practices are dynamically inserted into the template when calling the model.
- After that Run the Language Model Chain

```
In [36]: from langchain.chains import LLMChain

In [42]: mychain = LLMChain(llm=myllm , prompt=myprompt)

C:\Users\Vimal Daga\AppData\Local\Temp\ipykernel_20456\3255504829.py:1: LangChainDeprecationWarnin
g: The class `LLMChain` was deprecated in LangChain 0.1.17 and will be removed in 1.0. Use :meth:`
~RunnableSequence, e.g., `prompt | llm`` instead.
  mychain = LLMChain(llm=myllm , prompt=myprompt)
```

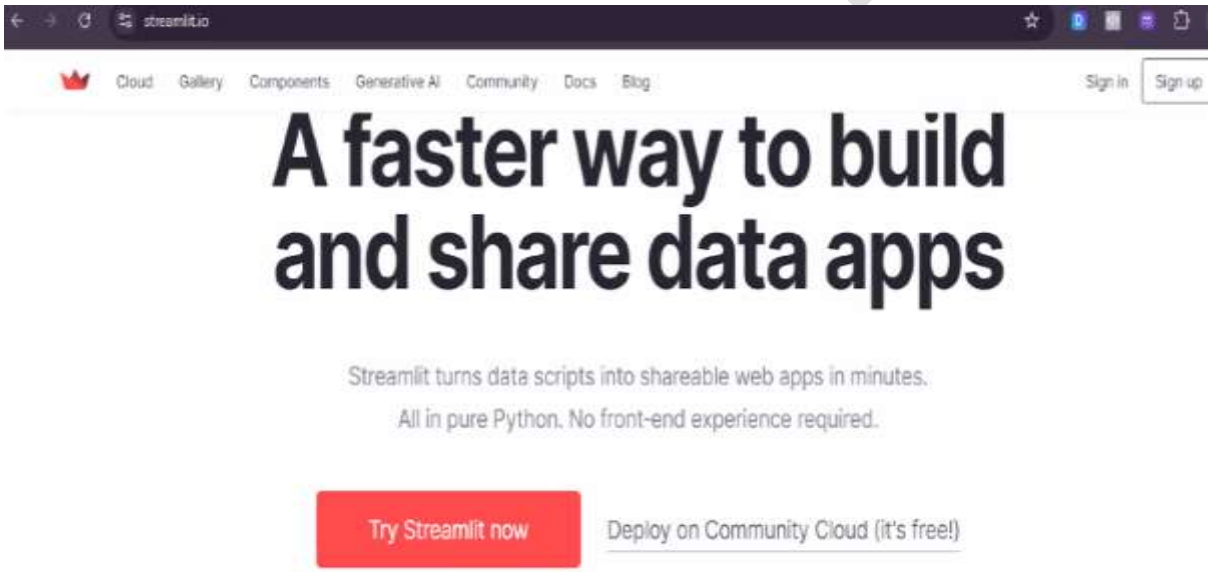
- **LLMChain** is a LangChain construct that ties the GPT-4 model (myllm) with the prompt template (myprompt).
- It runs the language model with the provided inputs.
- Now we Generating the Response using RAG (Retrieve and Generate)

```
In [64]:
def ai_response(message):
    best_practice = retrieve_data(message)
    response = mychain.run(message=message , best_practice=best_practice)
    return response
```

```
In [66]:
ai_response("what is the fees of ARTH program in dollars")
```

```
Out[66]: 'Response: The fees for the ARTH program are approximately 4000 rupees per month. If you need furt
her assistance or have more questions, feel free to reach out!'
```

- **ai_RAG_response()** is the core function that ties everything together.
- It first retrieves the best practices using retrieve_data(), then it calls the LLM chain (mychain.run()) to generate a response.
- The function returns the AI-generated response based on the customer's message and the best practices.
- Now our model is ready to use but we need an interface for that we use **Streamlit**



- To App Creation with Streamlit
- Streamlit can be used to create a web interface for applications
- how to set up a **Streamlit** project and first install the necessary libraries using pip install streamlit

```
Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~/Documents/GenAIOps_Training_2024
$ pip install streamlit
```

- Use Streamlit's predefined functions to display messages and icons on the page, along with customizing the user interface based on inputs
- After that import streamlit
- And use pre-created code for the interface
- Building a Streamlit App

In [75]:
```python
import streamlit as st
```

In [76]:
```python
# 5. Build an app with streamlit
def main():
    st.set_page_config(
        page_title="ARTH Customer response by AI", page_icon=":bird:")

    st.header("Customer response generator :bird:")
    message = st.text_area("my customer message")

    if message:
        st.write("Generating AI ARTH  message...")

        result = ai_RAG_response(message)

        st.info(result)


if __name__ == '__main__':
    main()
```

- **Streamlit** is used to create a simple web app
- **st.set_page_config()**: Sets the page title and icon for the app
- **st.header()**: Displays a title in the app
- **st.text_area()**: Creates a text box for the user to input a customer message
- **if message**: When the user enters a message, it triggers the generation of a response using ai_RAG_response()
- **st.write()**: Displays a loading message while the AI generates the response
- **st.info(result)**: Shows the final response generated by the AI
- But if you want to run this streamlit code for that we need to run it from the Command line
- For that we create a file paste all code into the file and run from the CMD

```
Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~/Documents/GenAIOps_Training_2024
$ notepad ARTH_AI.py

Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~/Documents/GenAIOps_Training_2024
$
```

```python
from langchain_community.document_loaders.csv_loader import CSVLoader

loader = CSVLoader(file_path="ARTH.csv", encoding='utf-8',  csv_args={
            'delimiter': '|',
        })

dataset = loader.load()

from dotenv import load_dotenv
load_dotenv()
import os
openAIApi_key = os.getenv("llm_api_key")


from langchain.embeddings.openai import OpenAIEmbeddings
myembedModel = OpenAIEmbeddings( openai_api_key=openAIApi_key)
```

```
from langchain.vectorstores import FAISS
db = FAISS.from_documents( documents=dataset , embedding=myembedModel )


def retrieve_data(query):
    similar_response = db.similarity_search(query)
    entire_data = [ doc.page_content for doc in similar_response ]
    return entire_data
```

```
from langchain.chat_models import ChatOpenAI
myllm = ChatOpenAI(model="gpt-4" , openai_api_key=openAIApi_key , temperature=0)
```

```
ARTHtemplate = """
You are a world class business development representative for ARTH Program at LinuxWorld informatics pvt ltd.
I will share a prospect's message with you and you will give me the best answer that
I should send to this prospect based on past best practies,
and you will follow ALL of the rules below:

1/ Response should be very similar or even identical to the past best practies,
in terms of length, ton of voice, logical arguments and other details

2/ If the best practice are irrelevant, then try to mimic the style of the best practice to prospect's message

Below is a message I received from the prospect:
{message}

Here is a list of best practies of how we normally respond to prospect in similar scenarios:
{best_practice}

Please write the best response that I should send to this prospect:
"""
```

```
from langchain.prompts import PromptTemplate
myprompt = PromptTemplate( template=ARTHtemplate , input_variables=["message" , "best_practice"] )


from langchain.chains import LLMChain
mychain = LLMChain(llm=myllm , prompt=myprompt)


def ai_RAG_response(message):
    best_practice = retrieve_data(message)
    response = mychain.run(message=message , best_practice=best_practice)
    return response


import streamlit as st
```

```
# 5. Build an app with streamlit
def main():
    st.set_page_config(
        page_title="ARTH Customer response by AI", page_icon=":bird:")

    st.header("Customer response generator :bird:")
    message = st.text_area("my customer message")

    if message:
        st.write("Generating AI ARTH  message...")

        result = ai_RAG_response(message)

        st.info(result)


if __name__ == '__main__':
    main()
```

- After that run the file with the below command

```
Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~/Documents/GenAIOps_Training_2024
$ streamlit.exe  run  ARTH_AI.py

  You can now view your Streamlit app in your browser.

  Local URL: http://localhost:8502
  Network URL: http://192.168.1.4:8502
```

- Now we create an AI application