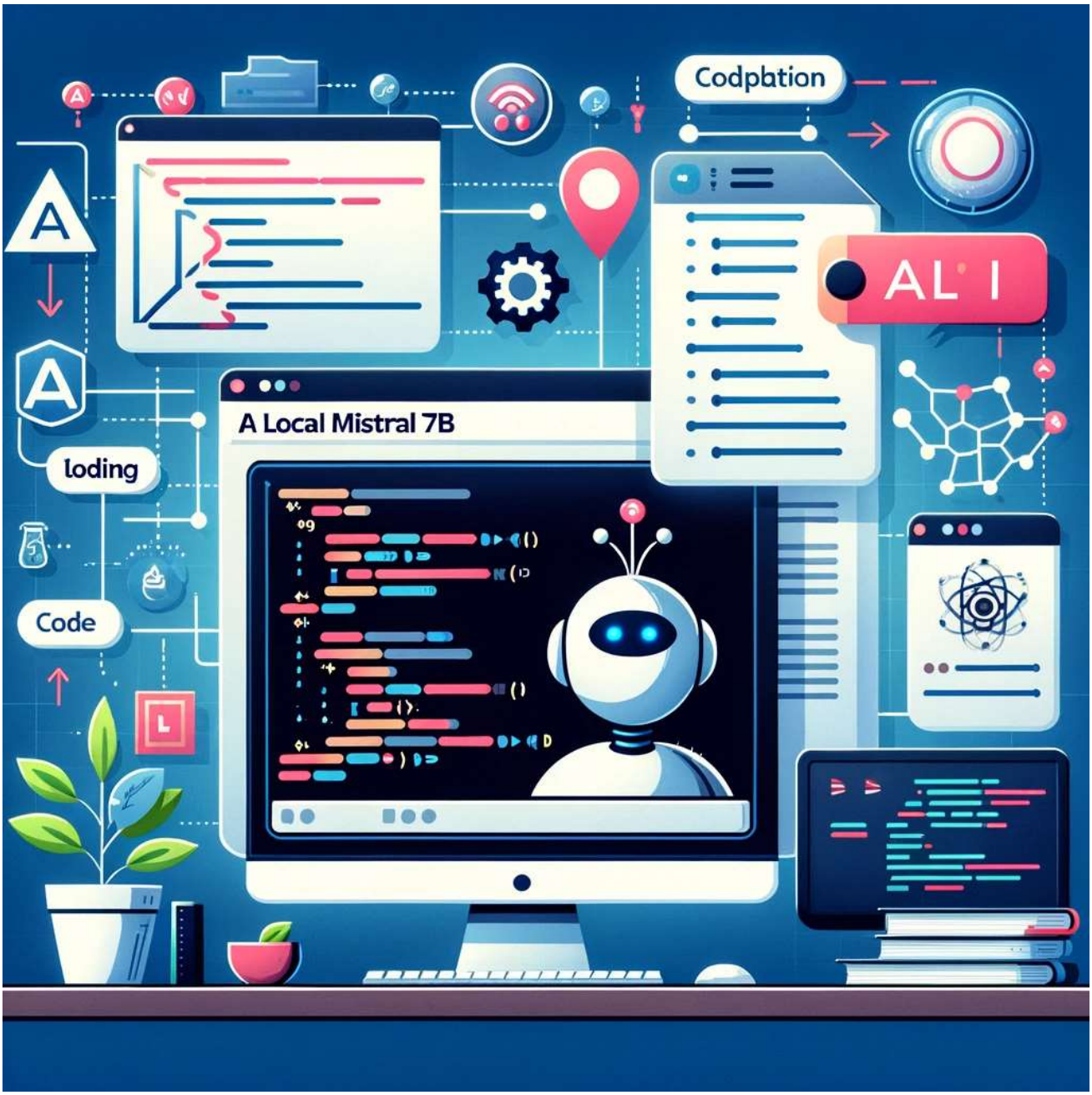

Guide to build your local Mistral 7B chatbot using Streamlit

Made by : Mohamed Ashour

Date : December 2023

About the author

The author, with a robust background spanning nearly seven years in the construction industry, brings a unique blend of expertise and academic achievement to the table. Holding a bachelor's degree in construction and engineering management, he has laid a solid foundation in the technical aspects of the industry. Further elevating his qualifications, the author pursued and attained a Master of Science degree in Commercial Management and Quantity Surveying, a discipline that marries the technicalities of construction with the nuances of business management.

Complementing their construction-centric education, the author also delved into the realm of data analytics, acquiring a degree that marks a significant pivot in their career trajectory. This educational journey is crowned by their chartered status from two prestigious institutions: the Royal Institution of Chartered Surveyor and the British Computer Society, reflecting a rare confluence of construction expertise and computational acumen.

In recent years, the author has shifted their focus towards the data world, dedicating the past three years to working intensively in this domain. Their interest particularly lies in the deployment of Artificial Intelligence (AI) and Machine Learning (ML) within the construction industry, a sector ripe for digital transformation. Recognizing the potential of AI and ML to revolutionize traditional practices, the author has been at the forefront of integrating these technologies into construction processes, aiming to enhance efficiency, accuracy, and overall project management.

One of the author's notable contributions is the development of a series of chatbots using open-source large language models. These chatbots represent a significant innovation, leveraging the power of AI to streamline communication, automate routine tasks, and provide intelligent assistance in various construction-related scenarios. The author's work in this area not only showcases their technical prowess but also their commitment to driving the construction industry forward through the adoption of cutting-edge technologies. Their unique blend of construction knowledge, data analytics expertise, and passion for AI and ML positions them as a visionary figure, poised to make a lasting impact on the industry.

You can reach out to me on my LinkedIn page: https://www.linkedin.com/in/mohamedashour-0727/ or via email on mohamed_ashour@apcmasterypath.co.uk.

Disclaimer

You can contact the author of this document via email on mohamed_ashour@apcmasterypath.co.uk & mo_ashour1@outlook.com.

Table of Contents

Executive Summary

This guide encapsulates some major key points regarding the process of building a local chatbot:

1. **Building a Local Chatbot**: The process involves installing necessary packages, loading documents, splitting text into chunks, creating embeddings, vector stores, and a Large Language Model (LLM). The chatbot is then integrated into a Streamlit app for user interaction.
2. **Different LLMs Available**: Various LLMs like GPT, BERT, ERNIE, T5, XLNet, and ChatGPT are available, each with unique strengths in language processing and suitable for different aspects of chatbot development.
3. **Use Cases in Construction Industry**: Local chatbots can enhance safety training, operational efficiency, communication, customer service, data analysis, and training in the construction industry.
4. **Advantages Over Cloud-Based Chatbots**: Local chatbots offer benefits such as enhanced data privacy and security, complete data control, reduced latency, operational independence from the internet, and cost-effectiveness.
5. **Limitations of Local Chatbots**: Challenges include handling complex issues, absence of human connection, resource intensity, scalability issues, limited language and context understanding, dependency on local infrastructure, and limited learning capabilities.

This guide provides a comprehensive overview of the development, application, advantages, and limitations of local chatbots, particularly in the context of the construction industry and their comparison with cloud-based alternatives.

Use cases of building a local chatbot

Local chatbots, tailored to specific needs and environments, can significantly benefit the construction industry in various ways:

1. **Safety Training and Hazard Awareness**: Chatbots can be instrumental in enhancing hazard awareness among construction workers. They can deliver safety training, provide reminders about safety protocols, and answer queries related to occupational safety and health practices [13].
2. **Operational Efficiency**: In the construction industry, chatbots can streamline operations by assisting in scheduling, resource allocation, and project management. They can provide quick access to project information, updates, and reports, thereby improving overall efficiency [14].
3. **Communication and Coordination**: Chatbots can facilitate better communication and coordination among various stakeholders in a construction project. They can serve as a central point for sharing updates, changes in plans, or urgent notifications, ensuring that all parties are informed in real-time [15].

4. **Customer Service and Engagement**: For construction companies dealing with clients, chatbots can handle inquiries, provide project updates, and address concerns, enhancing customer engagement and satisfaction.[16]
5. **Data Collection and Analysis**: Chatbots can collect data from various sources on the construction site, analyze it for insights, and provide recommendations for process improvements or risk mitigation.[17]
6. **Training and Onboarding**: New workers can be onboarded and trained using chatbots, which can provide interactive learning experiences and answer common queries about construction processes and safety practices. [18]

By integrating chatbots into their operations, construction companies can leverage these technologies to improve safety, efficiency, and communication, ultimately leading to more successful project outcomes.

Different LLMs available

Large Language Models (LLMs) have become increasingly sophisticated and varied, offering a range of capabilities for different applications, including chatbots. Some of the notable LLMs available are:

1. **GPT (Generative Pre-trained Transformer)**: Developed by OpenAI, GPT models (like GPT-3) are known for their ability to generate human-like text and are widely used in chatbots for their conversational abilities.[19]
2. **BERT (Bidirectional Encoder Representations from Transformers)**: Developed by Google, BERT excels in understanding the context of a word in a sentence, making it useful for tasks that require a deep understanding of language context. [20]
3. **ERNIE (Enhanced Representation through kNowledge Integration)**: Baidu's ERNIE is designed to better understand the nuances of language by incorporating knowledge graphs, enhancing its performance in language understanding tasks. [21]
4. **T5 (Text-To-Text Transfer Transformer)**: T5, also from Google, treats every language problem as a text-to-text problem, providing a versatile framework for various NLP tasks. [22]
5. **XLNet**: This is an extension of the Transformer model, which outperforms BERT in several benchmarks by learning bidirectional contexts and managing longer sequences more effectively. [23]
6. **ChatGPT**: A variant of the GPT model, specifically fine-tuned for conversational applications, making it ideal for chatbot implementations. [24]
7. **Bing Chat and Bard**: These are examples of chatbots powered by large language models, showcasing the practical application of LLMs in conversational AI. [24]

Each of these models has its strengths and is suitable for different aspects of chatbot development, from understanding user queries to generating coherent and contextually relevant responses.

Advantages of local chatbots

Local chatbots offer several distinct advantages over their cloud-based counterparts:

1. **Data Privacy and Security**: Local chatbots process and store data on-premises, significantly reducing the risk of data exposure and breaches. This is particularly important for sensitive information, as it minimizes the risk of exposure during transmission [1].
2. **Complete Data Control**: With in-house AI chatbots, organizations have full control over their data. This control extends to data security, usage, and compliance with regulatory requirements, offering a level of customization and security that cloud-based solutions may not provide [2].
3. **Reduced Latency**: Since local chatbots process data internally, they often have faster response times compared to cloud-based chatbots, where data needs to be sent to and from a remote server.[3]
4. **Operational Without Internet**: Local chatbots can function effectively without an internet connection, ensuring continuous operation even in environments with poor or no internet connectivity.[4]
5. **Cost-Effectiveness**: In the long run, local hosting can be more cost-effective than cloud hosting, especially for businesses with existing infrastructure capable of supporting a local chatbot [5].
6. **Customization and Integration**: Local chatbots can be highly customized to fit specific organizational needs and can be more easily integrated with existing in-house systems and databases.[6]

Limitations of local chatbots

Local chatbots offer several distinct advantages over their cloud-based counterparts:

1. **Data Privacy and Security**: Local chatbots process and store data on-premises, significantly reducing the risk of data exposure and breaches. This is particularly important for sensitive information, as it minimizes the risk of exposure during transmission [7].
2. **Complete Data Control**: With in-house AI chatbots, organizations have full control over their data. This control extends to data security, usage, and compliance with regulatory requirements, offering a level of customization and security that cloud-based solutions may not provide [8].
3. **Reduced Latency**: Since local chatbots process data internally, they often have faster response times compared to cloud-based chatbots, where data needs to be sent to and from a remote server.[9]
4. **Operational Without Internet**: Local chatbots can function effectively without an internet connection, ensuring continuous operation even in environments with poor or no internet connectivity.[10

5. **Cost-Effectiveness**: In the long run, local hosting can be more cost-effective than cloud hosting, especially for businesses with existing infrastructure capable of supporting a local chatbot [11].
6. **Customization and Integration**: Local chatbots can be highly customized to fit specific organizational needs and can be more easily integrated with existing in-house systems and databases.[12]

Building Process of a local chatbot

Building a local chatbot involves several key steps, leveraging technologies like OpenAI's LangChain and Streamlit. Here's an overview of the process:

1. **Installation of Required Packages**: Begin by installing necessary Python packages such as **langchain**, **torch**, **streamlit**, and others that are essential for chatbot development [25].
2. **Document Loading and Processing**: Load and process text documents, which can be done using the LangChain library. This step is crucial for the chatbot to have a knowledge base to draw from when answering questions [26].
3. **Vectorization and Storage**: Utilize OpenAIEmbeddings and FAISS from the LangChain library to vectorize and store text documents. This step is essential for efficient retrieval of information and response generation [26].
4. **Building the Chatbot Interface**: Streamlit is used to create a user-friendly interface for the chatbot. This allows users to interact with the chatbot in a web-based application [27].
5. **Integrating LLM (Large Language Model)**: Incorporate an LLM, such as a GPT variant or HuggingChat API, to power the chatbot. This model is responsible for understanding user queries and generating appropriate responses [28].
6. **Customization and Testing**: Customize the chatbot according to specific needs, such as a healthcare focus, and thoroughly test it to ensure accurate and relevant responses [30].
7. **Deployment**: Finally, deploy the chatbot locally, enabling it to operate independently of cloud services and offering enhanced privacy and data security.

This process highlights the integration of advanced NLP tools and frameworks to build a sophisticated local chatbot capable of handling various user queries effectively.

Complete code explanation

This section about the full explanation for the code used to build a local chatbot by deploying Mistral 7B Large Language Model under Langchain while utilising Streamlit service for the front end.

For clarity, the reader would have to view this section side by side with Appendix 2.

The provided code outlines the process of building a chatbot using Python libraries such as LangChain, Streamlit, and others. Here's a breakdown of the code:

Package Installation

- **Package Installation**: The code begins with installing essential Python libraries. **langchain** for NLP operations, **torch** and **accelerate** for machine learning computations, **sentence_transformers** for pre-trained language models, **streamlit** and **streamlit_chat** for building and integrating the chatbot into a web application, **faiss-cpu** for efficient similarity search, **tiktoken** and **huggingface-hub** for additional NLP functionalities, **pypdf** for PDF processing, and **ctransformers** for transformer models.

Importing Libraries

- **Importing Libraries**: Essential libraries are imported for various functionalities: Streamlit for the web app, LangChain for NLP tasks, and others for specific roles like document loading, text splitting, embedding creation, and memory management.

Document Loading Function

- **Load Documents**: **load_documents** function uses **DirectoryLoader** to load PDF documents from a specified directory. This function is crucial for providing the chatbot with a knowledge base.

Text Splitting Function

- **Split Text into Chunks**: The **split_text_into_chunks** function breaks down the loaded documents into smaller text chunks, making them manageable for processing.

Embedding Creation Function

- **Create Embeddings**: **create_embdeddings** function generates embeddings (vector representations) of text chunks using **HuggingFaceEmbeddings**, which is vital for understanding and processing the text data.

Vector Store Creation Function

- **Create Vector Store**: **create_vector_store** function builds a vector store from the embeddings using FAISS, allowing efficient retrieval of information.

LLM Model Creation Function

- **Create LLMS Model**: **create_llms_model** initializes a Large Language Model (LLM) using **CTransformers**, which is the core for generating chatbot responses.

Streamlit App Initialization

- **Initialize Streamlit App**: The Streamlit app is set up with titles and styles, serving as the user interface for the chatbot.

Loading Documents and Processing

- **Processing Documents**: The chatbot loads and processes documents, splits them into text chunks, creates embeddings, and sets up the vector store.

Conversation History and Memory

- **Initialize Conversation History and Memory**: The code initializes conversation history and sets up a memory for the chatbot using **ConversationBufferMemory**.

Conversational Chain Creation

- **Create Conversational Chain**: **ConversationalRetrievalChain** is created, linking the LLM, retriever, and memory to form the backbone of the chatbot's conversational capabilities.

Chat Function Definition

- **Define Chat Function**: **conversation_chat** function is defined to handle user queries and generate responses using the conversational chain.

Streamlit Interface for Chat

- **Display Chat History**: The Streamlit interface displays the chat history and allows users to input queries and receive responses.

This code represents a comprehensive approach to building a sophisticated chatbot capable of processing documents, understanding queries, and providing relevant responses in a user-friendly web application.

Final Thoughts

Based on what was showed and explained above and based on my experience with this initiative, these are my final remarks:

- The process of building a chatbot with a graphical user interface is becoming easier day by day. We now have free services such as streamlit and chainlit.
- The chatbot relies heavily on the specifications of the computer used and it does take a minute or so for a response.
- Using Mistral is quite efficient with PDFs as it chunks the text effectively and provides a very accurate answer.
- The chatbots that can be used right now rely heavily on having text documents with the text being mapped in the shape of a paragraph. Chatbots are not yet effective with diagrams, images, tables and calculations.
- Building your own chatbot can be done in a matter of hours without being an AI expert, however the process of optimization and commercialisation will have to be done by an expert.
- Construction companies will have to invest heavily in infrastructure in order to cope with the amazing technological advancements that are being developed every day.

Appendix 1 - Complete process for building a local chatbot

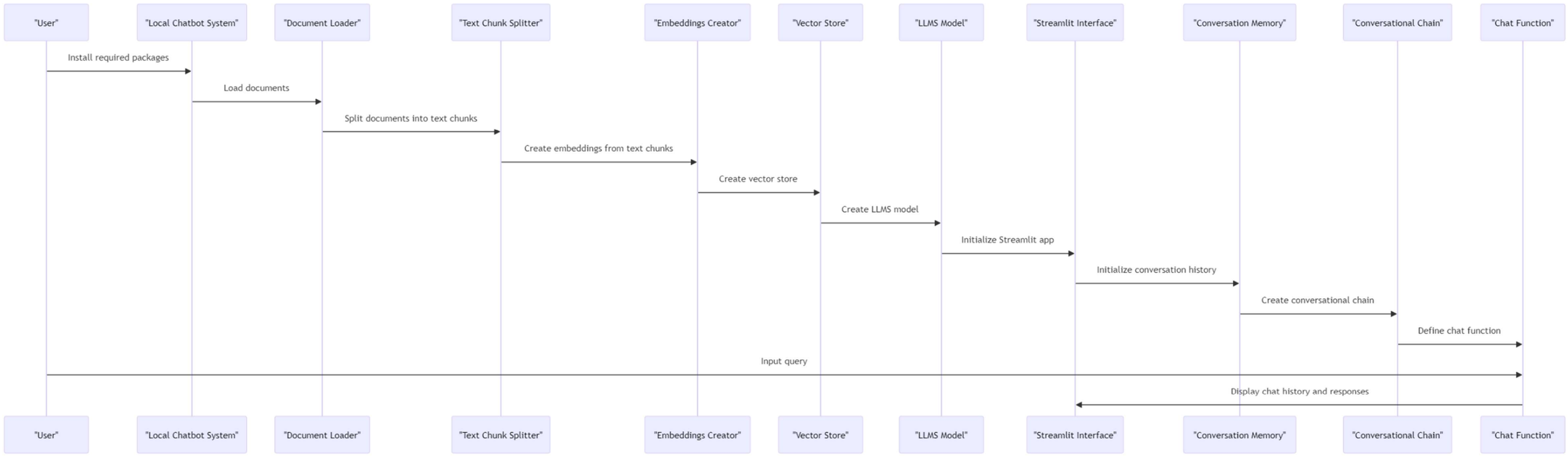

Figure 1: Complete process of building a local chatbot

Appendix 2 - Mistral 7B Chatbot complete code

```
#Install the required packages
##Pip install langchain
##Pip install torch
##Pip install accelerate
##Pip install sentence_transformers
##Pip install streamlit
##Pip install streamlit_chat
##Pip install faiss-cpu
##Pip install tiktoken
##Pip install huggingface-hub
##Pip install pypdf
##Pip install ctransformers

#Import required libraries
import streamlit as st
from streamlit_chat import message
from langchain.chains import ConversationalRetrievalChain
from langchain.document_loaders import PyPDFLoader, DirectoryLoader
from langchain.embeddings import HuggingFaceEmbeddings
from langchain.llms import CTransformers
from langchain.text_splitter import RecursiveCharacterTextSplitter
from langchain.vectorstores import FAISS
from langchain.memory import ConversationBufferMemory

#Create a function to load docuemnts

def load_documents():
    loader=DirectoryLoader('data/',glob="*.pdf",loader_cls=PyPDFLoader)
    documents= loader.load()
    return documents

#Function to split text into chunks
def split_text_into_chunks(documents):
    text_splitter= RecursiveCharacterTextSplitter(chunk_size=500,
chunk_overlap=50)
    text_chunks=text_splitter.split_documents(documents)
    return text_chunks

#Function to create embdeddings
def create_embdeddings():
    embeddings = HuggingFaceEmbeddings(model_name="sentence-transformers/all-
MiniLM-L6-v2",model_kwargs={'device':"cpu"})
    return embeddings
```

```
#Function to create vector store
def create_vector_store(text_chunks,embeddings):
    vector_store=FAISS.from_documents(text_chunks,embeddings)
    return vector_store

#Function to create LLMS Model
def create_llms_model():
    llm=CTransformers(model="TheBloke/Mistral-7B-Instruct-v0.1-GGUF",
                      config={'max_new_tokens':128,
                              'temperature':0.01},
                              streaming=True)
    return llm

#Initialize Streamlit app
st.title("Job Interview Prep Chatbot")
st.title("Personalised Job Success Friend")
st.markdown('<style>h1{color:orange;text-
align:center;}</style>',unsafe_allow_html=True)
st.subheader("Get your desired Job !")
st.markdown('<style>h3{color:pink;text-
align:center;}</style>',unsafe_allow_html=True)

#Loading of documents
documents=load_documents()

#Split text into chunks
text_chunks=split_text_into_chunks(documents)

#Create embeddings
embeddings=create_embdeddings()

#Create Vector Store
vector_store=create_vector_store(text_chunks,embeddings)

#Create LLMS model
llm=create_llms_model()

#Initialize the conversation history
if 'history' not in st.session_state:
    st.session_state['history'] = []

if 'generated' not in st.session_state:
    st.session_state['generated'] = ["Hello! Ask me anything about your dream
job!"]

if 'past' not in st.session_state:
    st.session_state['past'] = ["Hey! Let's have a fruitful chat"]
```

```
#Create memory
memory=
ConversationBufferMemory(memory_key="chat_history",return_messages=True)

#Create chain
chain = ConversationalRetrievalChain.from_llm(llm=llm,chain_type='stuff',
                                              retriever=vector_store.as_retrie
ver(search_kwargs={"k":2}),
                                              memory=memory)

#Define chat function
def conversation_chat(query):
    result=chain({"question":query,"chat_history":st.session_state['history']}
)
    st.session_state['history'].append((query,result["answer"]))
    return result["answer"]

#Display chat history
reply_container=st.container()
container=st.container()

with container:
    with st.form(key="my form",clear_on_submit=True):
        user_input=st.text_input("Question:",placeholder="Ask about your Job
Interview", key='input')
        submit_button=st.form_submit_button(label='send')

    if submit_button and user_input:
        output= conversation_chat(user_input)
        st.session_state['past'].append(user_input)
        st.session_state['generated'].append(output)

if st.session_state['generated']:
    with reply_container:
        for i in range(len(st.session_state['generated'])):
            message(st.session_state["past"][i],is_user=True,key=str(i)+'_user
',avatar_style="thumbs")
            message(st.session_state["generated"][i],key=str(i),avatar_style="
fun-emoji")
```

References

1. [medium.com - Creating Local Chatbots with GPT4All and LangChain for ...](https://medium.com/scrapehero/securing-your-conversations-creating-local-chatbots-with-gpt4all-and-langchain-for-data-privacy-1dfd211b43e8)
2. [linkedin.com - The Advantages of In-House AI Chatbots over Cloud ...](https://www.linkedin.com/pulse/advantages-in-house-ai-chatbots-over-cloud-based-maxwell-duchaine)
3. [researchgate.net - Comparative study of cloud platforms to develop a Chatbot](https://www.researchgate.net/publication/317865587_Comparative_study_of_cloud_platforms_to_develop_a_Chatbot)
4. [chatbotslife.com - Pros and Cons of using cloud platforms for building chatbots](https://chatbotslife.com/pros-and-cons-of-using-cloud-platforms-for-building-chatbots-9d69d6d6e432)
5. [mdpi.com - A Literature Survey of Recent Advances in Chatbots](https://www.mdpi.com/2078-2489/13/1/41)
6. [indiecreatorhub.com - Unleashing the Power of Chatbots: The Game-changer for ...](https://www.indiecreatorhub.com/unleashing-the-power-of-chatbots-the-game-changer-for-streamers/)
7. [medium.com - Creating Local Chatbots with GPT4All and LangChain for ...](https://medium.com/scrapehero/securing-your-conversations-creating-local-chatbots-with-gpt4all-and-langchain-for-data-privacy-1dfd211b43e8)
8. [linkedin.com - The Advantages of In-House AI Chatbots over Cloud ...](https://www.linkedin.com/pulse/advantages-in-house-ai-chatbots-over-cloud-based-maxwell-duchaine)
9. [researchgate.net - Comparative study of cloud platforms to develop a Chatbot](https://www.researchgate.net/publication/317865587_Comparative_study_of_cloud_platforms_to_develop_a_Chatbot)
10. [chatbotslife.com - Pros and Cons of using cloud platforms for building chatbots](https://chatbotslife.com/pros-and-cons-of-using-cloud-platforms-for-building-chatbots-9d69d6d6e432)
11. [mdpi.com - A Literature Survey of Recent Advances in Chatbots](https://www.mdpi.com/2078-2489/13/1/41)
12. [indiecreatorhub.com - Unleashing the Power of Chatbots: The Game-changer for ...](https://www.indiecreatorhub.com/unleashing-the-power-of-chatbots-the-game-changer-for-streamers/)
13. [frontiersin.org - Can a chatbot enhance hazard awareness in the ...](https://www.frontiersin.org/articles/10.3389/fpubh.2022.993700)
14. [linkedin.com - Revolutionizing Construction Operations: Unleashing the ...](https://www.linkedin.com/pulse/revolutionizing-construction-operations-unleashing-power-aitzaz-ahmed)

15. [researchgate.net - (PDF) A survey on construction and enhancement methods ...](https://www.researchgate.net/publication/336016707_A_survey_on_construction_and_enhancement_methods_in_service_chatbots_design)
16. [saifety.ai - AI Chatbots for Construction Industry: Revolutionary Stage](https://www.saifety.ai/how-do-ai-chatbots-improve-efficiency-on-construction-sites/)
17. [mindy-support.com - Using Chatbots to Increase Construction Efficiency](https://mindy-support.com/news-post/using-chatbots-to-increase-construction-efficiency/)
18. [searchenginejournal.com - The Future Of Chatbots: Use Cases & Opportunities You ...](https://www.searchenginejournal.com/future-of-chatbots/278595/)
19. [botpress.com - 13 Best Open Source Chatbot Platforms to Use in 2023](https://botpress.com/blog/open-source-chatbots)
20. [cloud.google.com - AI Chatbot](https://cloud.google.com/use-cases/ai-chatbot)
21. [mdpi.com - A Literature Survey of Recent Advances in Chatbots](https://www.mdpi.com/2078-2489/13/1/41)
22. [linkedin.com - The Advantages of In-House AI Chatbots over Cloud ...](https://www.linkedin.com/pulse/advantages-in-house-ai-chatbots-over-cloud-based-maxwell-duchaine)
23. [blog.hubspot.com - AI Chatbots: Our Top 18 Picks for 2023](https://blog.hubspot.com/marketing/best-ai-chatbot)
24. [sciencedirect.com - Chatbots: History, technology, and applications](https://www.sciencedirect.com/science/article/pii/S2666827020300062)
25. [youtube.com - Use OpenAI LangChain and Streamlit to create custom chatbot](https://www.youtube.com/watch?v=hkV-4NpMibw)
26. [linkedin.com - How to Use the Streamlit App to Build a Chatbot that Can ...](https://www.linkedin.com/pulse/how-use-streamlit-app-build-chatbot-can-respond-questions-shah)
27. [youtube.com - How To Build ChatGPT Clone With Streamlit and LangChain](https://www.youtube.com/watch?v=1vQdFx46FcY)
28. [streamlit.io - How to build an LLM-powered ChatBot with Streamlit](https://blog.streamlit.io/how-to-build-an-llm-powered-chatbot-with-streamlit/)
29. [reddit.com - Creating Chatbot using langchain and streamlit](https://www.reddit.com/r/learnpython/comments/16wvovf/creating_chatbot_using_langchain_and_streamlit/)
30. [youtube.com - Build Streamlit HealthCare Chatbot with Llama 2&Langchain](https://www.youtube.com/watch?v=XNmFIkViEBU)