Retrieval-Augmented Generation (RAG) Chat Application

Objective

The objective of this assignment is to build a RAG-based chatbot capable of accurate and contextually relevant responses, using Python and LangChain tools.

Code Documentation

Module-Level Documentation

Module Name: rag_chatbot

Description: Implements a Retrieval-Augmented Generation (RAG) pipeline using LangChain, FAISS, and Hugging Face.

Dependencies:

- transformers
- langchain
- faiss-cpu
- sentence-transformers
- huggingface-hub

Class and Function Documentation

Hugging Face Login

```
from huggingface_hub import login
def login_to_hugging_face(api_token):
    """
    Logs into the Hugging Face platform using the provided API token.
Args:
        api_token (str): API token for authentication.
 Returns:
        None
    """
    login(token=api_token)
    print("Logged in to Hugging Face successfully!")
```

Document Loader and Splitter

```
from langchain.document_loaders import TextLoader

from langchain.text_splitter import RecursiveCharacterTextSplitter


def load_and_split_documents(file_path, chunk_size=500, chunk_overlap=100):
    """
    Loads a text file and splits its content into chunks.
Args:
        file_path (str): Path to the document file.
        chunk_size (int): Size of each chunk in characters.
        chunk_overlap (int): Overlap between consecutive chunks.
 Returns:
        list: A list of split document chunks.
    """
    loader = TextLoader(file_path)
    documents = loader.load()
    text_splitter = RecursiveCharacterTextSplitter(chunk_size = chunk_size,  chunk_overlap =
chunk_overlap)
    return text_splitter.split_documents(documents)
```

Embedding and Vector Store

```
from langchain.vectorstores import FAISS
from langchain.embeddings import HuggingFaceEmbeddings


def create_vector_store(documents, model_name="sentence-transformers/all-MiniLM-L6-v2"):
    """
    Creates a FAISS vector store from a list of documents using Hugging Face embeddings.

    Args:
        documents (list): List of document chunks.
        model_name (str): Name of the Hugging Face embedding model.

    Returns:
        FAISS: A FAISS vector store instance.
    """
    embedding_model = HuggingFaceEmbeddings(model_name=model_name)
    vector_store = FAISS.from_documents(documents, embedding_model)
    return vector_store
```

Deliverables

1. **Python Code**:
    - File Name: rag_assignment.ipynb
    - Includes:
        - Document Loading
        - RAG Pipeline Implementation
        - Hugging Face Authentication
        - Commented and documented code for clarity.
2. **Sample Questions and Responses**:
    - File Name: responses.txt
    - Contents: Sample user questions and corresponding chatbot responses.
3. **GitHub Repository**:
    - Repository Link: GitHub Repository
    - Contents:
        - Code (rag_assignment.ipynb)
        - responses.txt
        - Readme
4. **Optional: Streamlit Deployment**:
    - Hosted Link: Streamlit Application

Installation and Usage

Prerequisites

- Python 3.8 or higher.
- Libraries:
    - langchain
    - faiss
    - pandas
    - streamlit

Steps to Run

1. Clone the GitHub repository:
2. git clone <repository_link>

   cd <repository_name>
3. Install dependencies:

   pip install -r requirements.txt
4. Launch the chatbot (if deployed with Streamlit):

   streamlit run app.py
5. Interact with the chatbot via the web interface

References

- LangChain Documentation: https://langchain.com/
- FAISS Documentation: https://faiss.io/
- Python Documentation: https://docs.python.org/3/