Artificial Intelligence Powered Document Q&A Assistant: Using AI to Interact with Your PDFs

Overview

This project demonstrates how to build an **AI powered Question & Answer (Q&A) assistant** that allows you to upload a PDF document and ask questions about its content — all in natural language.

Using open-source technologies such as **LlamaIndex**, **HuggingFace Transformers**, and **Sentence Transformers**, this tool extracts text from your PDF, understands it, and provides intelligent answers — just like ChatGPT but fully free and customizable.

Install the following essential packages to build our Q&A assistant:

- `llama-index-core`: Core framework for building LlamaIndex apps
- `llama-index-readers-file`: Load and parse PDF files
- `llama-index-vector-stores-chroma`: Store and query document embeddings using ChromaDB
- `llama-index-llms-huggingface`: Access open-source LLMs via HuggingFace
- `python-dotenv`: Manage environment variables securely
- `chromadb`: Embedding store and vector search backend
- `transformers`, `accelerate`: Efficient usage of transformer-based models
- `bitsandbytes`: Optimize model memory usage (e.g., 8-bit quantization)
- `nest_asyncio`: Handle nested asynchronous loops in notebooks
- `llama-parse`: Parse and structure PDF text intelligently

```
!pip install -q llama-index-core \
               llama-index-readers-file \
               llama-index-vector-stores-chroma \
               llama-index-llms-huggingface \
               python-dotenv \
               chromadb \
               transformers \
               accelerate \
               bitsandbytes \
               nest_asyncio \
               llama-parse
!pip install -q --upgrade llama-index
```

Show hidden output

Import essential modules to build and run the Q&A assistant:

- `VectorStoreIndex`, `SimpleDirectoryReader`, `Settings`: Core components from LlamaIndex
- `HuggingFaceLLM`: Hugging Face LLM wrapper for integration with LlamaIndex
- `AutoTokenizer`, `AutoModelForCausalLM`: Tokenizer and model loader from Hugging Face Transformers
- `files`: Module from `google.colab` to upload files interactively
- `os`: For environment and file path handling

```
from llama_index.core import VectorStoreIndex, SimpleDirectoryReader, Settings
from llama_index.llms.huggingface import HuggingFaceLLM
from transformers import AutoTokenizer, AutoModelForCausalLM
from google.colab import files
import os
```

Upload PDF Files:

Use the file upload widget to upload your PDF document(s).
This will allow the assistant to read and analyze the content.

```
uploaded = files.upload()  # Upload your PDF(s) here, e.g., ag-studio.pdf

# List uploaded files to confirm
print("Uploaded files:", list(uploaded.keys()))
```

Show hidden output

Load and Parse the PDF Document:

Use `SimpleDirectoryReader` from LlamaIndex to read the uploaded PDF file. This will extract the text content from the document so it can be indexed and queried.

```
documents = SimpleDirectoryReader(input_files=["ag-studio.pdf"]).load_data()
print(f"Loaded {len(documents)} documents.")
```

Show hidden output

Upload Environment Variables File:

Upload your `.env` file containing any required API keys or configurations. This ensures sensitive credentials (e.g., HuggingFace tokens) are loaded securely.

```
from google.colab import files
uploaded = files.upload()  # Select your `.env` file
```

Show hidden output

Load Environment Variables and Login to Hugging Face:

Load the `.env` file to securely retrieve the Hugging Face token, then authenticate using the `huggingface_hub.login()` function.

```
from dotenv import load_dotenv
from huggingface_hub import login

# Load the .env file
load_dotenv('colab_keys.env')  # e.g., huggingface_api.env

# Get token from env variable
hf_token = os.getenv("HUGGINGFACE_TOKEN_KEY")

# Login to Hugging Face
login(hf_token)
```

Initialize the Hugging Face Language Model:

Configure the `Mistral-7B-Instruct` model using `HuggingFaceLLM` and set it as the global LLM for use with LlamaIndex.

```
llm = HuggingFaceLLM(
    model_name="mistralai/Mistral-7B-Instruct-v0.2",  # Free & powerful model
    tokenizer_name="mistralai/Mistral-7B-Instruct-v0.2",
    device_map="auto"  # Automatically selects GPU if available
)

# Set global LLM settings for LlamaIndex
Settings.llm = llm
```

Show hidden output

Install LlamaIndex HuggingFace Embeddings:

Install the `llama-index-embeddings-huggingface` package to integrate Hugging Face embeddings with LlamaIndex.

```
!pip install -q llama-index-embeddings-huggingface
```

Set the Hugging Face Embedding Model:

Configure the `all-MiniLM-L6-v2` embedding model from Sentence-Transformers for use with LlamaIndex. This model will be used to create document embeddings.

```
from llama_index.embeddings.huggingface import HuggingFaceEmbedding

# Set Hugging Face Embedding model (free and powerful)
embed_model = HuggingFaceEmbedding(model_name="sentence-transformers/all-MiniLM-L6-v2")

# Set global embedding model for LlamaIndex
Settings.embed_model = embed_model
```

Build the Vector Store Index:

Create the index from the loaded documents using `VectorStoreIndex` to store the document embeddings for efficient querying.

```
index = VectorStoreIndex.from_documents(documents)
```

Query the Vector Store Index:

Use the `as_query_engine()` method to convert the index into a query engine, which can answer questions based on the document embeddings. An example query is shown below to retrieve information about the design goals.

```
query_engine = index.as_query_engine()

# Example query
response = query_engine.query("What are the design goals and please give details about them"
print("Response:\n", response)
```

 Show hidden output