Colab Notebook for Resume Analysis with Retrieval-Augmented Generation (RAG)

SimpleRAG_Gemini.ipynb - Colab

Srinivasan
srinigith@gmail.com

Introduction:

This Colab notebook demonstrates a system for asking questions and getting answers about information contained within a PDF resume document. It leverages the power of Retrieval-Augmented Generation (RAG) to provide informative responses that are tailored to the specific content of the resume.

Key Functionalities:

- PDF Parsing: Extracts text from a PDF resume using the PyPDFLoader library.
- Text Embeddings: Generates vector representations of the extracted text using a pre-trained language model (HuggingFaceEmbeddings).
- Document Retrieval: Creates a vector store (FAISS) to efficiently search and retrieve relevant parts of the resume based on the user's query.
- Large Language Model (LLM): Utilizes the Gemini-1.5-flash model from Google GenerativeAI to answer questions in an informative way.
- Retrieval-Augmented Generation (RAG): Combines the retrieval and LLM components. The retriever searches the resume for relevant sections based on the query, and the LLM uses the retrieved information to formulate a comprehensive answer.

Introduction:

This notebook showcases the following steps:

- Package Installation: Installs necessary libraries for PDF processing, text embeddings, vector storage, retrieval, LLM access, and answer generation.
- Document Loading: Loads the target resume PDF document using PyPDFLoader.
- Embedding Creation and Vector Store: Creates text embeddings for the resume content and builds a FAISS vector store for efficient retrieval.
- LLM Configuration: Sets up access to the Gemini-1.5-flash LLM model using a Google API key.
- Retrieval and QA Chain Definition: Defines the retrieval component using the FAISS vector store and the LLM for answer generation.
- Sample Q&A: Demonstrates how to ask various questions about the resume and retrieve answers using the RAG system.
- Optional: Full Resume Text Extraction: Includes code for extracting the complete text from the PDF using PyPDF for reference (unrelated to the RAG system).

```
%pip install 'langchain' 'langchain-community' 'langchain-google-genai'
'langchain-unstructured' 'langchain-huggingface' 'pypdf' 'faiss-gpu'
```

Notebook

```
from langchain_huggingface import HuggingFaceEmbeddings
from langchain.vectorstores import FAISS
from langchain.chains import RetrievalQA
from langchain.document_loaders import TextLoader, PyPDFLoader

import google.generativeai as genai
from langchain_google_genai import (ChatGoogleGenerativeAI, HarmBlockThreshold, HarmCategory,)

import os
from google.colab import userdata

# Load your documents
loader = PyPDFLoader("/content/sample_data/Srini_Resume.pdf")
documents = loader.load()

#Google key for API
g_api_key = userdata.get('GEMINI_API_KEY')
os.environ["GOOGLE_API_KEY"] = g_api_key
```

Notebook

```
# Create embeddings and vector store - free General Text Embeddings model from huggingface:
https://huggingface.co/thenlper/gte-large
embeddings = HuggingFaceEmbeddings(
  model_name="thenlper/gte-large"
)
vectorstore = FAISS.from_documents(documents, embeddings)

# Define LLM and retrieval QA chain
genai.configure(api_key=g_api_key)

#Large language model Gemini
llm = ChatGoogleGenerativeAI(
    model="gemini-1.5-flash",
    safety_settings={
        HarmCategory.HARM_CATEGORY_DANGEROUS_CONTENT: HarmBlockThreshold.BLOCK_NONE,
    },
    #tools=tools, enable_automatic_function_calling=True, system_instruction=tool_instruction
)
```

Notebook

```
#Reriever
retriever = vectorstore.as_retriever()
#Chain for Q&A
qa_chain = RetrievalQA.from_llm(llm, retriever=retriever)

# Ask a question about name
query = "What is the name of the applicant?"
answer = qa_chain.invoke(query)
print(answer)

# Year of experience
query = "What is his professional experience?"
answer = qa_chain.invoke(query)
print("\n")
print(answer)
```

Notebook

```
# Speciality?
query = "What is his most special skill set?"
answer = qa_chain.invoke(query)
print("\n")
print(answer)

# Fit?
query = "I am looking for a full-stack developer, will fit for the same?"
answer = qa_chain.invoke(query)
print("\n")
print(answer)

# Advanced fit?
query = "Is he suitable to handle a software architecture role?"
answer = qa_chain.invoke(query)
print("\n")
print(answer)
```

Notebook

```
# Not fit?
query = "Is he suitable for the finance manager role?"
answer = qa_chain.invoke(query)
print("\n")
print(answer)

# Specific experience?
query = "Does he have any specific client-side scripting skills?"
answer = qa_chain.invoke(query)
print("\n")
print(answer)

# Open question?
query = "Which country owns the football World Cup now?"
answer = qa_chain.invoke(query)
print("\n")
print(answer)
```

Notebook Response (Chat)

{'query': 'What is the name of the applicant?', 'result': "The applicant's name is Srinivas San.\n"}

{'query': 'What is his professional experience?', 'result': 'Srinivas has 19+ years of experience in software development, web and mobile technologies, cloud and clean/microservices architecture, and AI-driven solutions. His experience includes:\n\n* **REACHIN4.AI (Remote, US) - Technical Lead (APR 2023 - Present):** Developed progressive web applications using .NET Core, SQL, and CI/CD pipelines. Integrated ChatGPT 4.0 and OpenAI into client solutions.\n\n* **RENCATE Pvt Ltd, Puducherry - Technical Lead (JAN 2019 - MAR 2023):** Built microservice architecture using .NET Core and SQL for mortgage solutions. Deployed solutions using Azure services (CosmosDB, Storage Account, and Service Bus).\n\n* **iHORSE TECHNOLOGY Pvt Ltd, Puducherry - Technical Lead (SEP 2016 - DEC 2019):** Created web and mobile applications using Node.js, AngularJS, React, React Native, MongoDB, and AWS. Also handled database design and development.\n\n* **SSG CONSULTING Pvt Ltd, Puducherry - Senior Software Engineer (2013 - 2016):** Worked on web application enhancements (mortgage loan) and implemented business logic for databases.\n\n* **EFFINDI TECHNOLOGIES Pvt Ltd, Puducherry - Software Engineer (2011 - 2013):** Developed web and mobile applications and designed and implemented databases.\n\n* **INTEGRAS SOFTWARE SERVICES Pvt Ltd, Puducherry - Senior Programmer (2006 - 2011):** Developed web applications for internal tracking systems (ERP) and handled database business and logic development.\n\n* **SURVAY, Puducherry - Senior Programmer (2005 - 2006):** Designed and developed web applications and web graphics, including database design, development, and implementation.\n\n* **MAXEM INDIA Pvt Ltd, Puducherry - Graphics Designer cum Game Developer (2005 - 2006):** Created graphics designs and game developments, including 2D animations, creative concepts, and character development.\n'}

Notebook Response (Chat)

{'query': 'What is his most special skill set?', 'result': "Based on the provided text, Srinivasan's most specialized skill set appears to be in Machine Learning, Deep Learning, and Generative AI (including experience with Gemini, ChatGPT, OpenAI, and LLAMA3), along with Data Science and ML.Net.\n"}

{'query': 'I am looking for a full-stack developer, will fit for the same?', 'result': 'Based on the provided resume, Srinivas Vasan has extensive experience in many full-stack technologies. He has experience with front-end frameworks (React, Angular, Xamarin, Flutter, Ionic), back-end languages and frameworks (C/C++, C#.Net, VB.Net, ASP, ASP.Net, PHP, Python, ASP.NetCore, .NETCore, Node.js), and databases (SQLServer, MySQL, CosmosDB). His experience includes developing web and mobile applications, and his resume shows projects utilizing both front-end and back-end technologies. Therefore, he likely would fit the requirements for a full-stack developer role.\n'}

{'query': 'Is he suitable to handle a software architecture role?', 'result': 'Based on the provided text, Srinivas has extensive experience as a Technical Lead in software development, including microservices architecture, web and mobile technologies, and AI-driven solutions. His experience spans many years and various companies. This strongly suggests he would be suitable for a software architecture role.\n'}

Notebook Response (Chat)

{'query': 'Is he suitable for the finance manager role?', 'result': 'Based solely on the provided text, it is not possible to determine if Srinivas is suitable for a finance manager role. The resume details his extensive experience in software development and related technologies, but provides no information about his financial expertise or management experience in finance.\n'}

{'query': 'Does he have any specific client-side scripting skills?', 'result': 'Based on the provided text, Srinivas Vasan has experience with JavaScript, React, Angular, and Flutter. These are all client-side scripting technologies.\n'}

{'query': 'Which country owns the football World Cup now?', 'result': "I don't know.\n"}

Notebook functionalities explanation

1. Document Loading:

- **Purpose:** Load the PDF resume file for subsequent processing.
- **Method:**
  - Utilizes PyPDFLoader from the langchain.document_loaders library to efficiently load the PDF document.
  - Stores the loaded document content in the documents variable.

Code: Python

```
loader = PyPDFLoader("/content/sample_data/Srini_Resume.pdf")
documents = loader.load()
```

Notebook functionalities explanation

2. Embedding Creation and Vector Store:

- **Purpose:** Create vector representations of the resume text for efficient similarity searches.
- **Method:**
  - Uses HuggingFaceEmbeddings to generate embeddings for each piece of text in the documents list.
    - Employs the thenlper/gte-large model for efficient and effective text embedding.
  - Creates a FAISS vector store to store and index the document embeddings. FAISS is a fast library for efficient similarity search in high-dimensional spaces.

Code: Python

```
embeddings = HuggingFaceEmbeddings(
    model_name="thenlper/gte-large"
)
vectorstore = FAISS.from_documents(documents, embeddings)
```

Notebook functionalities explanation

3. LLM Configuration:

- **Purpose:** Initialize the Large Language Model (LLM) for generating answers.
- **Method:**
    - Uses ChatGoogleGenerativeAI from the langchain_google_genai library to access the Gemini-1.5-flash model from Google GenerativeAI.
    - Sets safety settings to control the type of responses generated by the LLM.

Code: Python

```
genai.configure(api_key=g_api_key)
 llm = ChatGoogleGenerativeAI(
    model="gemini-1.5-flash",
    safety_settings={
                                HarmCategory.HARM_CATEGORY_DANGEROUS_CONTENT:
HarmBlockThreshold.BLOCK_NONE,
    },
)
```

Notebook functionalities explanation

4. Retrieval and QA Chain Definition:

- **Purpose:** Combine the retriever (for finding relevant document sections) and the LLM (for generating answers) into a single chain.
- **Method:**
  - Creates a `RetrievalQA` chain using the `from_llm()` method.
    - Passes the initialized LLM and the retriever (obtained from the vector store) as arguments.

Code: Python

```
retriever = vectorstore.as_retriever()
qa_chain = RetrievalQA.from_llm(llm, retriever=retriever)
```

Notebook functionalities explanation

5. Question Answering:

- **Purpose:** Ask various questions about the resume and obtain answers using the created RAG chain.
- **Method:**
  - Defines a series of questions related to the resume content (e.g., applicant name, professional experience, skills).
  - Calls the qa_chain.invoke() method for each question, passing the query as input.
  - Prints the generated answers to the console.

Code: Python

```
query = "What is the name of the applicant?"
answer = qa chain.invoke(query)
print(answer)

# ... other questions and their corresponding answers ...
```

Conclusion

- This Colab notebook provides a practical example of how to leverage the power of RAG for resume analysis (any PDF document).
- By combining a robust document embedding model (thenlper/gte-large) with a fast and efficient vector store (FAISS), the system effectively retrieves relevant information from the resume based on user queries.
- The integration with the advanced Gemini-1.5-flash LLM allows for sophisticated and informative answers to a wide range of questions, from simple factual inquiries to more complex queries that require deeper understanding of the resume content.
- This approach demonstrates the potential of RAG for various applications, including resume screening, talent acquisition, and personalized recruitment.
- This Colan notebook URL is provided in the comments.
- Let’s see in the next document for RAG with tools and multi-agents.

Happy Coding!

Thank you