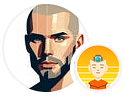
Multiple-PDF Chatbot using Langchain



Carlo C. · Follow

Published in AI monks.io · 6 min read · Oct 22, 2023

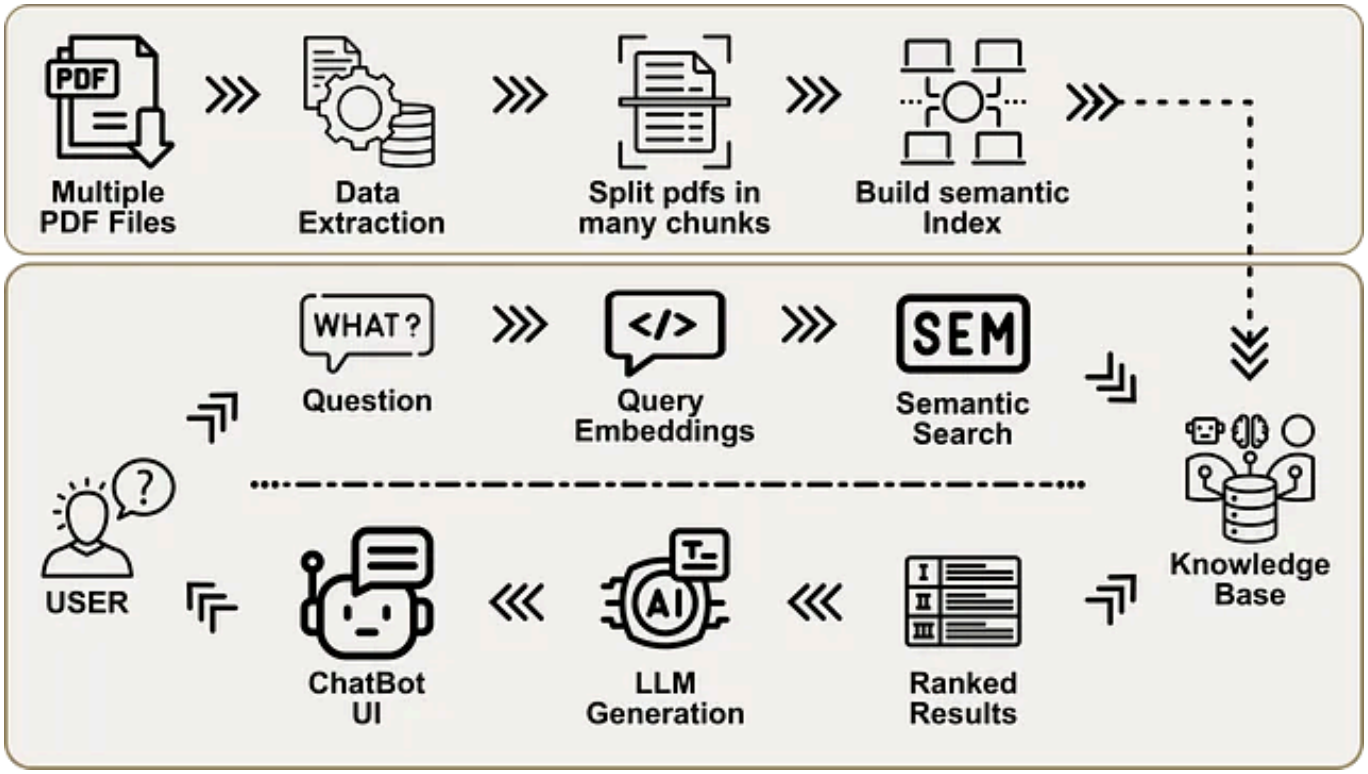
Pdf Chat by Author with ideogram.ai

The goal of the project is to create a question answering system based on information retrieval, which is able to answer questions posed by the user

using PDF documents uploaded from a directory as a source. Deployment can occur in several ways:

- Via Gradio UI
- Via Streamlit UI
- Through FastAPI
- Through Azure Endpoint
- etc....

The scheme used will be the following:



Problem Schema by Author with ideogram.ai

The steps are as follows:

- The first step is to install the necessary libraries for the project, such as langchain, torch, sentence_transformers, faiss-cpu, huggingface-hub, pypdf, accelerate, llama-cpp-python and transformers. These libraries provide the functionality to create the question answering system, such as text generation, embedding creation, approximate nearest neighbor search, PDF document loading, use of neural language model and other auxiliary operations .

```
# install the libraries needed for the project
!pip install langchain
!pip install torch
!pip install sentence_transformers
!pip install faiss-cpu
!pip install huggingface-hub
!pip install pypdf
!pip -q install accelerate
!pip install llama-cpp-python
!pip -q install git+https://github.com/huggingface/transformers
```

- The second step is to import the classes and modules from langchain, a library for text generation based on Markov chains and neural language models. These classes and modules allow you to create the main components of the question answering system, such as the retriever, the text_splitter, the embeddings, the vector_store and the llm.

```
# a class to create a question answering system based on information retrieval
from langchain.chains import RetrievalQA
# a class to create text embeddings using HuggingFace templates
from langchain.embeddings import HuggingFaceEmbeddings
# a class to create a neural language model using LlamaCpp, a C++ implementation
from langchain.llms import LlamaCpp
# a class for splitting text into fixed-sized chunks with an optional overlay
from langchain.text_splitter import RecursiveCharacterTextSplitter
```

```
# a class to create a vector index using FAISS, a library for approximate neares
from langchain.vectorstores import FAISS
# a class for loading PDF documents from a directory
from langchain.document_loaders import PyPDFDirectoryLoader
```

- The third step is to load PDF files from a directory using the PyPDFDirectoryLoader class, which extracts text from PDF documents and returns it in a list of tuples (file name, text extracted from PDF). T' list is then printed to verify the loaded data.

```
# load PDF files from a directory
loader = PyPDFDirectoryLoader("/PDF_Documents/")
data = loader.load()
# print the loaded data, which is a list of tuples (file name, text extracted fr
print(data)
```

- The fourth step is to split the extracted data into text chunks using the text_splitter, which splits the text based on the specified number of characters and overlap. This allows you to create more manageable and homogeneous text units. The number of chunks obtained is then printed to verify the division of the text. You can also get a single chunk by specifying the index in the chunk list.

```
# split the extracted data into text chunks using the text_splitter, which split
text_splitter = RecursiveCharacterTextSplitter(chunk_size=10000, chunk_overlap=2
text_chunks = text_splitter.split_documents(data)
# print the number of chunks obtained
len(text_chunks)
```

- The fifth step is to download the embeddings to be used to represent the text chunks in a vector space, using the pre-trained model “sentence-transformers/all-MiniLM-L6-v2”. This model is able to create semantic embeddings for sentences or paragraphs, i.e. vectors that capture the meaning of the text.

```
# download the embeddings to use to represent text chunks in a vector space,
embeddings = HuggingFaceEmbeddings(model_name="sentence-transformers/all-Mi
```

- The sixth step is to create embeddings for each text chunk using the FAISS class, which creates a vector index using FAISS and allows efficient searches between vectors. This vector index will be used as a retriever, i.e. as a component that retrieves the most relevant text chunks for a given query.

```
# create embeddings for each text chunk using the FAISS class, which creates a v
vector_store = FAISS.from_documents(text_chunks, embedding=embeddings)
```

- The seventh step is to load the mistral-7b-instruct-v0.1.Q1_K_M model, which is a neural language model trained to generate text based on user-provided instructions. The model is imported using the LlamaCpp class, which allows you to use a GPT-3 model in C++ with various parameters such as temperature, top_p, verbose and n_ctx (maximum number of tokens that can be generated).

```
# Import the neural language model using the LlamaCpp class, which allows you to
llm = LlamaCpp(
    streaming = True,
    model_path="/content/drive/MyDrive/Model/mistral-7b-instruct-v0.1.Q1_K_M.ggu
    temperature=0.75,
    top_p=1,
    verbose=True,
```


- The eighth step is to create a question answering system based on information retrieval using the RetrievalQA class, which takes as input a neural language model, a type of chain and a retriever (an object that allows you to retrieve text chunks most relevant to a query). The type of chain used is “stuff”, which indicates that the system must generate a response based on the information retrieved from the retriever.

```
# Create a question answering system based on information retrieval using the Re
qa = RetrievalQA.from_chain_type(llm=llm, chain_type="stuff", retriever=vector_s
```

- The ninth step is to define a query to ask the system and get an answer. The query used as an example is “What is linear regression model”, which asks what a linear regression model is. The response obtained from the system is an explanation of the concept with some examples.

```
# define a query to ask the system
query = "What is linear regression model"
```

```
# run the system and get a response
qa.run(query)
```

- The tenth step is to create an infinite loop to interact with the system, asking the user to enter a query or exit the program. The system takes the user's query and returns a response generated by the neural language model, based on the information retrieved by the retriever.

```
# Create an infinite loop to interact with the system, asking the user to enter
import sys

while True:
  user_input = input(f"Input Prompt: ")
  if user_input == 'exit':
    print('Exiting')
    sys.exit()
  if user_input == '':
    continue
  # pass the query to the system and print the response
  result = qa({'query': user_input})
  print(f"Answer: {result['result']}")
```

This work is commented on and supplemented by this work. This tutorial article showed how to create an information retrieval-based question answering system, using different libraries such as langchain, torch, sentence_transformers, llama-cpp-python and transformers. The system is able to answer questions posed by the user using PDF documents loaded from a directory as a source. i is based on three main components: a retriever, which retrieves the most relevant text chunks for a query; a neural language model, which generates a response based on instructions provided by the user; and a RetrievalQA class, which coordinates the two components

and returns the final response. The system is interactive and allows the user to ask various queries or exit the program. The system is also flexible and can be adapted to different data sources, embedding models, language models and chain types. We hope this tutorial article has been useful and interesting for readers who want to create their own information retrieval-based question answering system.

Machine Learning | Data Science | Data Scientist | NLP | Langchain

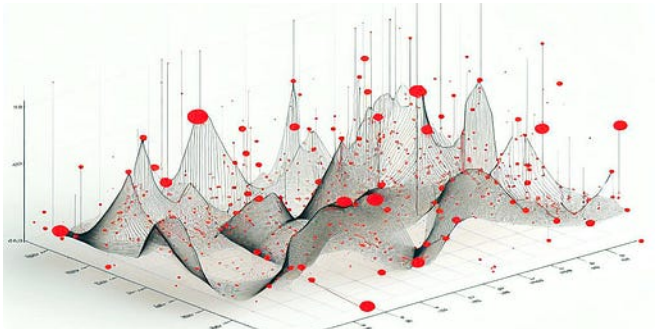
Written by Carlo C.

363 Followers · Writer for AI monks.io

Follow

Data scientist, avidly exploring ancient philosophy as a hobby to enhance my understanding of the world and human knowledge.

More from Carlo C. and AI monks.io



Carlo C. in AI monks.io

Anomaly detection for Time Series Analysis

Time series analysis is a very useful and powerful technique for studying data that...

12 min read · Nov 2, 2023

531 3

Webjinneeofficial in AI monks.io

50 Tasks that ChatGPT Can Do ir Seconds with These Simple...

You've probably heard of ChatGPT by now— that incredibly smart AI can understand and...

4 min read · Apr 9, 2024

321 5

Anish Singh Walia in AI monks.io

Top 7 Secret Websites That Pay You $100—$1000 to Work From...

In the digital age, online platforms are the gateway to financial freedom, offering...

8 min read · Mar 24, 2024

628 9

Carlo C. in AI monks.io

Physics-Informed Neural Networks for Inverse Design

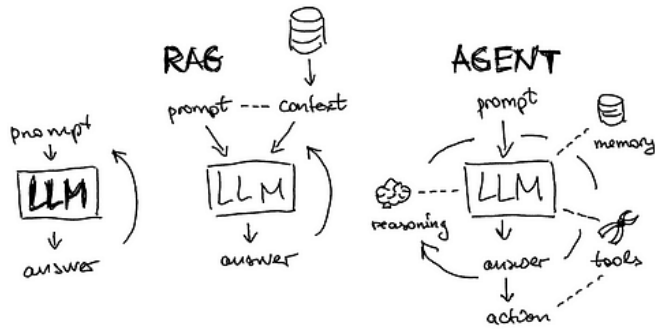
Physics-Informed Neural Networks (PINNs) represent a groundbreaking integration of...

4 min read · Mar 3, 2024

27 1

See all from Carlo C. See all from AI monks.io

Recommended from Medium



Alex Honchar in Towards Data Science

Intro to LLM Agents with Langchain: When RAG is Not...

First-order principles of brain structure for AI assistants

7 min read · Mar 15, 2024

2.1K 10

Madhav Thaker

(Part 1) Build your own RAG with Mistral-7B and LangChain

LLMs have taken the world by storm and rightfully so. They help you with every day...

14 min read · Nov 14, 2023

2.4K 15

Lists

Predictive Modeling w/ Python

20 stories · 1132 saves

Practical Guides to Machine Learning

10 stories · 1356 saves

Natural Language Processing

1407 stories · 905 saves

The New Chatbots: ChatGPT, Bard, and Beyond

12 stories · 362 saves

Ala Eddine GRINE in The Deep Hub

RAG chatbot powered by Langchain, OpenAI, Google...

Introduction

15 min read · Feb 14, 2024

155 2

Gavin Li in AI Advances

Run the strongest open-source LLM model: Llama3 70B with just ...

The strongest open source LLM model Llama3 has been released, Here is how you...

4 min read · 6 days ago

1K 2

Pieces in Pieces for Developers

How to Build a Langchain PDF Chatbot

7 min read · Oct 31, 2023

Fabio Matricardi in Generative AI

Llama3 is out and you can run it on your Computer!

After only 1 day from the release, here is how you can run even on your Laptop with CPU...

· 8 min read · Apr 20, 2024