Domain-Specific Chatbot for PDF-Based Learning

A Dissertation

Submitted by

Ramya Mercy Rajan

(AA.SC.P2MCA2107434)

in partial fulfilment of the requirements for the award of

the degree of

MASTER OF COMPUTER APPLICATIONS

November 2023

BONAFIDE CERTIFICATE

This is to certify that this dissertation titled "Domain-Specific Chatbot for PDF-Based Learning," submitted in partial fulfilment of the requirements for the award of the Degree of **Master of Computer Applications**, by Ramya Mercy Rajan AA.SC.P2MCA2107434, is a bona fide record of the work carried out by him/her under my supervision during the academic year 2022-2023 and that it has not been submitted, to the best of my knowledge, in part or in full, for the award of any other degree or diploma.

Project Guide’s name

Coordinator's name

Reviewer

Date:

DECLARATION

I do hereby declare that this dissertation titled "Domain-Specific Chatbot for PDF-Based Learning", submitted in partial fulfilment of the requirements for the award of the degree of **Master of Computer Applications,** is a true record of work carried out by me and that all information contained herein, which do not arise directly from my work, have been properly acknowledged and cited, using acceptable international standards. Further, I declare that the contents of this thesis have not been submitted, in part or in full, for the award of any other degree or diploma.

Signature of the student

Date: 07/11/2023 **Ramya Mercy Rajan**

Acknowledgements

I would like to express my deep gratitude and appreciation to all those who have supported me throughout the journey of completing my major project for the Master of Computer Application with a specialization in Artificial Intelligence.

I extend my heartfelt thanks to my academic guide, Ms. Deepa Sreedhar, whose guidance, mentorship, and expertise were instrumental in shaping this project. Your unwavering support, insightful feedback, and patience were the driving force behind the successful completion of this endeavor.

I would also like to acknowledge the contributions of MCA Program Coordinator, Dr. Manjusha Nair, Academic Head MCA, Ms. Jayashree Narayanan and Project Coordinator, Ms. Vidyalekshmi Vinod. Your support and guidance have played a crucial role in my academic journey, and I am truly grateful for the assistance and encouragement provided by all of you. Thank you for making this achievement possible.

I would like to extend my profound gratitude to my better half, Mr. Abin Alex, who offered encouragement, patience, and unwavering support throughout this demanding journey.

This project has been a significant learning experience, and I am thankful for the opportunity to work on a topic that holds great promise for the field of Artificial Intelligence and education. The knowledge and skills acquired during the course of this project will undoubtedly be invaluable in my future endeavors.

Abstract

This final project report summarizes the successful completion and outcomes of the project, which focused on the development of a domain-specific chatbot empowered with the capability to extract knowledge from PDF documents and deliver contextually pertinent responses. This comprehensive report provides an overview of the project's objectives, scope, introduction, historical context, related works, detailed problem statement, implemented methods and algorithms, and the achieved results.

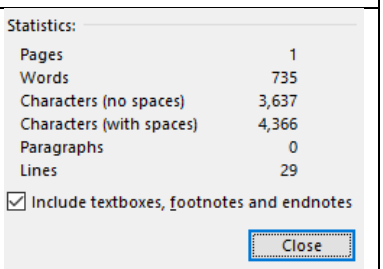
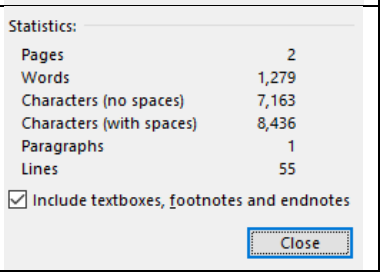
The primary objective of this project was to create an intelligent chatbot capable of autonomously acquiring knowledge from PDF documents in diverse domains. This chatbot harnesses the power of natural language processing (NLP) techniques to offer precise and context-aware answers to user inquiries, all driven by the content extracted from the supplied PDFs. This innovative project endeavors to produce an adaptable and information-rich chatbot with the potential to cater to a wide range of domains, including but not limited to fields like Artificial Intelligence learning platform. By doing so, it seeks to elevate the accessibility and efficacy of educational materials, thus benefiting learners and educators alike. The successful realization of this project underscores the potential of AI-driven technologies to enhance learning and information retrieval in various specialized domains.

Contents

List of Figures

List of Tables

I have collected of 2 educational PDF documents covering topics NLP and Supervised Learning with original number of characters as shown in table. Text extraction was performed using the PyPDF2 library, resulting in a total of 2 documents with 4400 characters from these documents.

S.No.	PDF Document	
1	NLP	
2	Supervised Learning	

Chapter 1

Introduction

1.1 Section A: Project Introduction

A.1 Project Overview

In an era driven by information and automation, our project represents a pioneering effort in the development of a domain-specific chatbot empowered with the remarkable capability to extract knowledge from PDF documents. This comprehensive report encapsulates the successful culmination of our endeavors, highlighting the fusion of cutting-edge technologies and innovative solutions.

A.2 Key Objectives

The project's central objectives were meticulously crafted:

Develop an autonomous and adaptive chatbot capable of comprehending and extracting knowledge from PDF documents across diverse domains.

Implement advanced NLP techniques to empower the chatbot to provide context-aware and accurate responses to user inquiries.

Elevate the accessibility and effectiveness of educational materials by focusing on specialized domains, such as the Artificial Intelligence learning platform.

Showcase the potential of AI-driven technologies to revolutionize learning and information retrieval, benefitting learners and educators alike.

BUSINESS USE CASE: ENHANCING MEDICAL EDUCATION WITH A DOMAIN-SPECIFIC CHATBOT

Medical education involves extensive reading of complex research papers, textbooks, and clinical guidelines to keep up with the ever-evolving field. The Domain-Specific Chatbot for PDF-Based Learning project seeks to streamline

this process by creating an intelligent chatbot that can extract crucial information from medical PDFs, providing medical students with quick and accurate answers to their queries.

BUSINESS USE CASE: MEDICAL LEARNING PLATFORM

Scenario: A medical education platform aims to provide medical students with comprehensive resources to supplement their coursework. This platform hosts a vast library of medical research papers, clinical studies, and textbooks in PDF format. However, the sheer volume of information can be overwhelming, making it challenging for students to quickly find and comprehend relevant insights.

Solution: The platform integrates the Domain-Specific Chatbot for PDF-Based Learning, which becomes an indispensable learning companion for medical students.

Benefits:

Instant Answers to Medical Queries: Medical students often encounter complex medical concepts while studying. Instead of spending valuable time searching through PDFs, they can ask the chatbot questions. The chatbot leverages its ability to extract knowledge from PDFs and provides concise, accurate answers, helping students grasp concepts efficiently.

Efficient Research: When medical students need information for assignments, research projects, or presentations, they can use the chatbot to quickly gather pertinent data from a multitude of PDF resources. This reduces the time spent on data collection, allowing more time for analysis and synthesis.

Real-time Updates: The medical field evolves rapidly, with new research shaping clinical practices. The chatbot can be designed to track and analyze the latest research papers and clinical guidelines, ensuring that students receive up-to-date and evidence-based information.

Personalized Learning: By understanding a student's specific area of interest or specialization, the chatbot can recommend relevant articles, studies, and resources, tailoring the learning experience to individual needs.

Accessible Learning: The chatbot provides a user-friendly interface that accommodates students with varying learning styles and abilities. It allows students to access and understand complex medical content without being overwhelmed.

Improved Learning Outcomes: With quick access to accurate information and simplified explanations, students can enhance their understanding of medical concepts, leading to improved academic performance and clinical competence.

In the realm of medical education, the Domain-Specific Chatbot for PDF-Based Learning has the potential to revolutionize how medical students interact with and learn from extensive PDF resources. By offering quick, accurate, and personalized insights, the chatbot can significantly enhance the efficiency and effectiveness of medical education, ultimately producing more knowledgeable and skilled medical professionals.

Background

The project aims to address the challenge of making information from PDF-based learning materials more accessible. By developing a chatbot that learns from PDFs, it seeks to provide instant and accurate responses to user queries, enhancing the learning experience.

1.2 Section B: Project Significance

B.1 Empowering Knowledge Extraction

In a world inundated with PDF documents and digital resources, the ability to swiftly and accurately extract knowledge is paramount. Our project represents a groundbreaking approach to this challenge by creating a chatbot that not only understands the intricacies of these documents but also facilitates the dissemination of this knowledge in a user-friendly manner.

B.2 Catalyzing Learning and Education

Beyond its technical prowess, our project holds the promise of transforming education and learning. By focusing on domains like the Artificial Intelligence learning platform, we aim to make specialized knowledge more accessible, thereby benefiting students, educators, and enthusiasts seeking to expand their horizons.

B.3 The Promise of AI-Driven Innovation

The successful realization of our project underscores the transformative potential of AI-driven technologies. It showcases how intelligent automation can enhance learning and information retrieval across specialized domains, opening new frontiers for knowledge dissemination.

Chapter 2

Problem Definition

The project addresses the challenge of efficiently extracting relevant information from PDF documents and using that information to create a knowledgeable chatbot. The objective is to bridge the gap between static PDF learning materials and interactive learning experiences.

The exponential growth of digital content, particularly in the form of PDF documents, has presented a significant challenge: the efficient extraction and utilization of knowledge contained within these documents. Our project tackles this challenge head-on by introducing an intelligent chatbot, supported by state-of-the-art natural language processing (NLP) techniques. This chatbot not only extracts pertinent information from PDFs but also provides contextually precise responses to user queries.

Problem

In today's rapidly evolving digital landscape, there's a compelling need for a chatbot capable of efficiently extracting and delivering domain-specific knowledge from PDF documents. The challenge lies in empowering users, including professionals and enthusiasts in specialized fields, to access and utilize information stored in PDFs with ease and accuracy.

Example of the Problem

Consider a legal consultancy firm that manages an extensive collection of legal documents and case studies in PDF format. Legal professionals frequently require swift access to specific legal precedents and insights. However, they often encounter the challenge of spending substantial time manually searching through PDFs. This can result in delays in client services and the risk of overlooking critical legal information. The solution lies in developing a domain-specific chatbot capable of rapidly extracting relevant legal knowledge from PDFs. Such a chatbot can provide context-aware responses to user inquiries, ultimately enhancing the efficiency of legal research and services.

Chapter 3

Related Work

Related Works/Existing System

Reviews of Earlier Research

Some educational platforms incorporate search functionality to retrieve information from their content repositories, which may include PDFs. However, these systems primarily emphasize keyword-based search and retrieval, lacking the ability to engage in contextually rich conversations or comprehend complex queries beyond simple keyword matching.

Unique Features of the Project:

- Contextually Relevant Conversations: Unlike existing document summarizers or search-driven platforms, our project aims to create a chatbot that engages users in contextually relevant conversations. It will not only extract information from PDFs but also comprehend user queries in the context of the document's subject matter, providing in-depth, conversational responses.

- Domain-Specific Language Understanding: This project involves training a domain-specific language model that understands and generates text specific to the chosen subject, such as medical topics. This ensures that the chatbot's responses are accurate, relevant, and aligned with the nuances of the domain.

- Seamless Integration with Learning: Our project targets the educational context directly by seamlessly integrating with learning materials. Unlike general-purpose chatbots, our system will focus on educational content extraction and engagement, catering to the needs of students and learner

Chapter4

Requirements

The design of this project contains both hardware and software. The specifications are listed below.

4.1 Hardware

The hardware components of this project play a crucial role in ensuring its functionality and effectiveness. The following are the key hardware specifications:

- Central Processing Unit (CPU): The project utilizes a high-performance CPU to handle complex data processing tasks. The CPU's processing power is optimized to efficiently execute the software algorithms responsible for extracting and understanding information from PDF documents.

- Memory (RAM): To facilitate rapid data retrieval and processing, the system is equipped with ample RAM. This ensures that the chatbot can quickly access and manipulate information stored in memory, providing users with prompt responses to their queries.

- Storage: A substantial storage capacity is provided to accommodate the vast collection of PDF documents and associated data. This storage space allows for the archiving of educational materials and the efficient retrieval of relevant information.

- Graphics Processing Unit (GPU): While primarily a software-focused project, certain machine learning tasks benefit from GPU acceleration. Therefore, a dedicated GPU is integrated into the hardware configuration to enhance the performance of deep learning algorithms used in natural language processing.

- Network Connectivity: The project relies on network connectivity to access external resources, such as online research papers and updates. High-speed internet connectivity is essential to ensure seamless communication between the chatbot and external databases.

4.2 Software

The software component of this project is the heart of its functionality, enabling the chatbot to extract knowledge from PDF documents and engage in contextually rich conversations. Here are the key software specifications:

- Operating System: The project runs on a robust operating system (e.g., Linux or Windows) chosen for its stability and compatibility with the required software libraries and frameworks.

- Programming Languages: The primary programming languages employed include Python for its versatility and a wide range of libraries relevant to natural language processing and machine learning.

- Natural Language Processing and PDF Libraries: To enable the chatbot to understand and generate human-like text, the project leverages state-of-the-art natural language processing libraries such as OpenAI Embedding, Open AI LLM ChromaDB, Pinecone, Conversational Retrieval Chain Gradio ,GPT 3.5, PDFminer,pdf2image, GooglePalm LLM, HuggingFace Instruct Embeddings and vector store FAISS (Facebook AI Similarity Search (Faiss) and RetrievalQA were used.

- Multiple models, each with different capabilities and price points. Prices are per 1,000 tokens. It is considered as pieces of words, where 1,000 tokens is about 750 words. This paragraph is 35 tokens.
- **GPT 3.5** With broad general knowledge and domain expertise, GPT 3.5 can follow complex instructions in natural language and solve difficult problems with accuracy.
- User Interface (UI): The software includes a user-friendly interface through which users can interact with the chatbot. The UI is designed to provide a seamless and intuitive experience for users seeking information. Gradio and from IPython.display imported display

  imported ipywidgets as widgets were used.
- Integration with PDF Processing Tools: Specialized PDF processing libraries and tools are integrated into the software to parse, extract, and analyze information from PDF documents. These tools ensure the chatbot's ability to comprehend and respond to user inquiries.

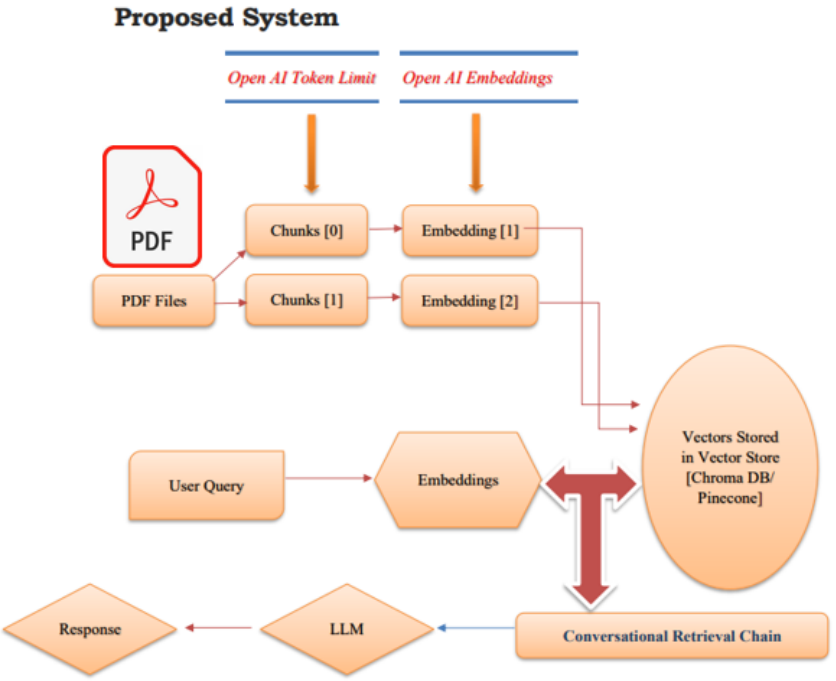
These hardware and software specifications collectively form the foundation of the project, enabling it to deliver on its goal of efficiently extracting knowledge from PDF documents and engaging users in contextually relevant conversations.

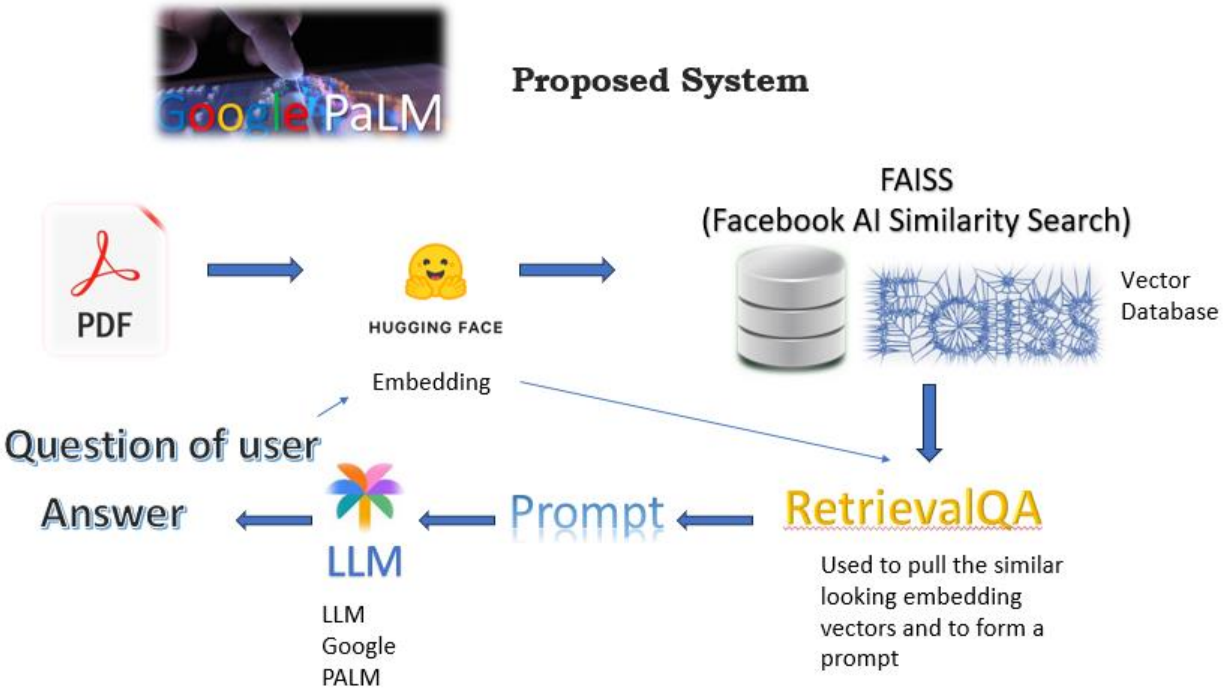
Chapter 5

Proposed System

OpenAI

Behind ChatGPT there is a model called GPT4. GPT4 is the LLM behind chat GPT. The company who build this is OpenAI. Similar to GPT4 there are 2 other LLM's in the market Lama by META and Palm by Google. Lama and Palm are free but Open AI charges.




Google PaLM
Proposed System
FAISS
(Facebook AI Similarity Search)
PDF
HUGGING FACE
Vector
Database
Embedding
Question of user
Answer
LLM
Prompt
RetrievalQA
Used to pull the similar looking embedding vectors and to form a prompt
LLM
Google
PALM

Chapter 6

Results and Analysis

1. Introduction

- The aim of this project was to create a domain-specific chatbot tailored for PDF-based learning. This chatbot was designed to assist users in extracting information and answering questions from educational PDF documents. The technologies and frameworks used for this project include as OpenAI Embedding, Open AI LLM ChromaDB, Pinecone, Conversational Retrieval Chain Gradio ,GPT 3.5, PDFminer,pdf2image, GooglePalm LLM, HuggingFace Instruct Embeddings and vector store FAISS (Facebook AI Similarity Search (Faiss) and RetrievalQA.

2. Data Preparation

I have collected 2 educational PDF documents covering topics **NLP and Supervised Learning.**
Text extraction was performed using the PyPDF2 library, resulting in a total of 2 documents with 4400 characters from these documents.

3. Text Splitting and Vectorization

To optimize the processing of PDF content, a recursive character-based text splitter was applied with a chunk size of 500 characters and a chunk overlap of 0.
The extracted text data was converted into vectorized embeddings using OpenAI's embeddings and stored in a Chroma DB vector store / Pinecone vector store.

Also for another trial extracted text data was converted into vectorized embeddings using HuggingFaceInstruct embeddings and stored in a vector store FAISS .

4.Conversational Retrieval Chain - OpenAI/ Retrieval QA

The project utilized a Conversational Retrieval Chain to enable chat history retrieval, allowing the chatbot to maintain context and provide coherent responses.

The retriever component used a similarity search with a k value of 2 to retrieve relevant documents.The question-answering (QA) component employed OpenAI's LLM with a temperature setting of 0 to generate responses.
The Retrieval QA used in the case of Google Palm and Conversational Retrieval chain used in the case of Open AI, looks for similar vectors as mentioned above. Conversational Retrieval chain also saves/remembers the chat history and it shows the previous chats. Then these similar vectors are converted into original sentence/ chunks and then we can form a prompt. These text chunks will be possible answers that we have in our pdf files. And when we give our prompt to our llm, OpenAI / GooglePalm will produce a coherent human readable answer.

5. Chat History Retrieval used for Open AI / Prompt Template for Google Palm in this project.

Chat history retrieval was implemented to maintain context during user interactions.The chat history variable was used to store previous user queries and chatbot responses, enabling the chatbot to provide meaningful and context-aware answers.The chat history was essential for tracking the conversation's progress and ensuring relevant responses.
Remembering chat history | 🦜🔗 Langchain

Prompt template was added while using Google Palm which ask for the exact answer and avoid giving wrong answer.
Using a Retriever | 🦜🔗 Langchain

```
from langchain.prompts import PromptTemplate

prompt_template = """Given the following context and a question, generate an answer based on this context only.
In the answer try to provide as much text as possible from "response" section in the source document context without making much changes.
If the answer is not found in the context, kindly state "I don't know." Don't try to make up an answer.

CONTEXT: {context}

QUESTION: {question}"""“

PROMPT = PromptTemplate(
        template=prompt_template,     input_variables=["context", "question"]
)
chain_type_kwargs = {"prompt": PROMPT}
```

6. Pinecone, ChromaDB and Hugging Face Integration

Pinecone was used as the vectorstore for storing and indexing vectorized document embeddings created from PDF documents.
ChromaDB, in conjunction with OpenAI embeddings, was utilized to efficiently handle the storage and retrieval of vectorized data.
Pinecone and ChromaDB play a crucial role in enabling fast and accurate document retrieval.
HuggingFaceInstruct Embedding was used for embedding trial with Google Palm LLM.
Pinecone | 🦜🔗 Langchain
InstructEmbeddings | 🦜🔗 Langchain

7. Widgets and Gradio Integration

The project integrated widgets and Gradio to create a user-friendly chatbot interface.
Widgets and Gradio were used to provide an interactive chat interface where users could input questions and receive responses from the chatbot.
The Gradio interface allowed users to initiate conversations, and chat history was maintained throughout interactions.
This interactive interface enhanced user experience and made the chatbot more accessible.

8. Chatbot Interaction

A sample interaction with the chatbot was conducted:
User Query: "What is Multi-layer perceptron?"
Chatbot Response: “Multi-layer perceptron is a classifier in which the weights of the network are found by solving a quadratic programming problem with linear constraints, rather than by solving a non-convex, unconstrained minimization problem as in standard neural network training.”

9. Discussion of Results

The chatbot demonstrated effective text extraction and vectorization from PDF documents.
Responses generated by the chatbot were coherent and relevant to user queries.

10. Use Cases and Applications

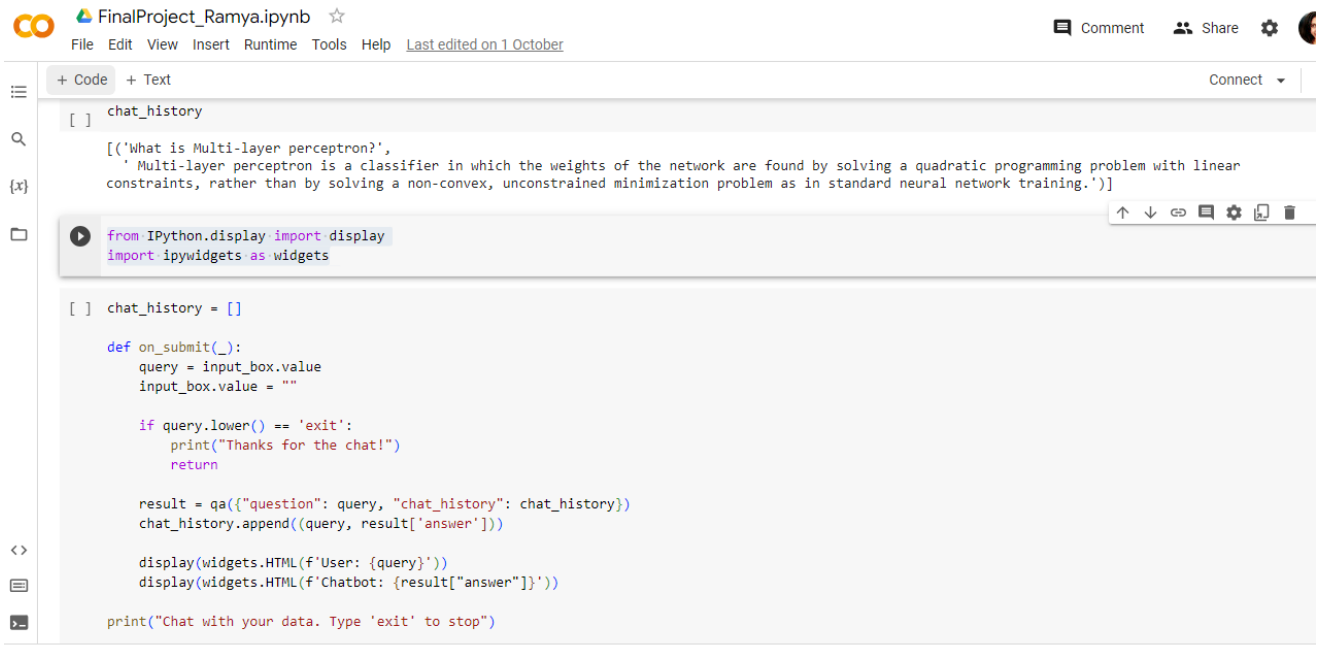
The chatbot has potential applications in educational institutions, online courses, and self-paced learning.
It can assist learners in quickly retrieving information from extensive PDF materials.

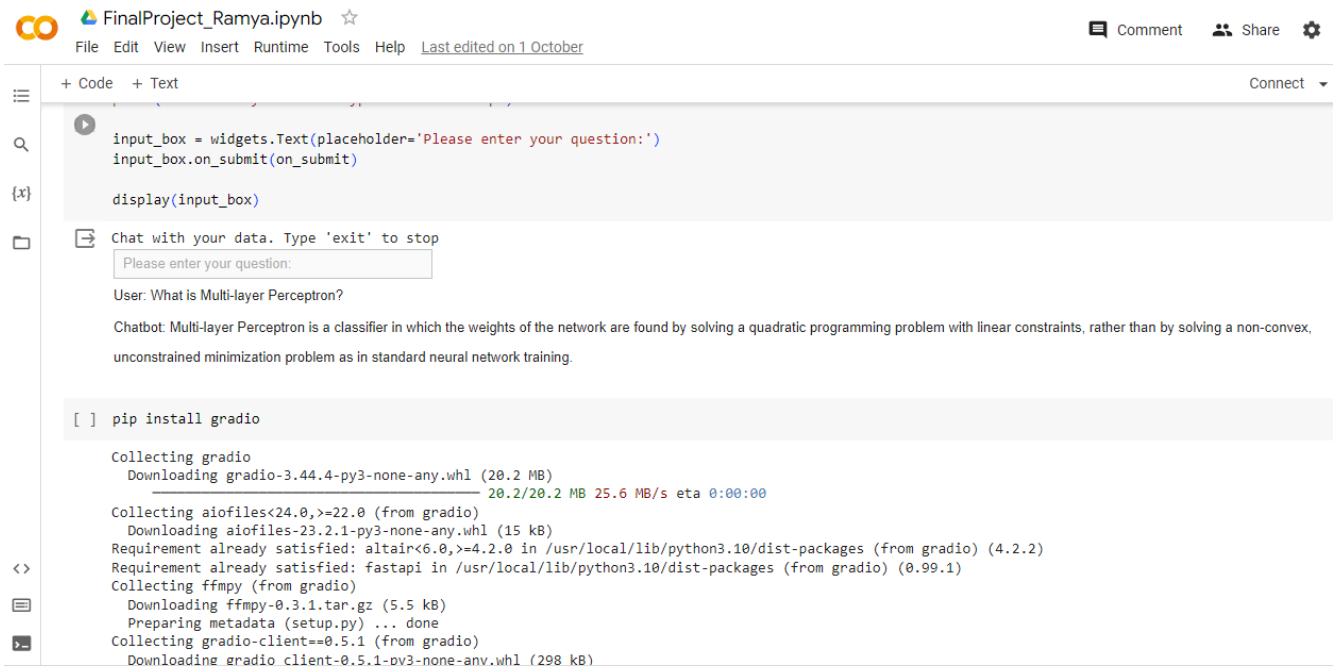
This tool can potentially improve user efficiency and learning outcomes.

In conclusion, the development of a domain-specific chatbot for PDF-based learning has shown promise. The chatbot successfully extracts text from PDFs and provides relevant responses to user queries. However, there is room for improvement in handling complex queries and document formatting. Future work will focus on refining the chatbot's capabilities and expanding its use cases.

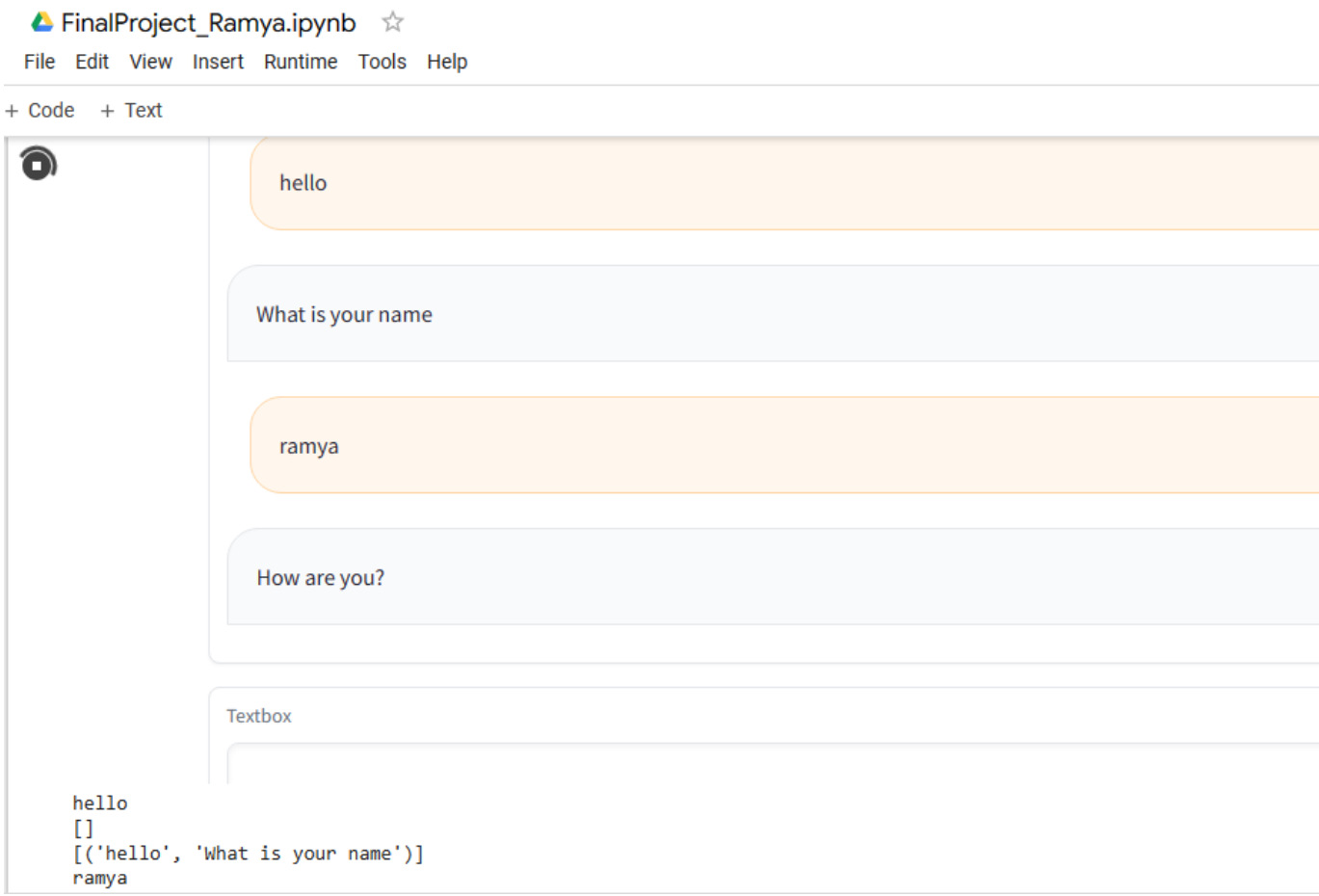
11. Future Work

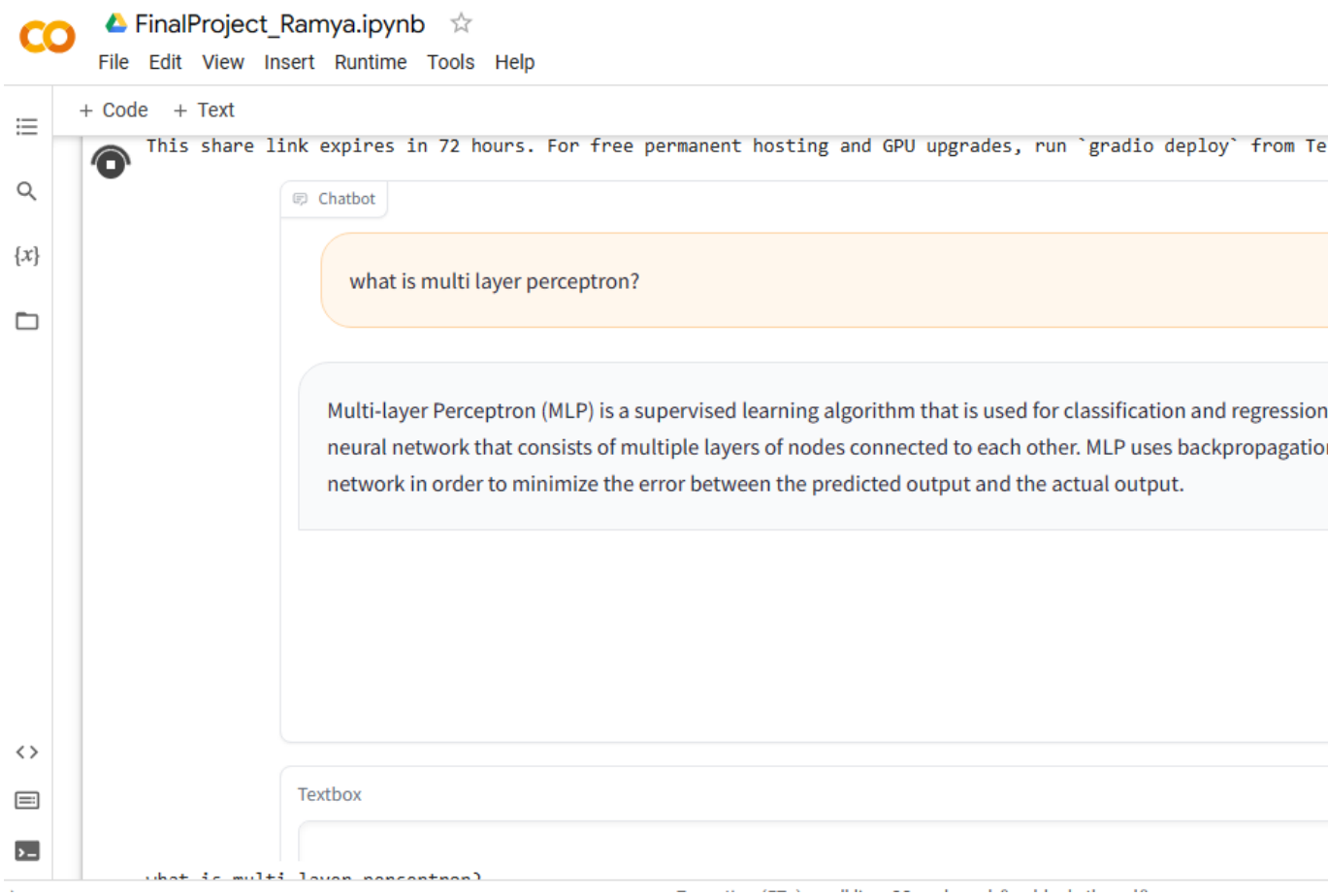
Enhance the chatbot's natural language understanding capabilities.
Improve handling of documents with complex structures and formatting.

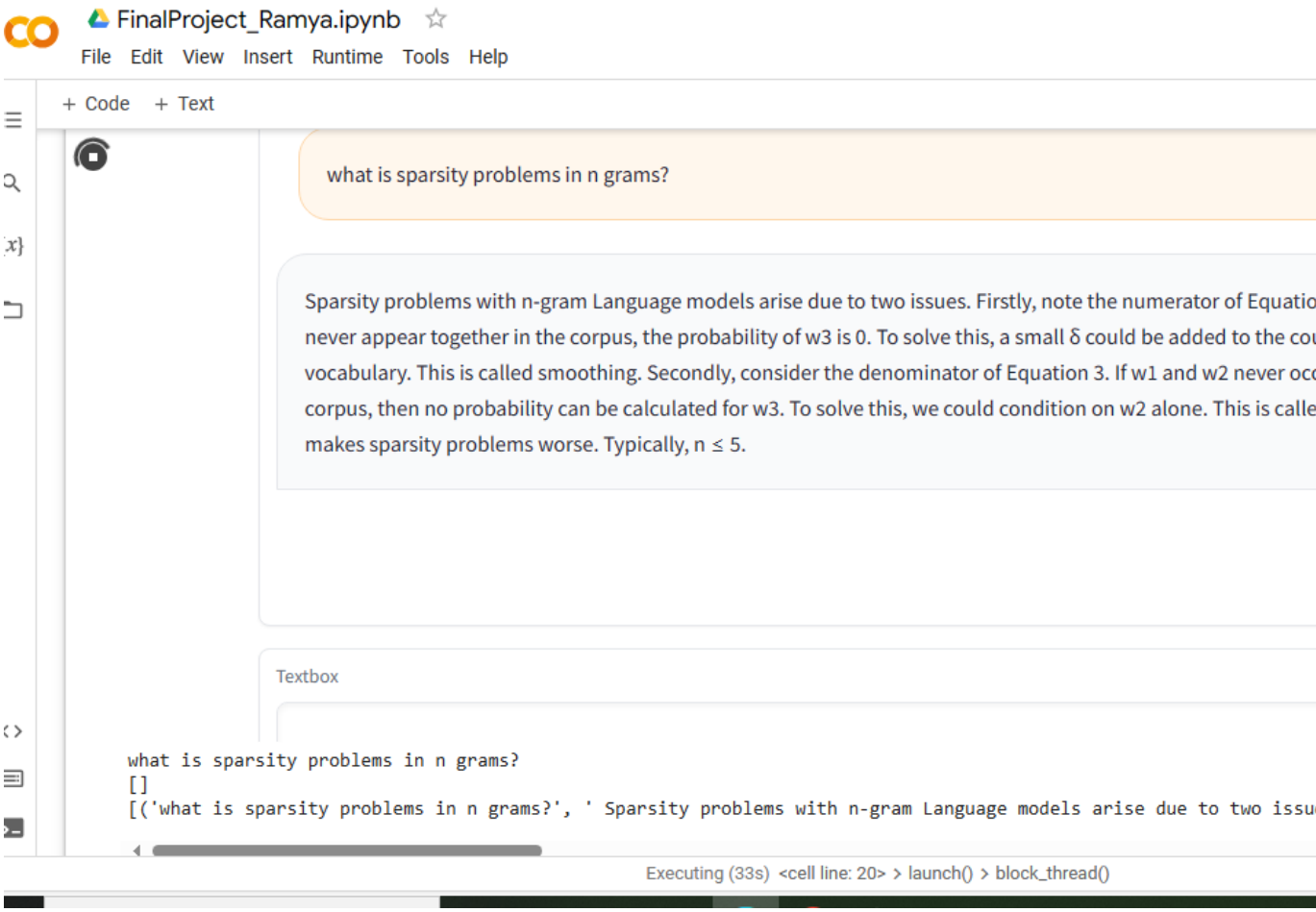

FinalProject_Ramya.ipynb
File Edit View Insert Runtime Tools Help Last edited on 1 October
Comment
Share
Connect
+ Code
+ Text
[ ] chat_history
[('What is Multi-layer perceptron?',
' Multi-layer perceptron is a classifier in which the weights of the network are found by solving a quadratic programming problem with linear constraints, rather than by solving a non-convex, unconstrained minimization problem as in standard neural network training.')]
from IPython.display import display
import ipywidgets as widgets
[ ] chat_history = []
def on_submit(_):
query = input_box.value
input_box.value = ""
if query.lower() == 'exit':
print("Thanks for the chat!")
return
result = qa({"question": query, "chat_history": chat_history})
chat_history.append((query, result['answer']))
display(widgets.HTML(f'User: {query}'))
display(widgets.HTML(f'Chatbot: {result["answer"]}'))
print("Chat with your data. Type 'exit' to stop")

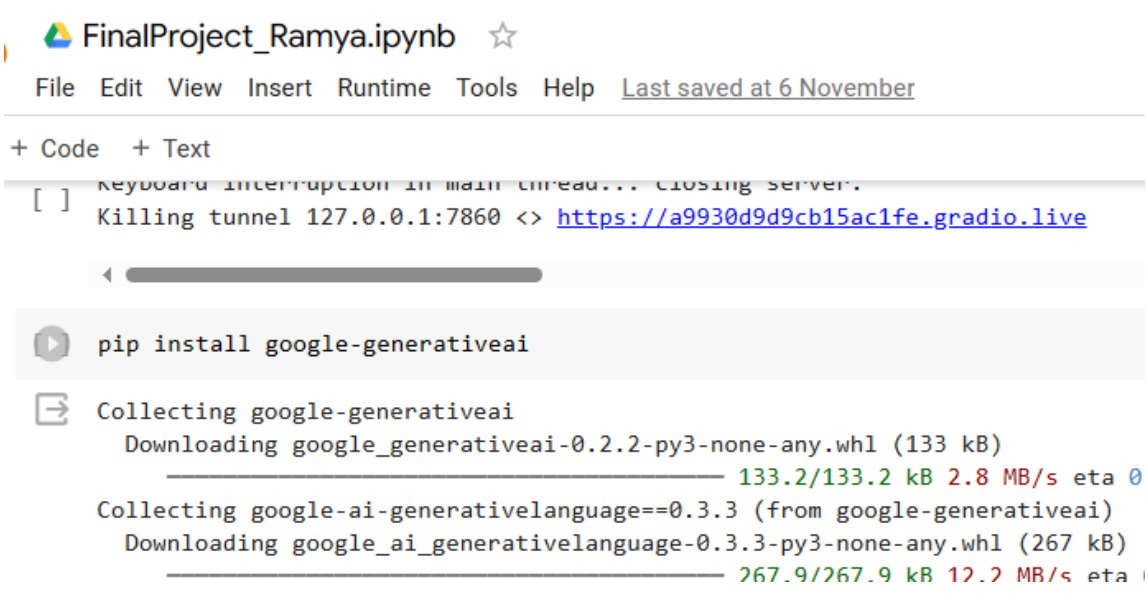

FinalProject_Ramya.ipynb
File Edit View Insert Runtime Tools Help Last edited on 1 October
Comment
Share
Connect
+ Code
+ Text
input_box = widgets.Text(placeholder='Please enter your question:')
input_box.on_submit(on_submit)
display(input_box)
Chat with your data. Type 'exit' to stop
Please enter your question:
User: What is Multi-layer Perceptron?
Chatbot: Multi-layer Perceptron is a classifier in which the weights of the network are found by solving a quadratic programming problem with linear constraints, rather than by solving a non-convex, unconstrained minimization problem as in standard neural network training.
[ ] pip install gradio
Collecting gradio
Downloading gradio-3.44.4-py3-none-any.whl (20.2 MB)
20.2/20.2 MB 25.6 MB/s eta 0:00:00
Collecting aiofiles<24.0,>=22.0 (from gradio)
Downloading aiofiles-23.2.1-py3-none-any.whl (15 kB)
Requirement already satisfied: altair<6.0,>=4.2.0 in /usr/local/lib/python3.10/dist-packages (from gradio) (4.2.2)
Requirement already satisfied: fastapi in /usr/local/lib/python3.10/dist-packages (from gradio) (0.99.1)
Collecting ffmpy (from gradio)
Downloading ffmpy-0.3.1.tar.gz (5.5 kB)
Preparing metadata (setup.py) ... done
Collecting gradio-client==0.5.1 (from gradio)
Downloading gradio_client-0.5.1-py3-none-any.whl (298 kB)

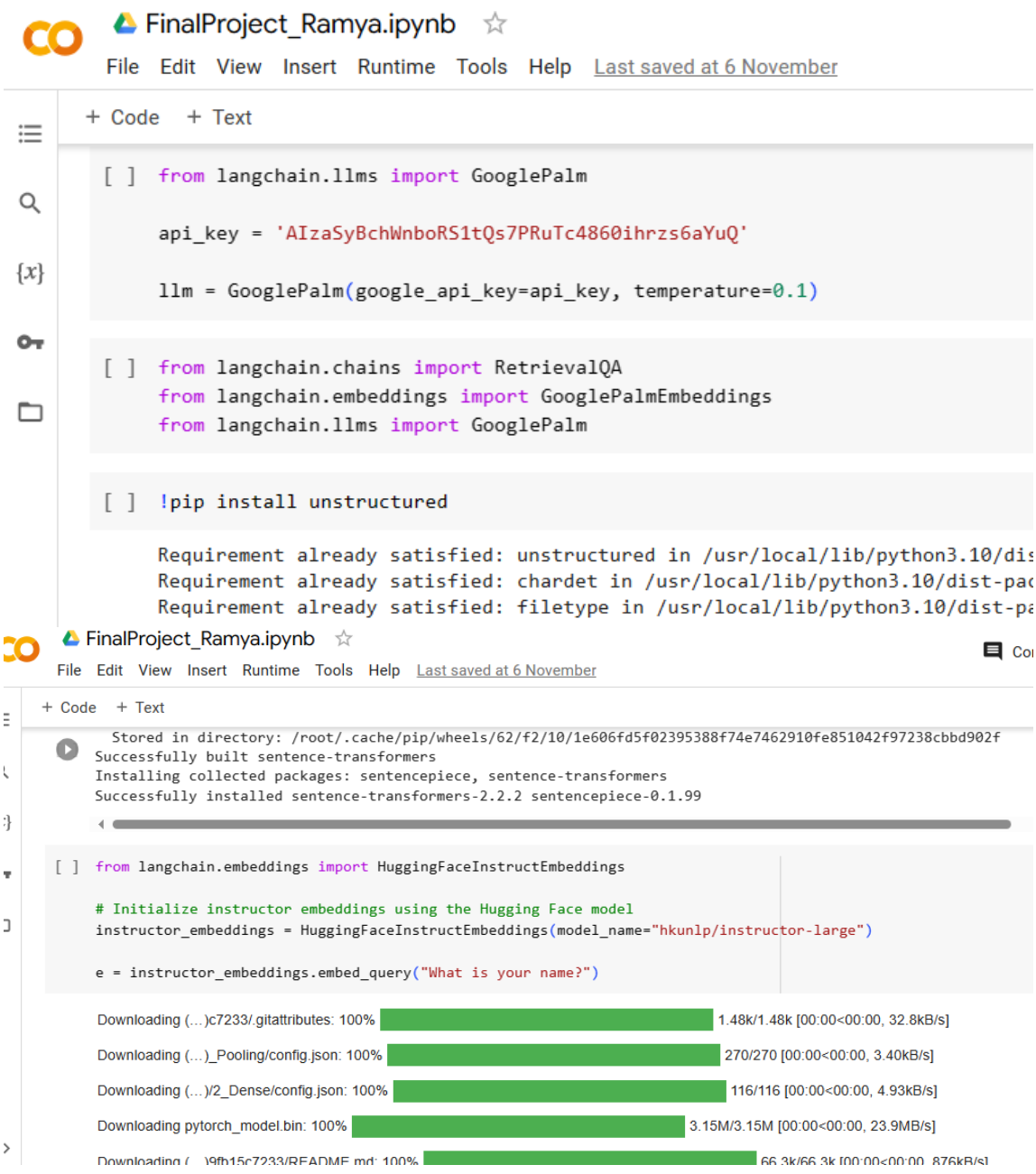

FinalProject_Ramya.ipynb
File Edit View Insert Runtime Tools Help Last edited on 1 October
+ Code + Text
import gradio as gr
with gr.Blocks() as demo:
chatbot = gr.Chatbot()
msg = gr.Textbox()
clear = gr.Button("Clear")
def respond(user_message, chat_history):
print(user_message)
print(chat_history)
# Get response from QA chain
response = qa({"question": user_message, "chat_history": chat_history})
# Append user message and response to chat history
chat_history.append((user_message, response["answer"]))
print(chat_history)
return "", chat_history
msg.submit(respond, [msg, chatbot], [msg, chatbot], queue=False)
clear.click(lambda: None, None, chatbot, queue=False)
demo.launch(debug=True, share=True)
Colab notebook detected. This cell will run indefinitely so that you can see errors and logs. To turn off, set de
Running on public URL: https://1ca08456d41becaa26.gradio.live
This share link expires in 72 hours. For free permanent hosting and GPU upgrades, run `gradio deploy` from Termin


FinalProject_Ramya.ipynb
File Edit View Insert Runtime Tools Help
+ Code + Text
hello
What is your name
ramya
How are you?
Textbox
hello
[]
[('hello', 'What is your name')]
ramya


FinalProject_Ramya.ipynb
File Edit View Insert Runtime Tools Help
+ Code + Text
This share link expires in 72 hours. For free permanent hosting and GPU upgrades, run `gradio deploy` from Te
Chatbot
what is multi layer perceptron?
Multi-layer Perceptron (MLP) is a supervised learning algorithm that is used for classification and regression
neural network that consists of multiple layers of nodes connected to each other. MLP uses backpropagatio
network in order to minimize the error between the predicted output and the actual output.
Textbox


FinalProject_Ramya.ipynb
File Edit View Insert Runtime Tools Help
+ Code + Text
what is sparsity problems in n grams?
Sparsity problems with n-gram Language models arise due to two issues. Firstly, note the numerator of Equatio
never appear together in the corpus, the probability of w3 is 0. To solve this, a small δ could be added to the cou
vocabulary. This is called smoothing. Secondly, consider the denominator of Equation 3. If w1 and w2 never occ
corpus, then no probability can be calculated for w3. To solve this, we could condition on w2 alone. This is calle
makes sparsity problems worse. Typically, n ≤ 5.
Textbox
what is sparsity problems in n grams?
[]
[('what is sparsity problems in n grams?', ' Sparsity problems with n-gram Language models arise due to two issu
Executing (33s) <cell line: 20> > launch() > block_thread()


FinalProject_Ramya.ipynb
File Edit View Insert Runtime Tools Help Last saved at 6 November
+ Code + Text
Killing tunnel 127.0.0.1:7860 <> https://a9930d9d9cb15ac1fe.gradio.live
pip install google-generativeai
Collecting google-generativeai
Downloading google_generativeai-0.2.2-py3-none-any.whl (133 kB)
133.2/133.2 kB 2.8 MB/s eta 0
Collecting google-ai-generativelanguage==0.3.3 (from google-generativeai)
Downloading google_ai_generativelanguage-0.3.3-py3-none-any.whl (267 kB)


FinalProject_Ramya.ipynb
File Edit View Insert Runtime Tools Help Last saved at 6 November
+ Code + Text
[ ] from langchain.llms import GooglePalm
api_key = 'AIzaSyBchWnboRS1tQs7PRuTc4860ihrzs6aYuQ'
llm = GooglePalm(google_api_key=api_key, temperature=0.1)
[ ] from langchain.chains import RetrievalQA
from langchain.embeddings import GooglePalmEmbeddings
from langchain.llms import GooglePalm
[ ] !pip install unstructured
Requirement already satisfied: unstructured in /usr/local/lib/python3.10/dis
Requirement already satisfied: chardet in /usr/local/lib/python3.10/dist-pac
Requirement already satisfied: filetype in /usr/local/lib/python3.10/dist-pa
FinalProject_Ramya.ipynb
File Edit View Insert Runtime Tools Help Last saved at 6 November
+ Code + Text
Stored in directory: /root/.cache/pip/wheels/62/f2/10/1e606fd5f02395388f74e7462910fe851042f97238cbbd902f
Successfully built sentence-transformers
Installing collected packages: sentencepiece, sentence-transformers
Successfully installed sentence-transformers-2.2.2 sentencepiece-0.1.99
[ ] from langchain.embeddings import HuggingFaceInstructEmbeddings
# Initialize instructor embeddings using the Hugging Face model
instructor_embeddings = HuggingFaceInstructEmbeddings(model_name="hkunlp/instructor-large")
e = instructor_embeddings.embed_query("What is your name?")
Downloading (…)c7233/.gitattributes: 100% 1.48k/1.48k [00:00<00:00, 32.8kB/s]
Downloading (…)_Pooling/config.json: 100% 270/270 [00:00<00:00, 3.40kB/s]
Downloading (…)/2_Dense/config.json: 100% 116/116 [00:00<00:00, 4.93kB/s]
Downloading pytorch_model.bin: 100% 3.15M/3.15M [00:00<00:00, 23.9MB/s]

FinalProject_Ramya.ipynb ☆

File Edit View Insert Runtime Tools Help Last saved at 6 November

+ Code + Text

```
[ ] len(e)
```

```
768
```

```
[ ] e[:5]
```

```
[-0.0354379303753376,
 0.03277767822146416,
 -0.02763950638473034,
 0.01750076562166214,
 0.03374621272087097]
```

```
[ ] #pip install faiss
```

```
ERROR: Could not find a version that satisfies the requirement faiss (from versions: none)
ERROR: No matching distribution found for faiss
```

```
[ ] from langchain.vectorstores import FAISS

    # Create a FAISS instance for vector database from 'data'
    vectordb = FAISS.from_documents(documents=documents,
                                    embedding=instructor_embeddings)
```

FinalProject_Ramya.ipynb ☆

File Edit View Insert Runtime Tools Help Last saved at 6 November

Comment

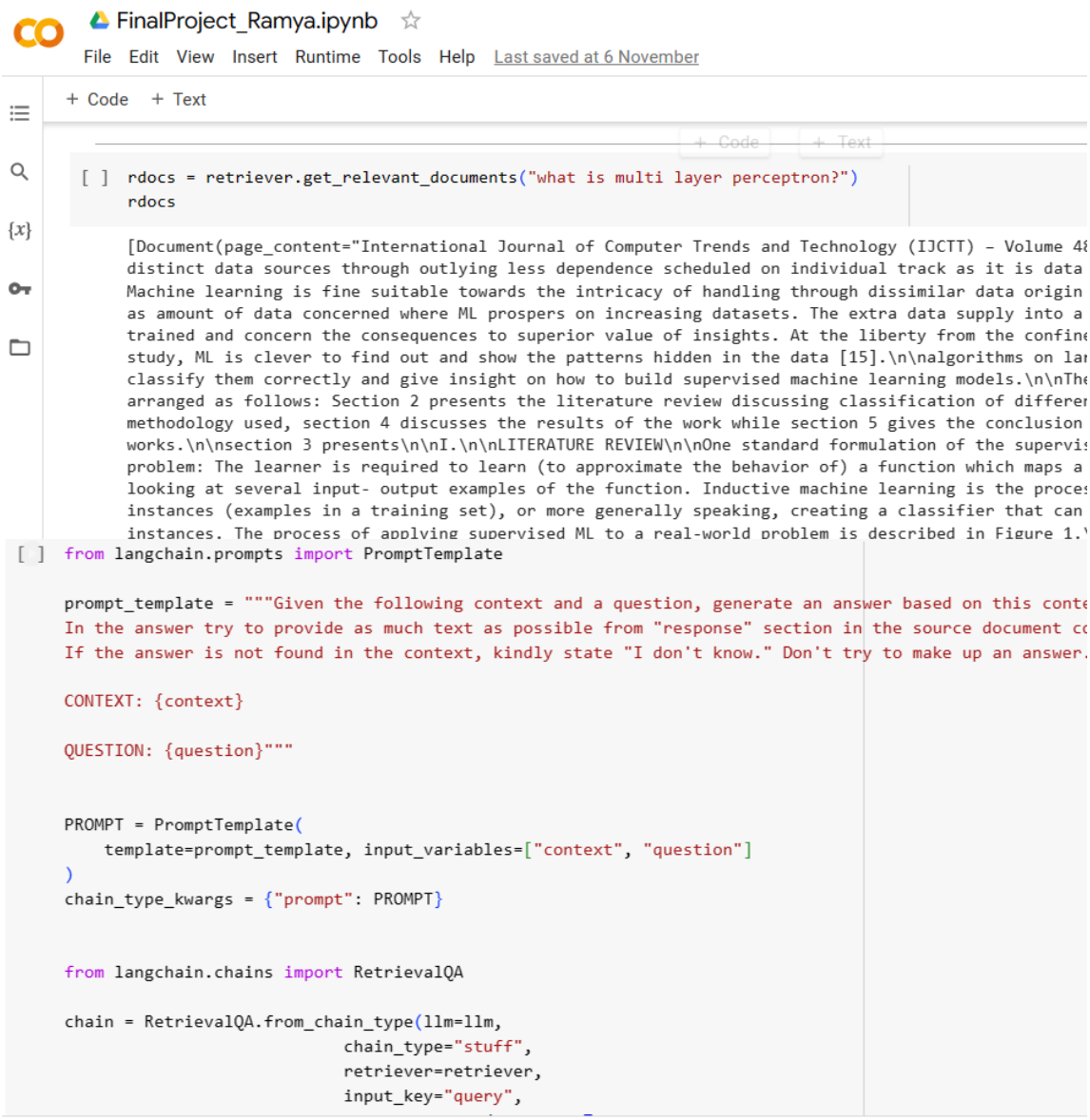
+ Code + Text

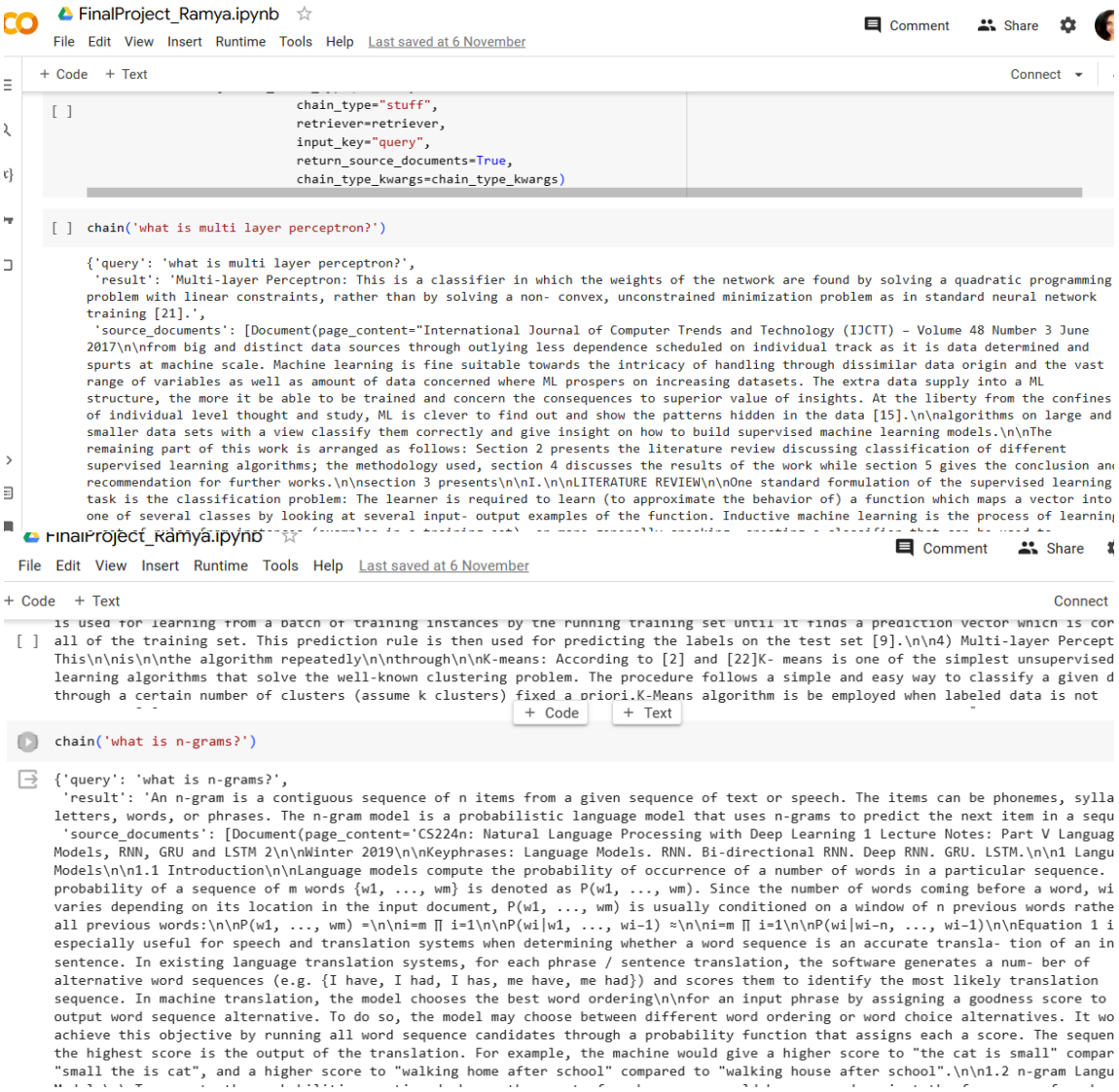
```
[ ] from langchain.vectorstores import FAISS

    # Create a FAISS instance for vector database from 'data'
    vectordb = FAISS.from_documents(documents=documents,
                                    embedding=instructor_embeddings)

    # Create a retriever for querying the vector database
    retriever = vectordb.as_retriever(score_threshold = 0.7)
```

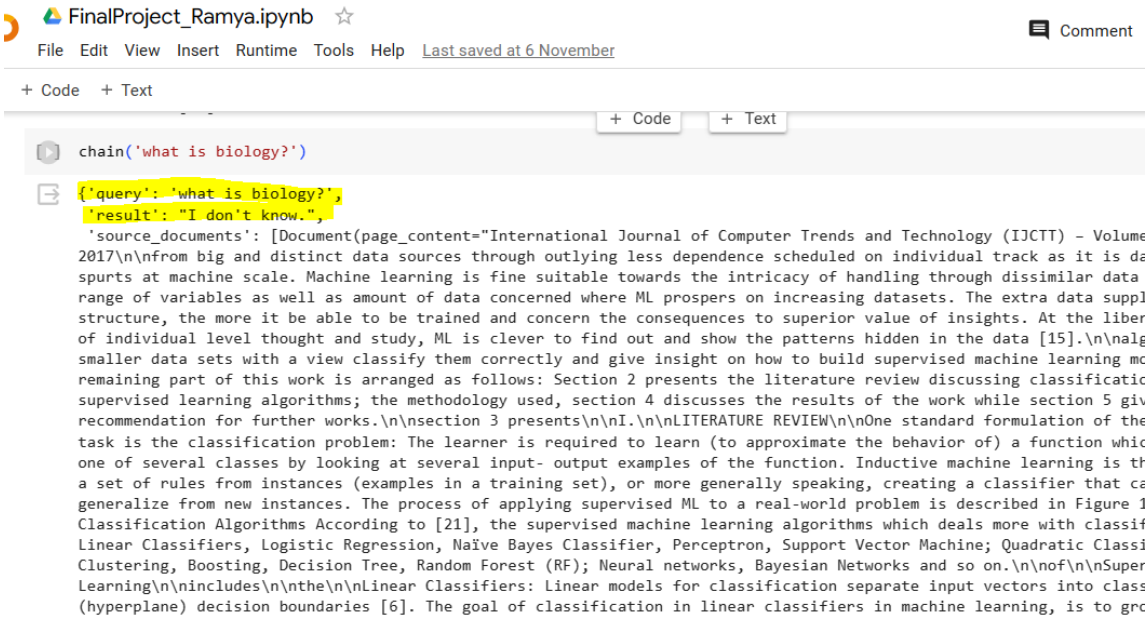
```
[ ] pip install faiss-gpu
```

```
Collecting faiss-gpu
  Downloading faiss_gpu-1.7.2-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (85.5 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 85.5/85.5 MB 10.9 MB/s eta 0:00:00
Installing collected packages: faiss-gpu
Successfully installed faiss-gpu-1.7.2
```

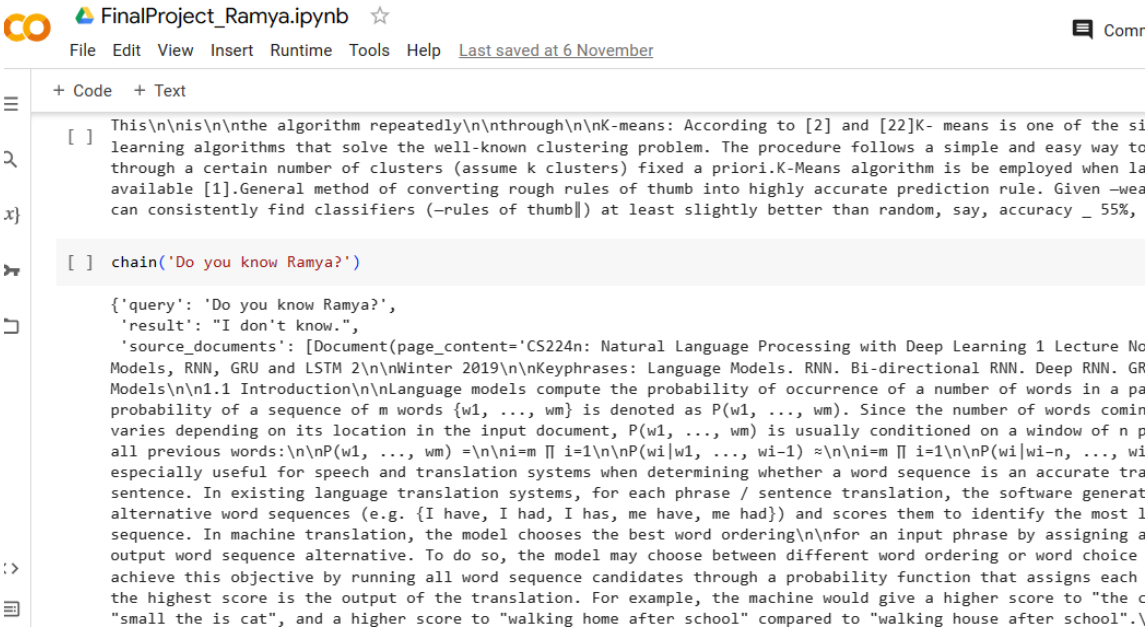
https://python.langchain.com/docs/modules/data_connection/retrievers/ get_relevant_documents is actually extracting the meaning of th questions given and not just giving the answer to the question.


FinalProject_Ramya.ipynb
File Edit View Insert Runtime Tools Help Last saved at 6 November
+ Code + Text
[ ] rdocs = retriever.get_relevant_documents("what is multi layer perceptron?")
rdocs
[Document(page_content="International Journal of Computer Trends and Technology (IJCTT) – Volume 48
distinct data sources through outlying less dependence scheduled on individual track as it is data
Machine learning is fine suitable towards the intricacy of handling through dissimilar data origin
as amount of data concerned where ML prospers on increasing datasets. The extra data supply into a
trained and concern the consequences to superior value of insights. At the liberty from the confine
study, ML is clever to find out and show the patterns hidden in the data [15].\n\nalgorithms on lar
classify them correctly and give insight on how to build supervised machine learning models.\n\nThe
arranged as follows: Section 2 presents the literature review discussing classification of differer
methodology used, section 4 discusses the results of the work while section 5 gives the conclusion
works.\n\nsection 3 presents\n\nI.\n\nLITERATURE REVIEW\n\nOne standard formulation of the supervis
problem: The learner is required to learn (to approximate the behavior of) a function which maps a
looking at several input- output examples of the function. Inductive machine learning is the proces
instances (examples in a training set), or more generally speaking, creating a classifier that can
instances. The process of applying supervised ML to a real-world problem is described in Figure 1.
[ ] from langchain.prompts import PromptTemplate
prompt_template = """Given the following context and a question, generate an answer based on this conte
In the answer try to provide as much text as possible from "response" section in the source document co
If the answer is not found in the context, kindly state "I don't know." Don't try to make up an answer.
CONTEXT: {context}
QUESTION: {question}"""
PROMPT = PromptTemplate(
template=prompt_template, input_variables=["context", "question"]
)
chain_type_kwargs = {"prompt": PROMPT}
from langchain.chains import RetrievalQA
chain = RetrievalQA.from_chain_type(llm=llm,
chain_type="stuff",
retriever=retriever,
input_key="query",


FinalProject_Ramya.ipynb
File Edit View Insert Runtime Tools Help Last saved at 6 November
Comment
Share
Connect
+ Code + Text
chain_type="stuff",
retriever=retriever,
input_key="query",
return_source_documents=True,
chain_type_kwargs=chain_type_kwargs)
chain('what is multi layer perceptron?')
{'query': 'what is multi layer perceptron?',
'result': 'Multi-layer Perceptron: This is a classifier in which the weights of the network are found by solving a quadratic programming problem with linear constraints, rather than by solving a non- convex, unconstrained minimization problem as in standard neural network training [21].',
'source_documents': [Document(page_content="International Journal of Computer Trends and Technology (IJCTT) – Volume 48 Number 3 June 2017\n\nfrom big and distinct data sources through outlying less dependence scheduled on individual track as it is data determined and spurts at machine scale. Machine learning is fine suitable towards the intricacy of handling through dissimilar data origin and the vast range of variables as well as amount of data concerned where ML prospers on increasing datasets. The extra data supply into a ML structure, the more it be able to be trained and concern the consequences to superior value of insights. At the liberty from the confines of individual level thought and study, ML is clever to find out and show the patterns hidden in the data [15].\n\nalgorithms on large and smaller data sets with a view classify them correctly and give insight on how to build supervised machine learning models.\n\nThe remaining part of this work is arranged as follows: Section 2 presents the literature review discussing classification of different supervised learning algorithms; the methodology used, section 4 discusses the results of the work while section 5 gives the conclusion an recommendation for further works.\n\nsection 3 presents\n\nI.\n\nLITERATURE REVIEW\n\nOne standard formulation of the supervised learning task is the classification problem: The learner is required to learn (to approximate the behavior of) a function which maps a vector into one of several classes by looking at several input- output examples of the function. Inductive machine learning is the process of learnin
FinalProject_Ramya.ipynb
File Edit View Insert Runtime Tools Help Last saved at 6 November
Comment
Share
Connect
+ Code + Text
is used for learning from a batch of training instances by the running training set until it finds a prediction vector which is cor
all of the training set. This prediction rule is then used for predicting the labels on the test set [9].\n\n4) Multi-layer Percept
This\n\nis\n\nthe algorithm repeatedly\n\nthrough\n\nK-means: According to [2] and [22]K- means is one of the simplest unsupervised
learning algorithms that solve the well-known clustering problem. The procedure follows a simple and easy way to classify a given d
through a certain number of clusters (assume k clusters) fixed a priori.K-Means algorithm is be employed when labeled data is not
+ Code + Text
chain('what is n-grams?')
{'query': 'what is n-grams?',
'result': 'An n-gram is a contiguous sequence of n items from a given sequence of text or speech. The items can be phonemes, sylla
letters, words, or phrases. The n-gram model is a probabilistic language model that uses n-grams to predict the next item in a sequ
'source_documents': [Document(page_content='CS224n: Natural Language Processing with Deep Learning 1 Lecture Notes: Part V Languag
Models, RNN, GRU and LSTM 2\n\nWinter 2019\n\nKeyphrases: Language Models. RNN. Bi-directional RNN. Deep RNN. GRU. LSTM.\n\n1 Langu
Models\n\n1.1 Introduction\n\nLanguage models compute the probability of occurrence of a number of words in a particular sequence.
probability of a sequence of m words {w1, ..., wm} is denoted as P(w1, ..., wm). Since the number of words coming before a word, wi
varies depending on its location in the input document, P(w1, ..., wm) is usually conditioned on a window of n previous words rathe
all previous words:\n\nP(w1, ..., wm) =\n\ni=m ∏ i=1\n\nP(wi|w1, ..., wi-1) ≈\n\ni=m ∏ i=1\n\nP(wi|wi-n, ..., wi-1)\n\nEquation 1 i
especially useful for speech and translation systems when determining whether a word sequence is an accurate transla- tion of an in
sentence. In existing language translation systems, for each phrase / sentence translation, the software generates a num- ber of
alternative word sequences (e.g. {I have, I had, I has, me have, me had}) and scores them to identify the most likely translation
sequence. In machine translation, the model chooses the best word ordering\n\nfor an input phrase by assigning a goodness score to
output word sequence alternative. To do so, the model may choose between different word ordering or word choice alternatives. It wo
achieve this objective by running all word sequence candidates through a probability function that assigns each a score. The sequen
the highest score is the output of the translation. For example, the machine would give a higher score to "the cat is small" compar
"small the is cat", and a higher score to "walking home after school" compared to "walking house after school".\n\n1.2 n-gram Langu



Based on the prompt instruction it did not give the wrong answer to the question which is not given in pdf.



Chapter 7

Conclusion

Transforming Knowledge Accessibility with Domain-Specific AI-Powered PDF based ChatGPT

In the age of digital information, our project has endeavored to bridge the gap between the wealth of knowledge stored within PDF documents and its accessibility to learners, educators, professionals, and enthusiasts. The culmination of my efforts has resulted in the development of a Domain-Specific AI-Powered PDF ChatGPT, a transformative solution poised to revolutionize the way we interact with and extract value from digital content.

Throughout this project, I have faced and addressed several key challenges. The formidable task of efficiently extracting knowledge from PDF documents, especially as their volume continues to grow, was met with innovative solutions. Our AI knowledge extraction engine, equipped with natural language processing and machine learning capabilities, demonstrates the potential to unlock insights from text, graphics, and multimedia content, providing users with a seamless and enriched learning experience.

Moreover, my commitment to making specialized knowledge accessible has borne fruit. Fields such as Artificial Intelligence, often regarded as esoteric, have become open books for learners with varying levels of expertise. Our domain-specific modules empower individuals to delve into advanced topics, enriching their understanding and potentially expanding their career horizons.

In conclusion, our Domain-Specific AI-Powered PDF ChatGPT stands as a testament to innovation and determination. It embodies our commitment to democratizing knowledge, making it accessible to all, regardless of their background or expertise. With this project, I have taken a significant step toward a future where knowledge is not confined within the pages of PDF documents but flows freely to empower individuals and enrich collective understanding. The journey continues, as I look forward to further advancements and opportunities in the realm of AI-driven education and information accessibility.

References

1. GPT-4 and LangChain: Building Python Chatbot with PDF Integration | Next Idea Tech Blog

2. Question Answering | 🦜🔗 Langchain

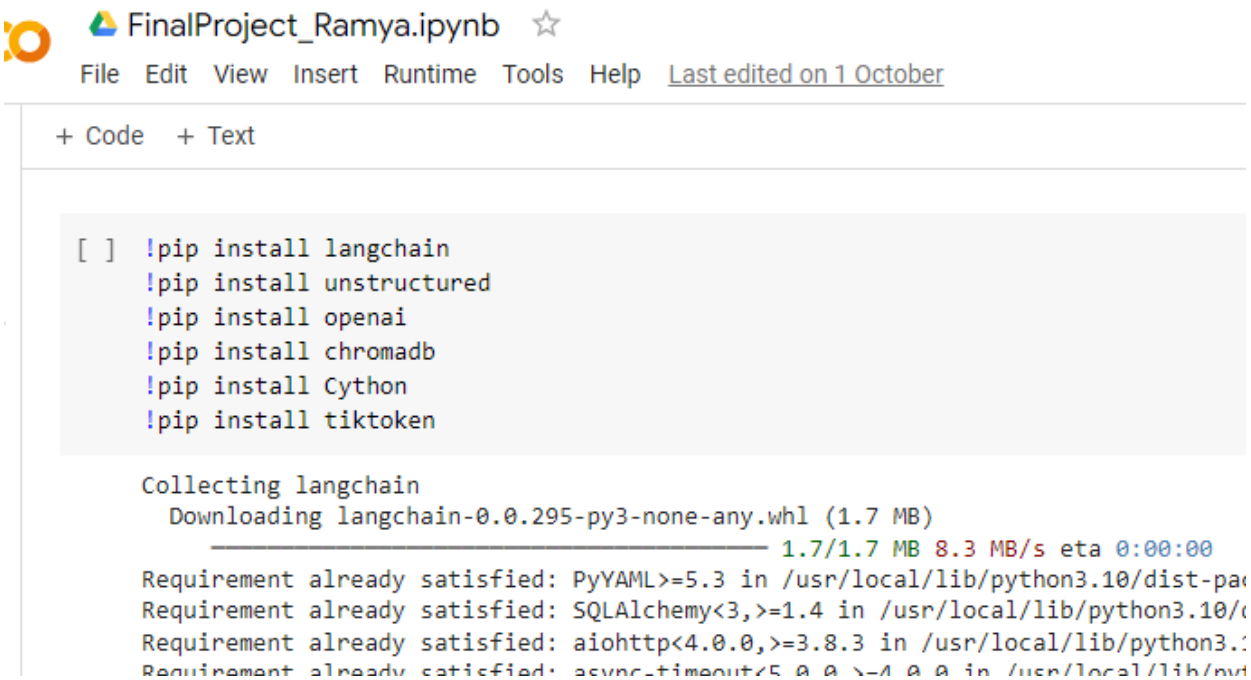
3. Guide to Chroma DB | A Vector Store for Your Generative AI LLMs (analyticsvidhya.com)

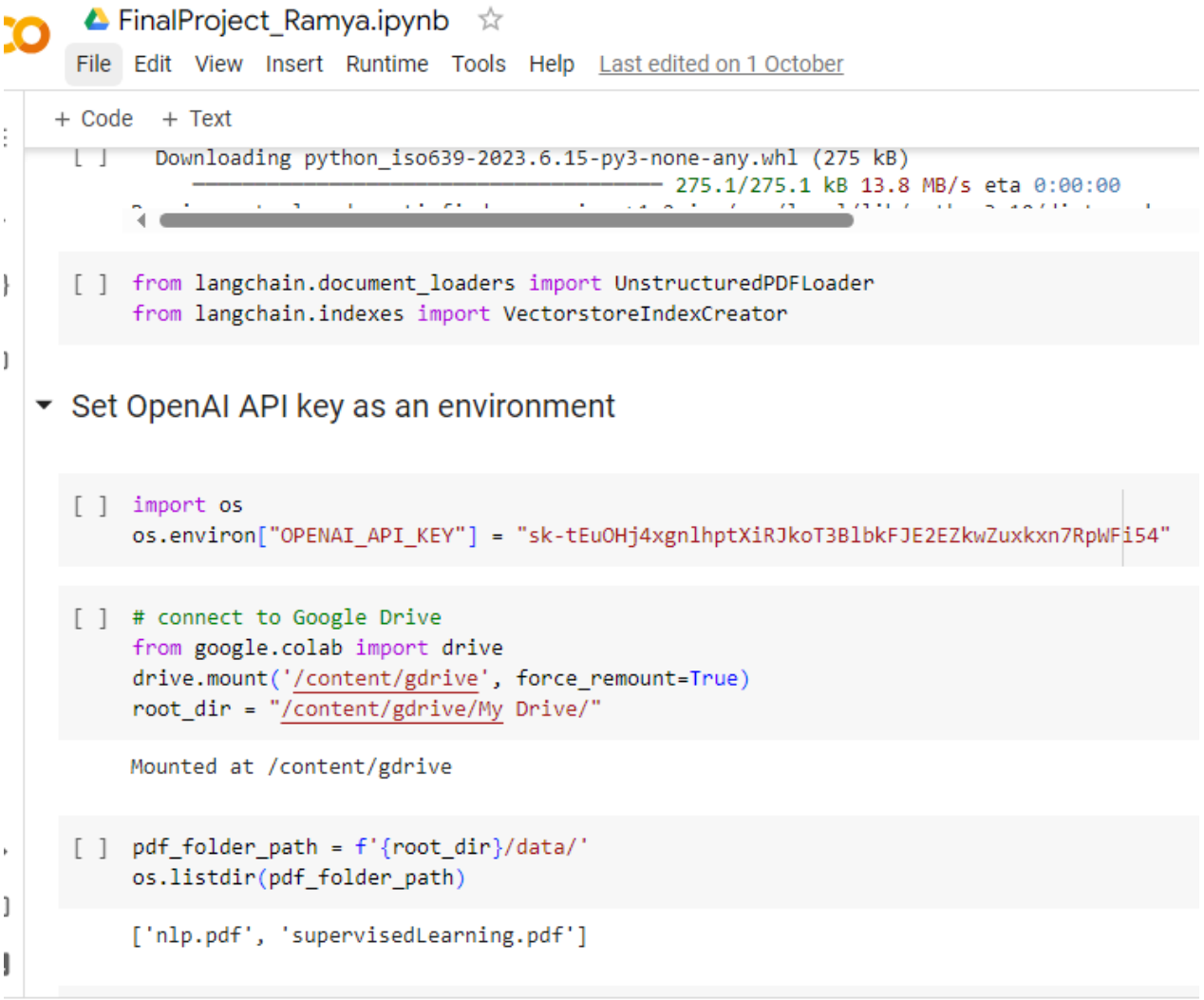
4. Chroma | 🦜🔗 Langchain & Pinecone | 🦜🔗 LangchainInstructEmbeddings | 🦜🔗 Langchain

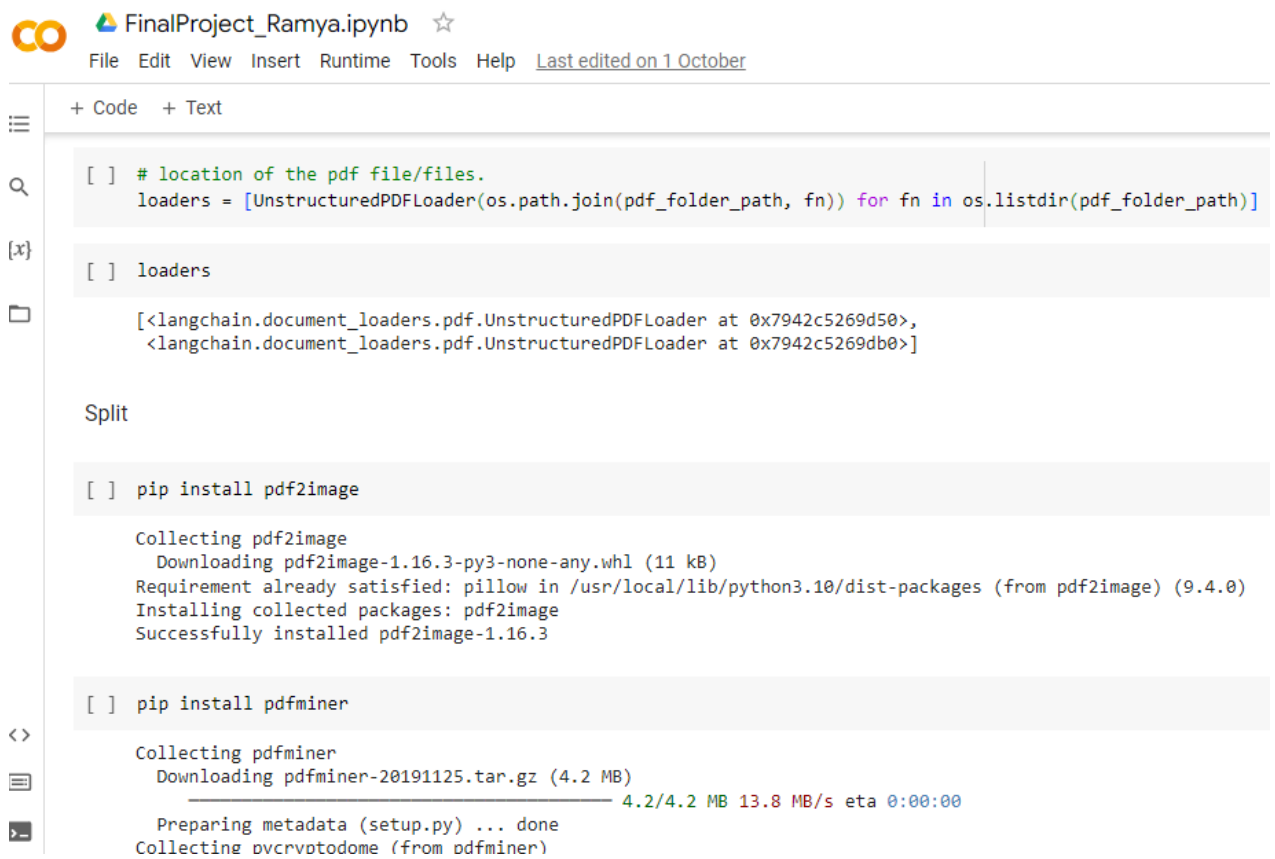
5. Langchain Documentation: https://langchain.org/docs & OpenAI Documentation: https://openai.com/docs langchain/llms/googlepalm | 🦜🔗 Langchain

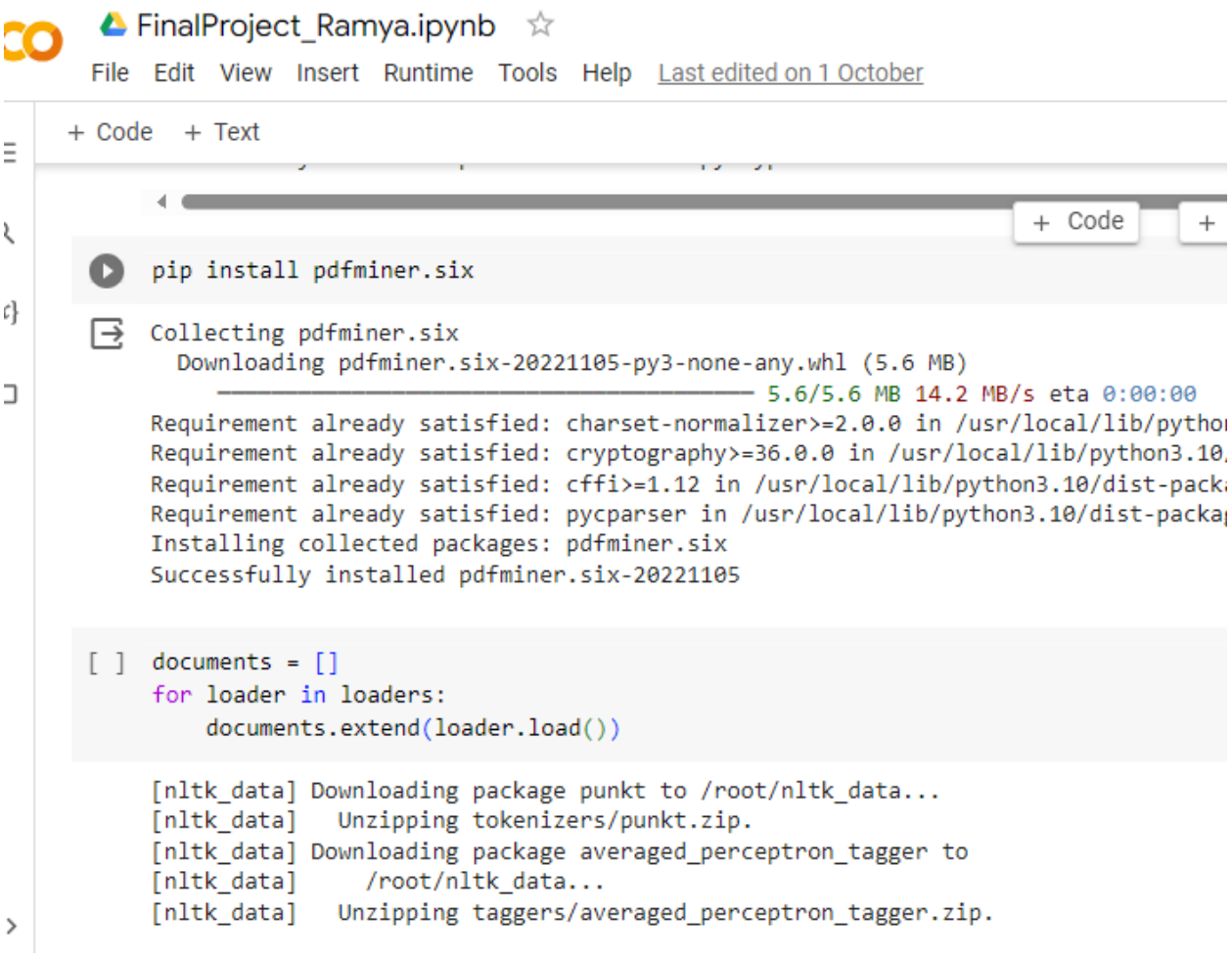
Appendix A

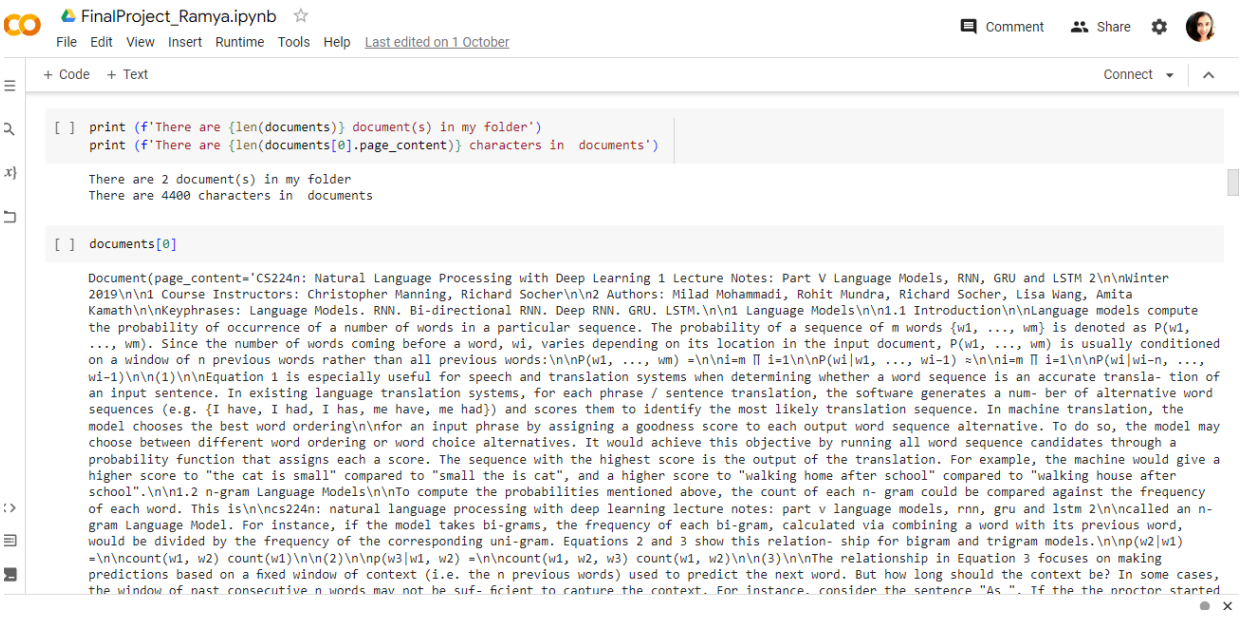
Source code

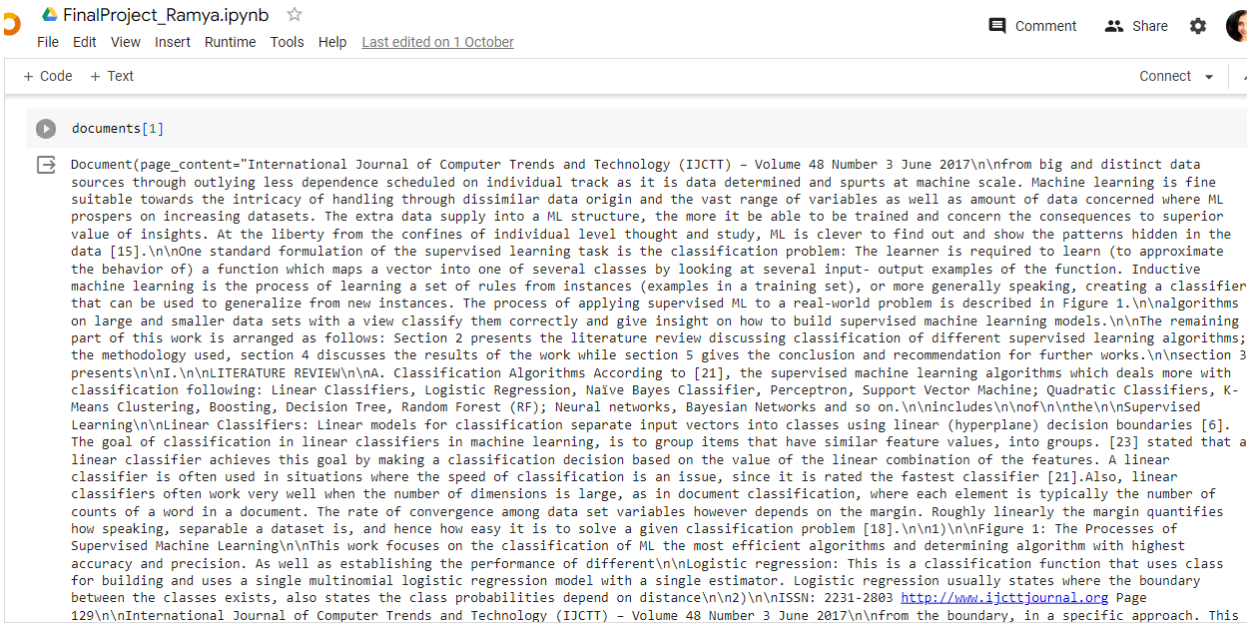



FinalProject_Ramya.ipynb
File Edit View Insert Runtime Tools Help Last edited on 1 October
+ Code + Text
Downloading python_iso639-2023.6.15-py3-none-any.whl (275 kB)
275.1/275.1 kB 13.8 MB/s eta 0:00:00
from langchain.document_loaders import UnstructuredPDFLoader
from langchain.indexes import VectorstoreIndexCreator
Set OpenAI API key as an environment
import os
os.environ["OPENAI_API_KEY"] = "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"
# connect to Google Drive
from google.colab import drive
drive.mount('/content/gdrive', force_remount=True)
root_dir = "/content/gdrive/My Drive/"
Mounted at /content/gdrive
pdf_folder_path = f'{root_dir}/data/'
os.listdir(pdf_folder_path)
['nlp.pdf', 'supervisedLearning.pdf']

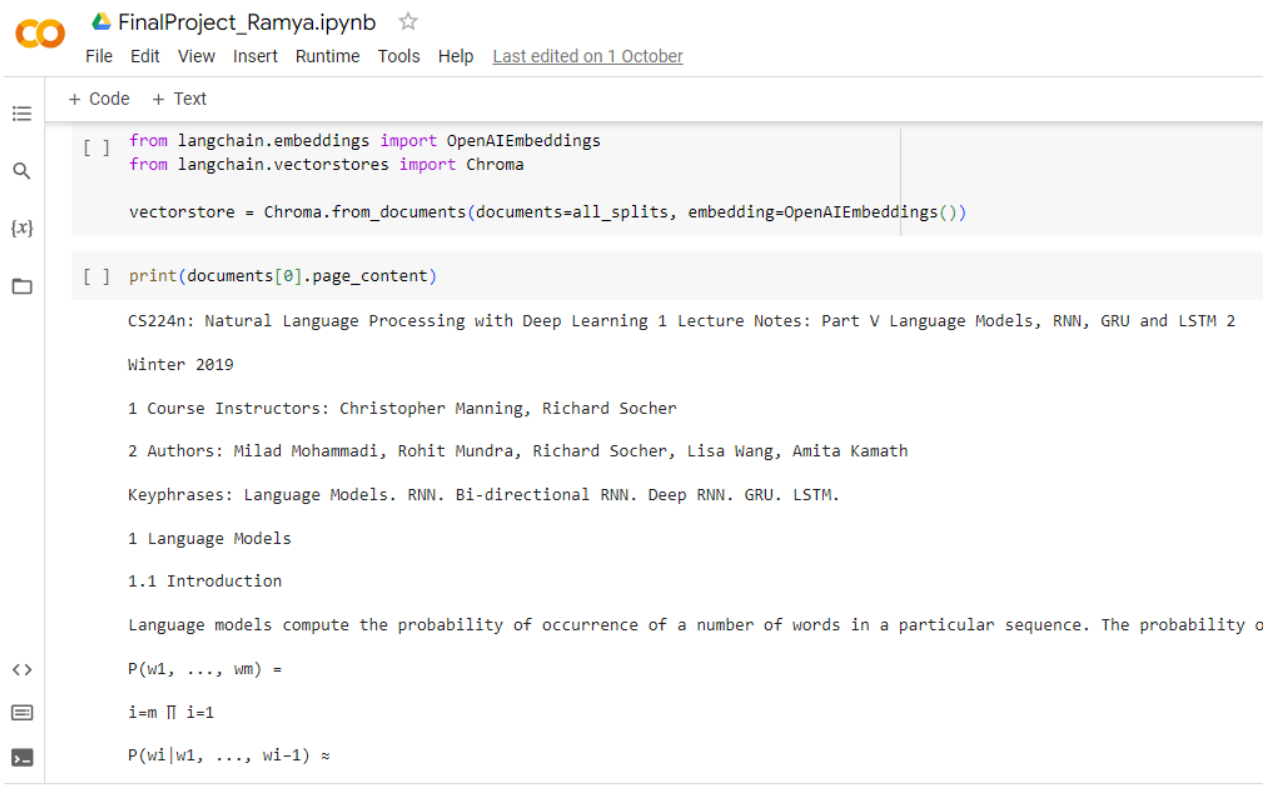

FinalProject_Ramya.ipynb
File Edit View Insert Runtime Tools Help Last edited on 1 October
+ Code + Text
[ ] # location of the pdf file/files.
loaders = [UnstructuredPDFLoader(os.path.join(pdf_folder_path, fn)) for fn in os.listdir(pdf_folder_path)]
[ ] loaders
[<langchain.document_loaders.pdf.UnstructuredPDFLoader at 0x7942c5269d50>,
<langchain.document_loaders.pdf.UnstructuredPDFLoader at 0x7942c5269db0>]
Split
[ ] pip install pdf2image
Collecting pdf2image
Downloading pdf2image-1.16.3-py3-none-any.whl (11 kB)
Requirement already satisfied: pillow in /usr/local/lib/python3.10/dist-packages (from pdf2image) (9.4.0)
Installing collected packages: pdf2image
Successfully installed pdf2image-1.16.3
[ ] pip install pdfminer
Collecting pdfminer
Downloading pdfminer-20191125.tar.gz (4.2 MB)
4.2/4.2 MB 13.8 MB/s eta 0:00:00
Preparing metadata (setup.py) ... done
Collecting pycryptodome (from pdfminer)

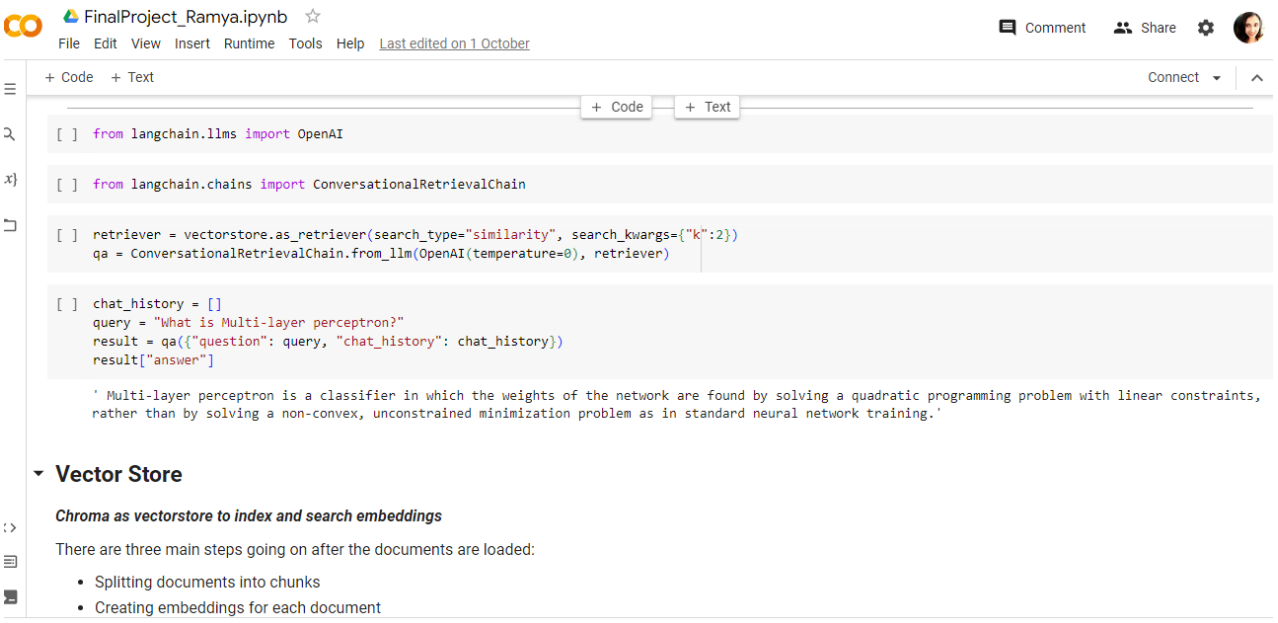

FinalProject_Ramya.ipynb
File Edit View Insert Runtime Tools Help Last edited on 1 October
+ Code + Text
+ Code
pip install pdfminer.six
Collecting pdfminer.six
Downloading pdfminer.six-20221105-py3-none-any.whl (5.6 MB)
5.6/5.6 MB 14.2 MB/s eta 0:00:00
Requirement already satisfied: charset-normalizer>=2.0.0 in /usr/local/lib/python
Requirement already satisfied: cryptography>=36.0.0 in /usr/local/lib/python3.10/
Requirement already satisfied: cffi>=1.12 in /usr/local/lib/python3.10/dist-packa
Requirement already satisfied: pycparser in /usr/local/lib/python3.10/dist-packa
Installing collected packages: pdfminer.six
Successfully installed pdfminer.six-20221105
[ ] documents = []
for loader in loaders:
documents.extend(loader.load())
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data] Unzipping tokenizers/punkt.zip.
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data] /root/nltk_data...
[nltk_data] Unzipping taggers/averaged_perceptron_tagger.zip.

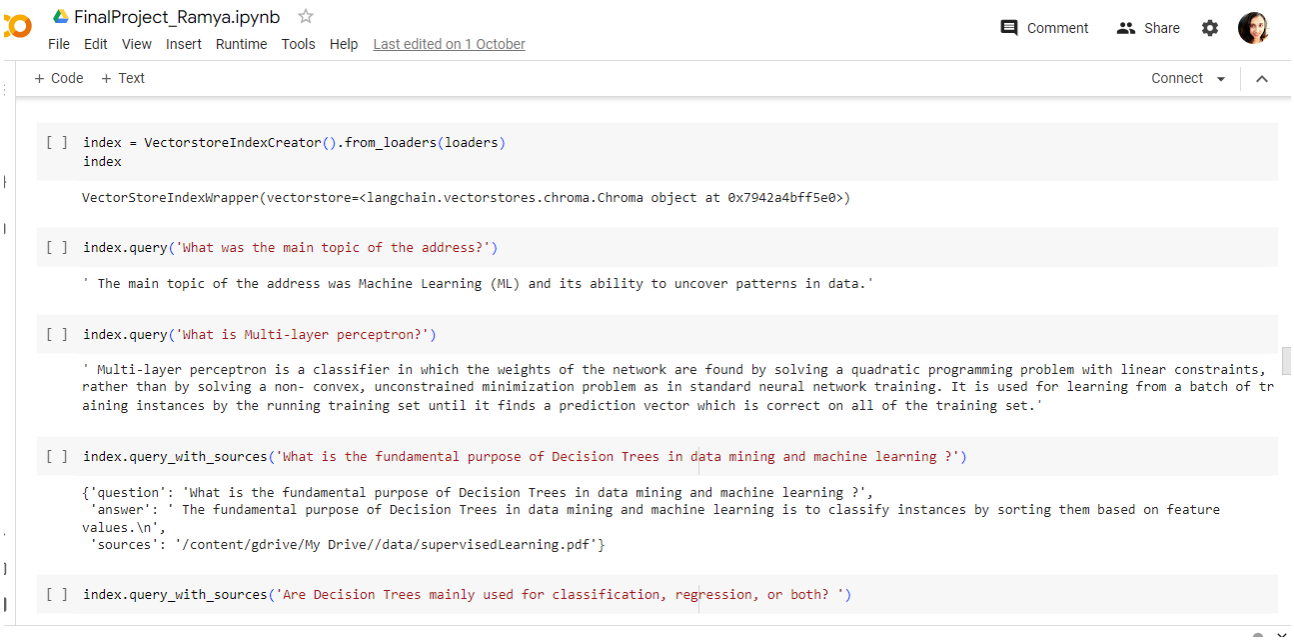

FinalProject_Ramya.ipynb
File Edit View Insert Runtime Tools Help Last edited on 1 October
Comment Share
+ Code + Text
Connect
[ ] print (f'There are {len(documents)} document(s) in my folder')
print (f'There are {len(documents[0].page_content)} characters in documents')
There are 2 document(s) in my folder
There are 4400 characters in documents
[ ] documents[0]
Document(page_content='CS224n: Natural Language Processing with Deep Learning 1 Lecture Notes: Part V Language Models, RNN, GRU and LSTM 2\n\nWinter 2019\n\n1 Course Instructors: Christopher Manning, Richard Socher\n\n2 Authors: Milad Mohammadi, Rohit Mundra, Richard Socher, Lisa Wang, Amita Kamath\n\nKeyphrases: Language Models. RNN. Bi-directional RNN. Deep RNN. GRU. LSTM.\n\n1 Language Models\n\n1.1 Introduction\n\nLanguage models compute the probability of occurrence of a number of words in a particular sequence. The probability of a sequence of m words {w1, ..., wm} is denoted as P(w1, ..., wm). Since the number of words coming before a word, wi, varies depending on its location in the input document, P(w1, ..., wm) is usually conditioned on a window of n previous words rather than all previous words:\n\nP(w1, ..., wm) =\n\ni=m ∏ i=1\n\nP(wi|w1, ..., wi-1) ≈\n\ni=m ∏ i=1\n\nP(wi|wi-n, ..., wi-1)\n\n(1)\n\nEquation 1 is especially useful for speech and translation systems when determining whether a word sequence is an accurate transla- tion of an input sentence. In existing language translation systems, for each phrase / sentence translation, the software generates a num- ber of alternative word sequences (e.g. {I have, I had, I has, me have, me had}) and scores them to identify the most likely translation sequence. In machine translation, the model chooses the best word ordering\n\nfor an input phrase by assigning a goodness score to each output word sequence alternative. To do so, the model may choose between different word ordering or word choice alternatives. It would achieve this objective by running all word sequence candidates through a probability function that assigns each a score. The sequence with the highest score is the output of the translation. For example, the machine would give a higher score to "the cat is small" compared to "small the is cat", and a higher score to "walking home after school" compared to "walking house after school".\n\n1.2 n-gram Language Models\n\nTo compute the probabilities mentioned above, the count of each n- gram could be compared against the frequency of each word. This is\n\ncs224n: natural language processing with deep learning lecture notes: part v language models, rnn, gru and lstm 2\n\ncalled an n-gram Language Model. For instance, if the model takes bi-grams, the frequency of each bi-gram, calculated via combining a word with its previous word, would be divided by the frequency of the corresponding uni-gram. Equations 2 and 3 show this relation- ship for bigram and trigram models.\n\np(w2|w1) =\n\ncount(w1, w2) count(w1)\n\n(2)\n\np(w3|w1, w2) =\n\ncount(w1, w2, w3) count(w1, w2)\n\n(3)\n\nThe relationship in Equation 3 focuses on making predictions based on a fixed window of context (i.e. the n previous words) used to predict the next word. But how long should the context be? In some cases, the window of past consecutive n words may not be suf- ficient to capture the context. For instance, consider the sentence "As ". If the the proctor started

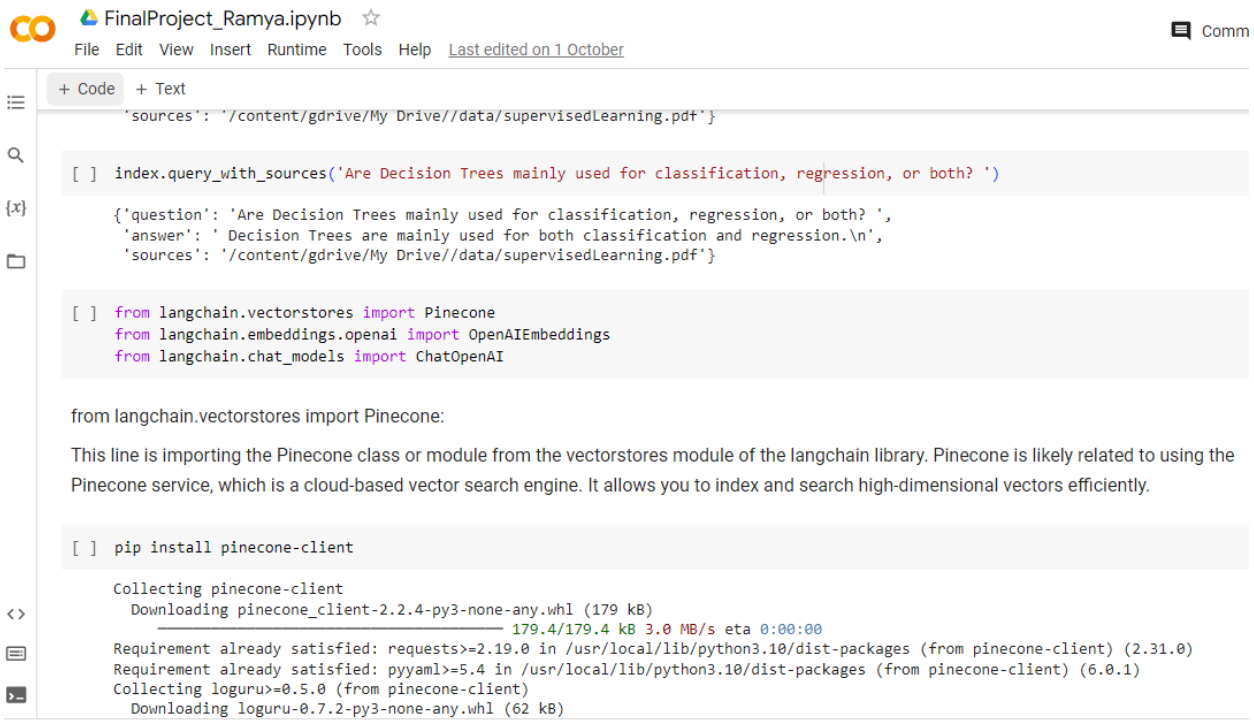



```
[ ] from langchain.text_splitter import RecursiveCharacterTextSplitter

    text_splitter = RecursiveCharacterTextSplitter(chunk_size = 500, chunk_overlap = 0)
    all_splits = text_splitter.split_documents(documents)
```



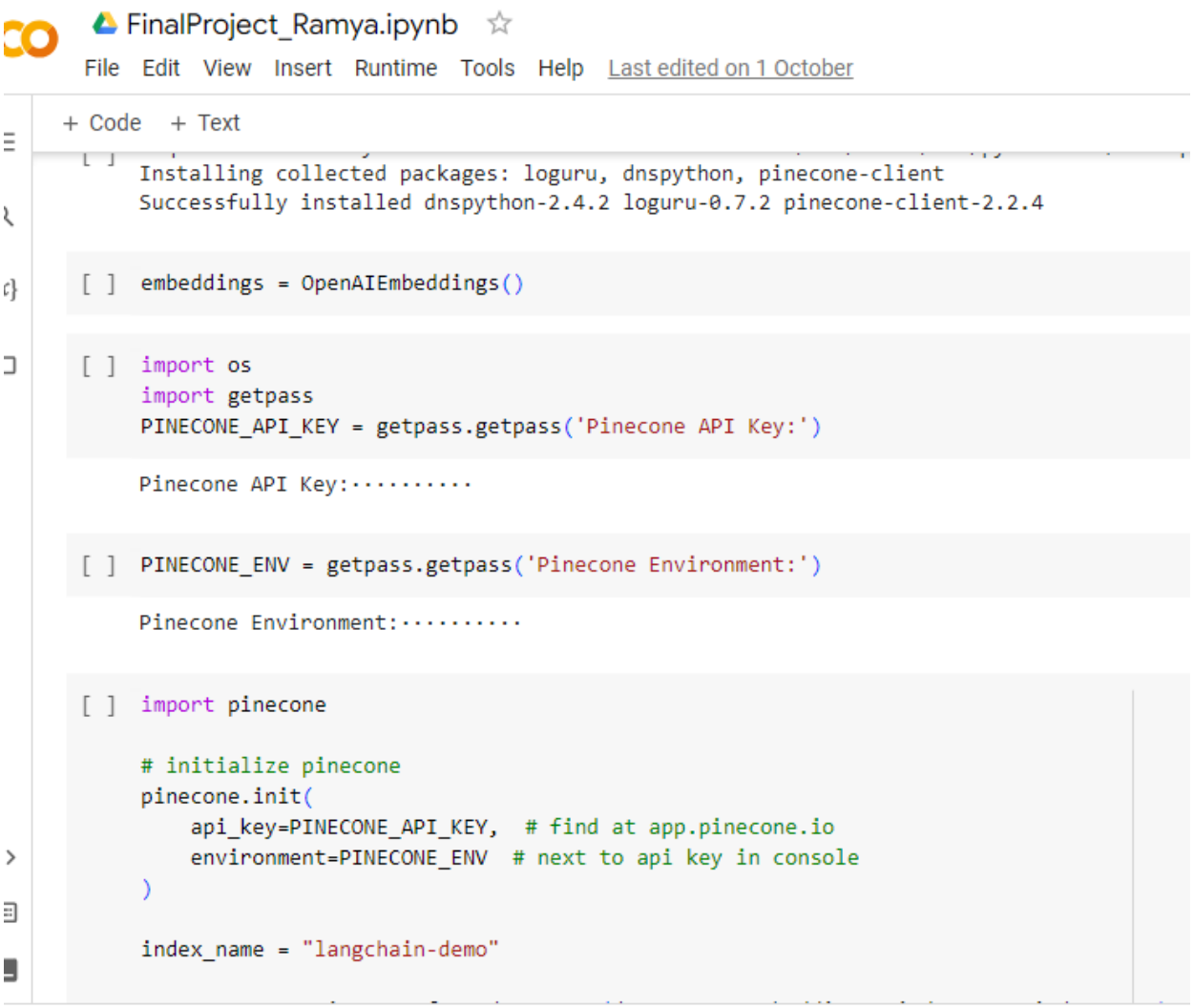

FinalProject_Ramya.ipynb
File Edit View Insert Runtime Tools Help Last edited on 1 October
Comment
Share
+ Code + Text
Connect
+ Code + Text
[ ] from langchain.llms import OpenAI
[ ] from langchain.chains import ConversationalRetrievalChain
[ ] retriever = vectorstore.as_retriever(search_type="similarity", search_kwargs={"k":2})
qa = ConversationalRetrievalChain.from_llm(OpenAI(temperature=0), retriever)
[ ] chat_history = []
query = "What is Multi-layer perceptron?"
result = qa({"question": query, "chat_history": chat_history})
result["answer"]
' Multi-layer perceptron is a classifier in which the weights of the network are found by solving a quadratic programming problem with linear constraints, rather than by solving a non-convex, unconstrained minimization problem as in standard neural network training.'
Vector Store
Chroma as vectorstore to index and search embeddings
There are three main steps going on after the documents are loaded:
Splitting documents into chunks
Creating embeddings for each document

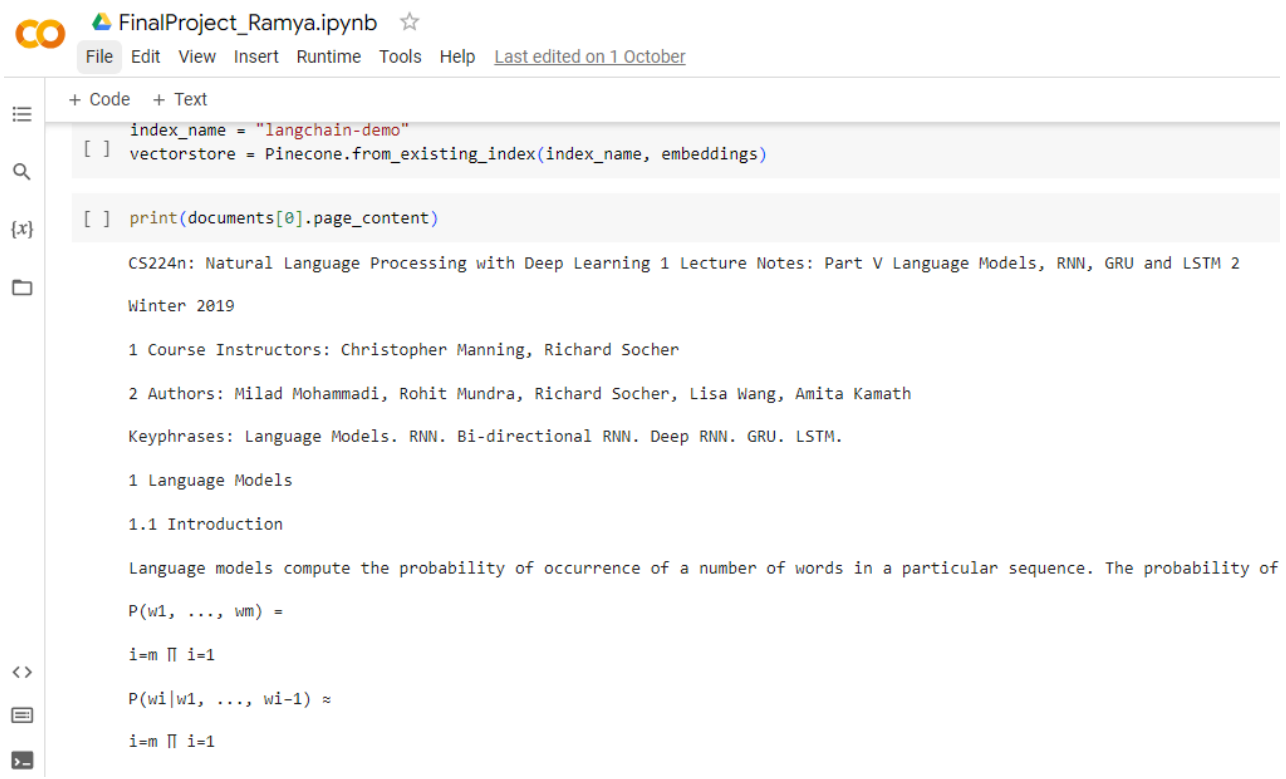

FinalProject_Ramya.ipynb
File Edit View Insert Runtime Tools Help Last edited on 1 October
Comment
Share
+ Code + Text
Connect
[ ] index = VectorstoreIndexCreator().from_loaders(loaders)
index
VectorStoreIndexWrapper(vectorstore=<langchain.vectorstores.chroma.Chroma object at 0x7942a4bff5e0>)
[ ] index.query('What was the main topic of the address?')
' The main topic of the address was Machine Learning (ML) and its ability to uncover patterns in data.'
[ ] index.query('What is Multi-layer perceptron?')
' Multi-layer perceptron is a classifier in which the weights of the network are found by solving a quadratic programming problem with linear constraints, rather than by solving a non- convex, unconstrained minimization problem as in standard neural network training. It is used for learning from a batch of tr aining instances by the running training set until it finds a prediction vector which is correct on all of the training set.'
[ ] index.query_with_sources('What is the fundamental purpose of Decision Trees in data mining and machine learning ?')
{'question': 'What is the fundamental purpose of Decision Trees in data mining and machine learning ?',
'answer': ' The fundamental purpose of Decision Trees in data mining and machine learning is to classify instances by sorting them based on feature values.\n',
'sources': '/content/gdrive/My Drive//data/supervisedLearning.pdf'}
[ ] index.query_with_sources('Are Decision Trees mainly used for classification, regression, or both? ')

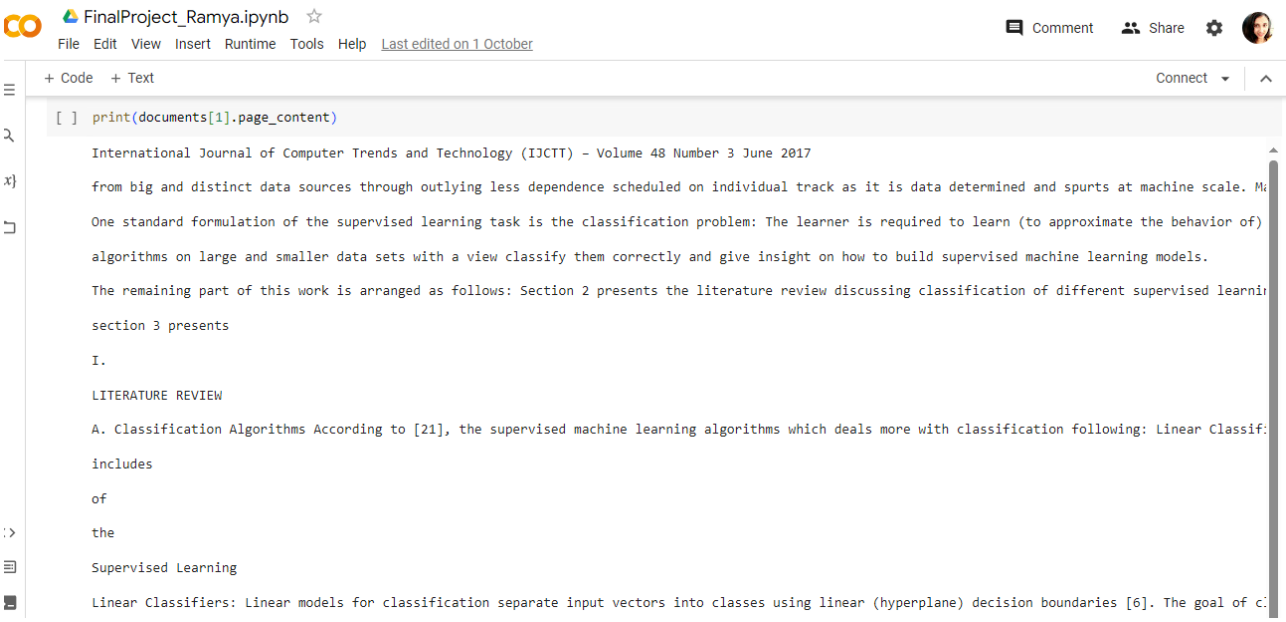


from langchain.vectorstores import Pinecone:

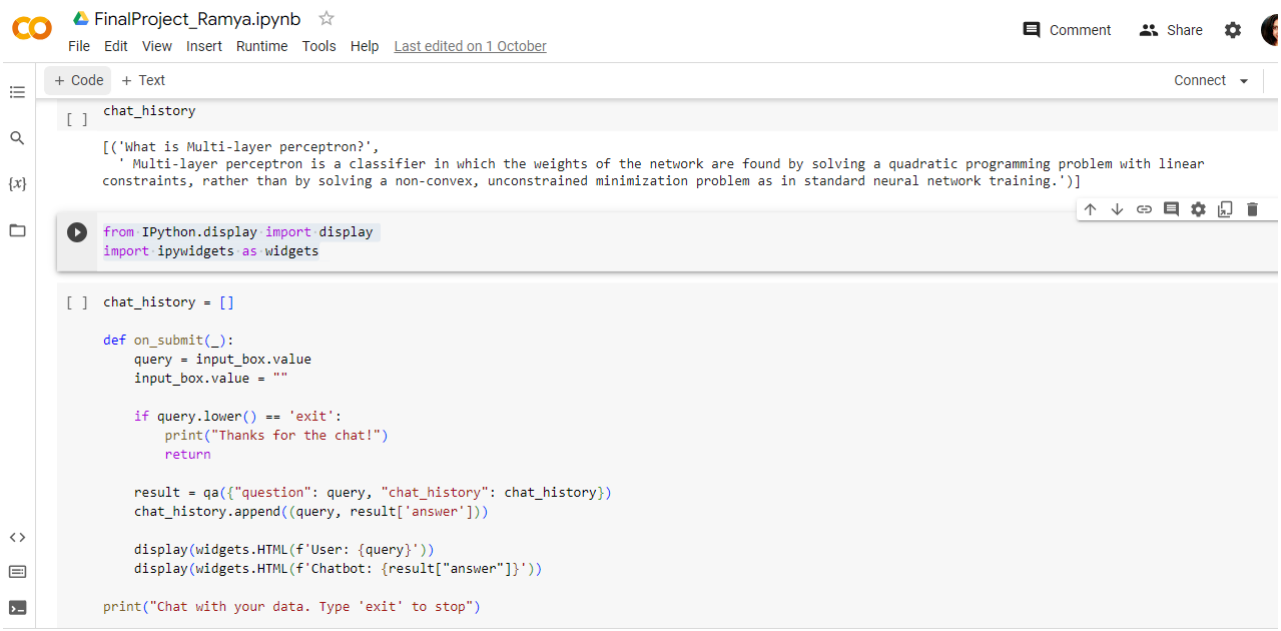
This line is importing the Pinecone class or module from the vectorstores module of the langchain library. Pinecone is likely related to using the Pinecone service, which is a cloud-based vector search engine. It allows you to index and search high-dimensional vectors efficiently.

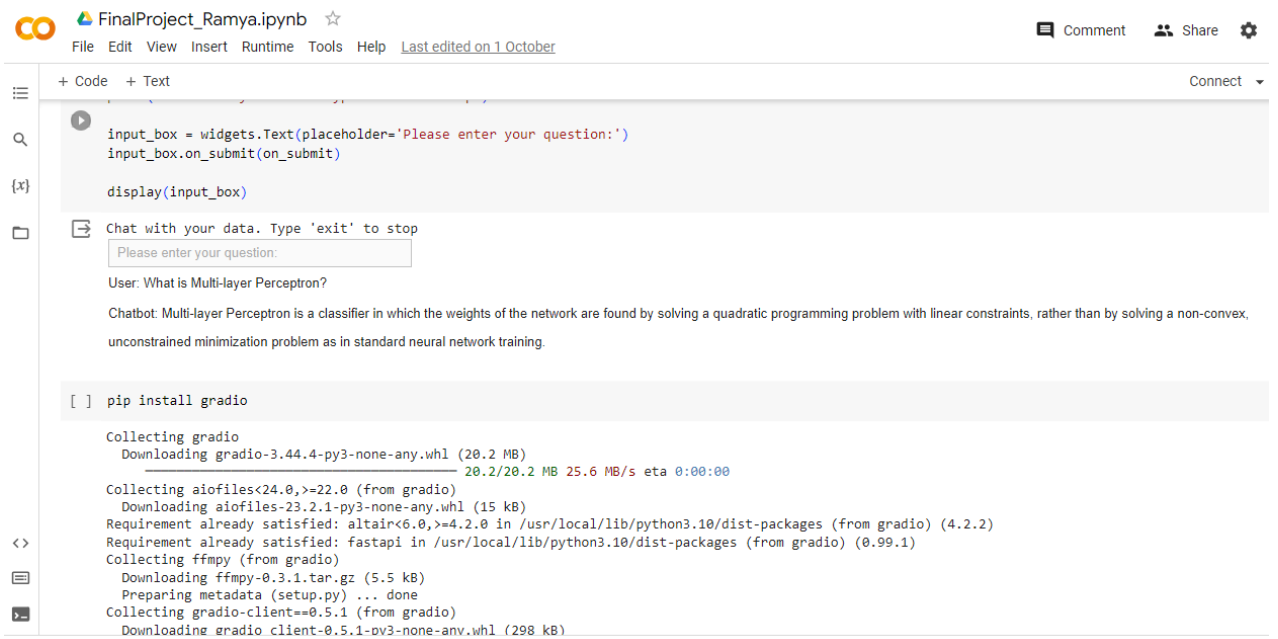

FinalProject_Ramya.ipynb
File Edit View Insert Runtime Tools Help Last edited on 1 October
+ Code + Text
Installing collected packages: loguru, dnspython, pinecone-client
Successfully installed dnspython-2.4.2 loguru-0.7.2 pinecone-client-2.2.4
embeddings = OpenAIEmbeddings()
import os
import getpass
PINECONE_API_KEY = getpass.getpass('Pinecone API Key:')
Pinecone API Key:
PINECONE_ENV = getpass.getpass('Pinecone Environment:')
Pinecone Environment:
import pinecone
# initialize pinecone
pinecone.init(
api_key=PINECONE_API_KEY, # find at app.pinecone.io
environment=PINECONE_ENV # next to api key in console
)
index_name = "langchain-demo"


FinalProject_Ramya.ipynb
File Edit View Insert Runtime Tools Help Last edited on 1 October
+ Code + Text
index_name = "langchain-demo"
vectorstore = Pinecone.from_existing_index(index_name, embeddings)
print(documents[0].page_content)
CS224n: Natural Language Processing with Deep Learning 1 Lecture Notes: Part V Language Models, RNN, GRU and LSTM 2
Winter 2019
1 Course Instructors: Christopher Manning, Richard Socher
2 Authors: Milad Mohammadi, Rohit Mundra, Richard Socher, Lisa Wang, Amita Kamath
Keyphrases: Language Models. RNN. Bi-directional RNN. Deep RNN. GRU. LSTM.
1 Language Models
1.1 Introduction
Language models compute the probability of occurrence of a number of words in a particular sequence. The probability of
P(w1, ..., wm) =
i=m ∏ i=1
P(wi|w1, ..., wi-1) ≈
i=m ∏ i=1

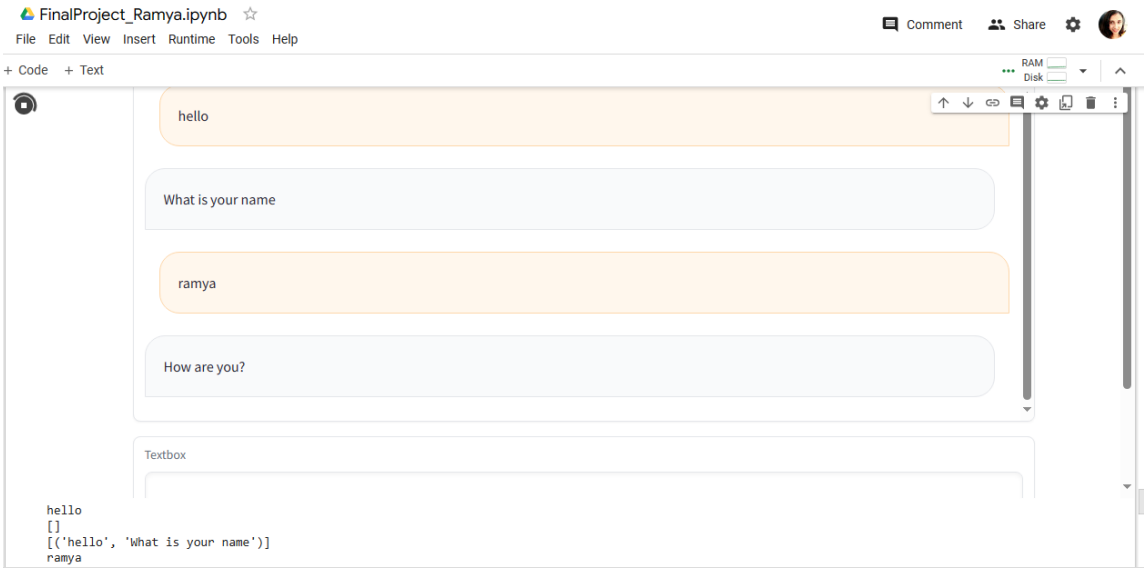

FinalProject_Ramya.ipynb
File Edit View Insert Runtime Tools Help Last edited on 1 October
Comment Share
+ Code + Text
Connect
print(documents[1].page_content)
International Journal of Computer Trends and Technology (IJCTT) – Volume 48 Number 3 June 2017
from big and distinct data sources through outlying less dependence scheduled on individual track as it is data determined and spurts at machine scale. M
One standard formulation of the supervised learning task is the classification problem: The learner is required to learn (to approximate the behavior of)
algorithms on large and smaller data sets with a view classify them correctly and give insight on how to build supervised machine learning models.
The remaining part of this work is arranged as follows: Section 2 presents the literature review discussing classification of different supervised learni
section 3 presents
I.
LITERATURE REVIEW
A. Classification Algorithms According to [21], the supervised machine learning algorithms which deals more with classification following: Linear Classif
includes
of
the
Supervised Learning
Linear Classifiers: Linear models for classification separate input vectors into classes using linear (hyperplane) decision boundaries [6]. The goal of c


FinalProject_Ramya.ipynb
File Edit View Insert Runtime Tools Help Last edited on 1 October
Comment
Share
Connect
+ Code + Text
the algorithm repeatedly
[ ] retriever = vectorstore.as_retriever(search_type="similarity", search_kwargs={"k":2})
qa = ConversationalRetrievalChain.from_llm(OpenAI(temperature=0), retriever)
[ ] chat_history = []
query = "What is Multi-layer perceptron?"
result = qa({"question": query, "chat_history": chat_history})
result["answer"]
' Multi-layer perceptron is a classifier in which the weights of the network are found by solving a quadratic programming problem with linear constraints, rather than by solving a non-convex, unconstrained minimization problem as in standard neural network training.'
[ ] chat_history.append((query, result["answer"]))
chat_history
[('What is Multi-layer perceptron?',
' Multi-layer perceptron is a classifier in which the weights of the network are found by solving a quadratic programming problem with linear constraints, rather than by solving a non-convex, unconstrained minimization problem as in standard neural network training.')]
from IPython.display import display
import ipywidgets as widgets
[ ] chat_history = []


FinalProject_Ramya.ipynb
File Edit View Insert Runtime Tools Help Last edited on 1 October
Comment
Share
Connect
+ Code + Text
[ ] chat_history
[('What is Multi-layer perceptron?',
' Multi-layer perceptron is a classifier in which the weights of the network are found by solving a quadratic programming problem with linear constraints, rather than by solving a non-convex, unconstrained minimization problem as in standard neural network training.')]
from IPython.display import display
import ipywidgets as widgets
[ ] chat_history = []
def on_submit(_):
query = input_box.value
input_box.value = ""
if query.lower() == 'exit':
print("Thanks for the chat!")
return
result = qa({"question": query, "chat_history": chat_history})
chat_history.append((query, result['answer']))
display(widgets.HTML(f'User: {query}'))
display(widgets.HTML(f'Chatbot: {result["answer"]}'))
print("Chat with your data. Type 'exit' to stop")

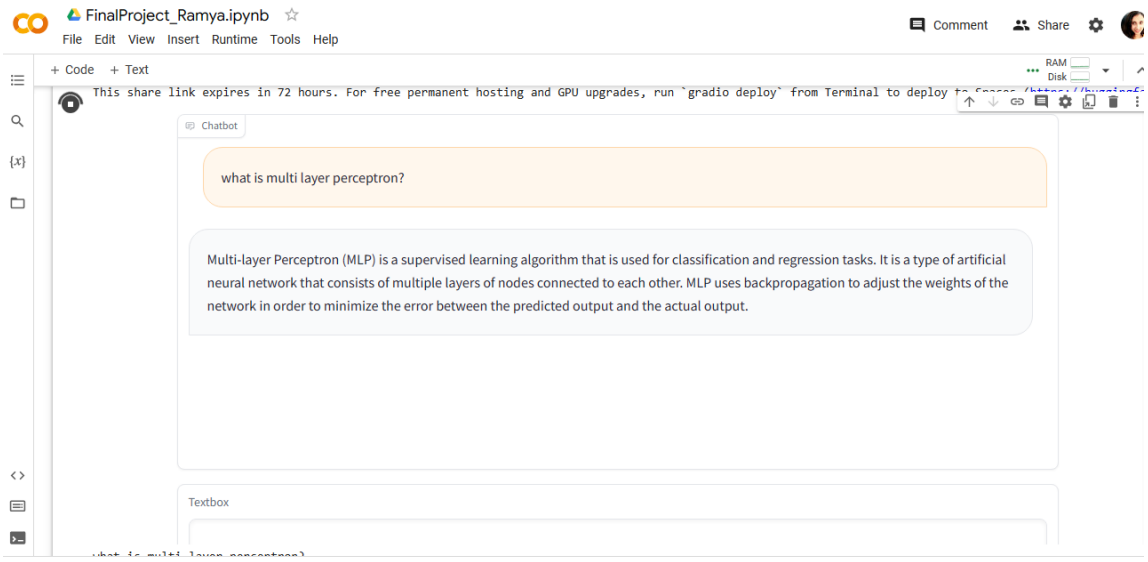

FinalProject_Ramya.ipynb
File Edit View Insert Runtime Tools Help Last edited on 1 October
Comment
Share
+ Code + Text
Connect
input_box = widgets.Text(placeholder='Please enter your question:')
input_box.on_submit(on_submit)
display(input_box)
Chat with your data. Type 'exit' to stop
Please enter your question:
User: What is Multi-layer Perceptron?
Chatbot: Multi-layer Perceptron is a classifier in which the weights of the network are found by solving a quadratic programming problem with linear constraints, rather than by solving a non-convex, unconstrained minimization problem as in standard neural network training.
[ ] pip install gradio
Collecting gradio
Downloading gradio-3.44.4-py3-none-any.whl (20.2 MB)
20.2/20.2 MB 25.6 MB/s eta 0:00:00
Collecting aiofiles<24.0,>=22.0 (from gradio)
Downloading aiofiles-23.2.1-py3-none-any.whl (15 kB)
Requirement already satisfied: altair<6.0,>=4.2.0 in /usr/local/lib/python3.10/dist-packages (from gradio) (4.2.2)
Requirement already satisfied: fastapi in /usr/local/lib/python3.10/dist-packages (from gradio) (0.99.1)
Collecting ffmpy (from gradio)
Downloading ffmpy-0.3.1.tar.gz (5.5 kB)
Preparing metadata (setup.py) ... done
Collecting gradio-client==0.5.1 (from gradio)
Downloading gradio_client-0.5.1-py3-none-any.whl (298 kB)

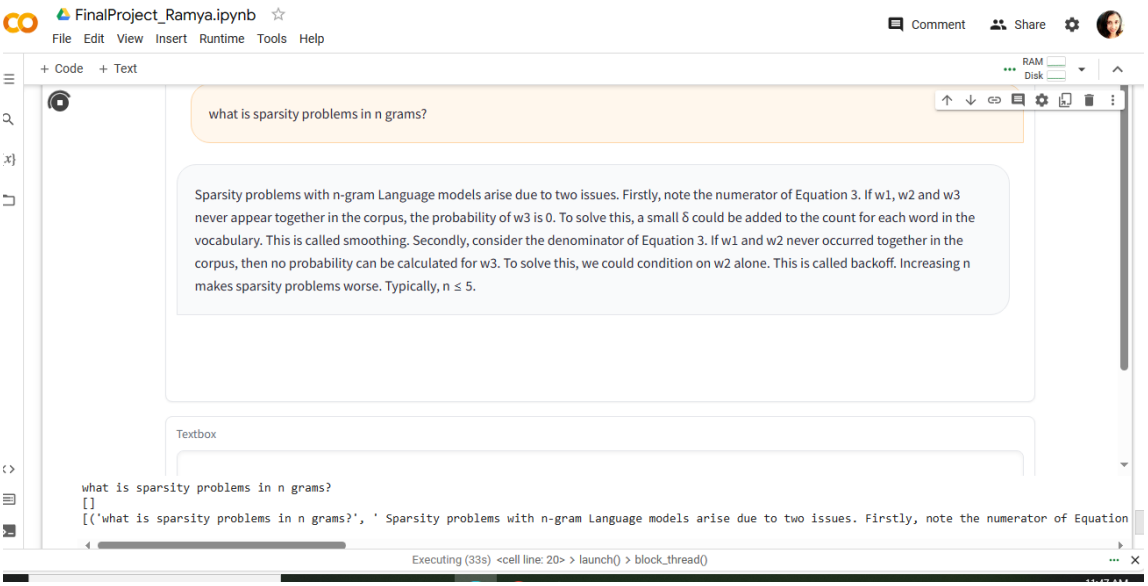

FinalProject_Ramya.ipynb
File Edit View Insert Runtime Tools Help Last edited on 1 October
+ Code + Text
import gradio as gr
with gr.Blocks() as demo:
chatbot = gr.Chatbot()
msg = gr.Textbox()
clear = gr.Button("Clear")
def respond(user_message, chat_history):
print(user_message)
print(chat_history)
# Get response from QA chain
response = qa({"question": user_message, "chat_history": chat_history})
# Append user message and response to chat history
chat_history.append((user_message, response["answer"]))
print(chat_history)
return "", chat_history
msg.submit(respond, [msg, chatbot], [msg, chatbot], queue=False)
clear.click(lambda: None, None, chatbot, queue=False)
demo.launch(debug=True, share=True)
Colab notebook detected. This cell will run indefinitely so that you can see errors and logs. To turn off, set de
Running on public URL: https://1ca08456d41becaa26.gradio.live
This share link expires in 72 hours. For free permanent hosting and GPU upgrades, run `gradio deploy` from Termin



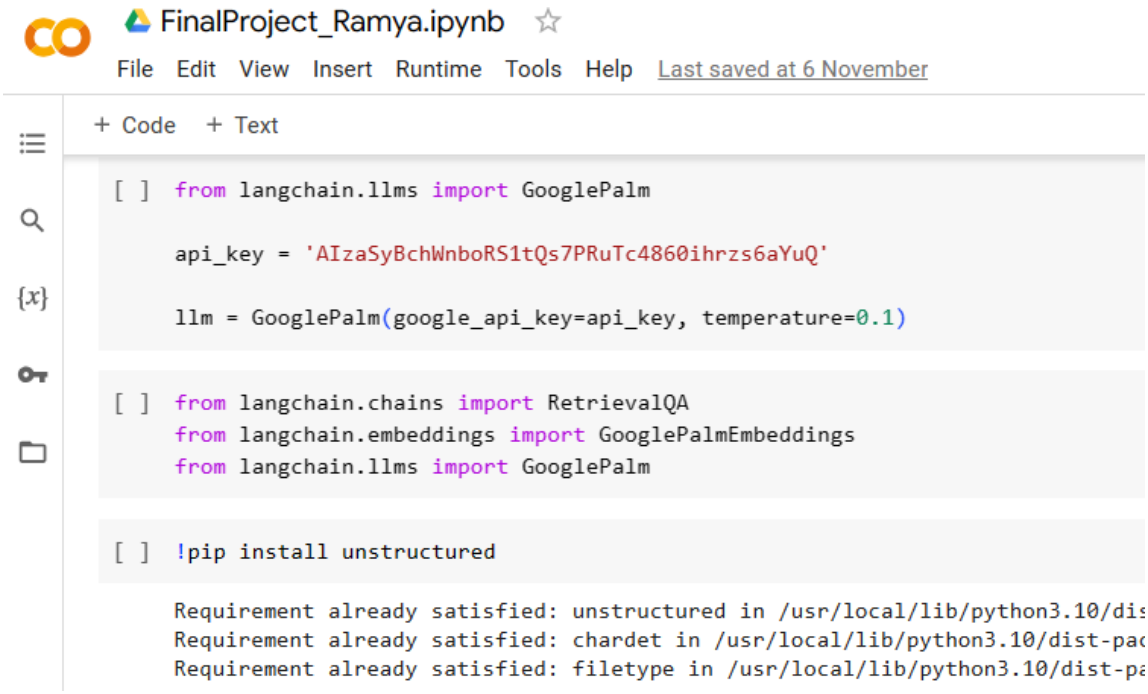

```
[51] import gradio as gr
     with gr.Blocks() as demo:
         chatbot = gr.Chatbot()
         msg = gr.Textbox()
         clear = gr.Button("Clear")

         def respond(user_message, chat_history):
             print(user_message)
             print(chat_history)
             # Get response from QA chain
             response = qa({"question": user_message, "chat_history": chat_history})
             # Append user message and response to chat history
             chat_history.append((user_message, response["answer"]))
             print(chat_history)
             return "", chat_history

         msg.submit(respond, [msg, chatbot], [msg, chatbot], queue=False)
         clear.click(lambda: None, None, chatbot, queue=False)

     demo.launch(debug=True, share=True)
```

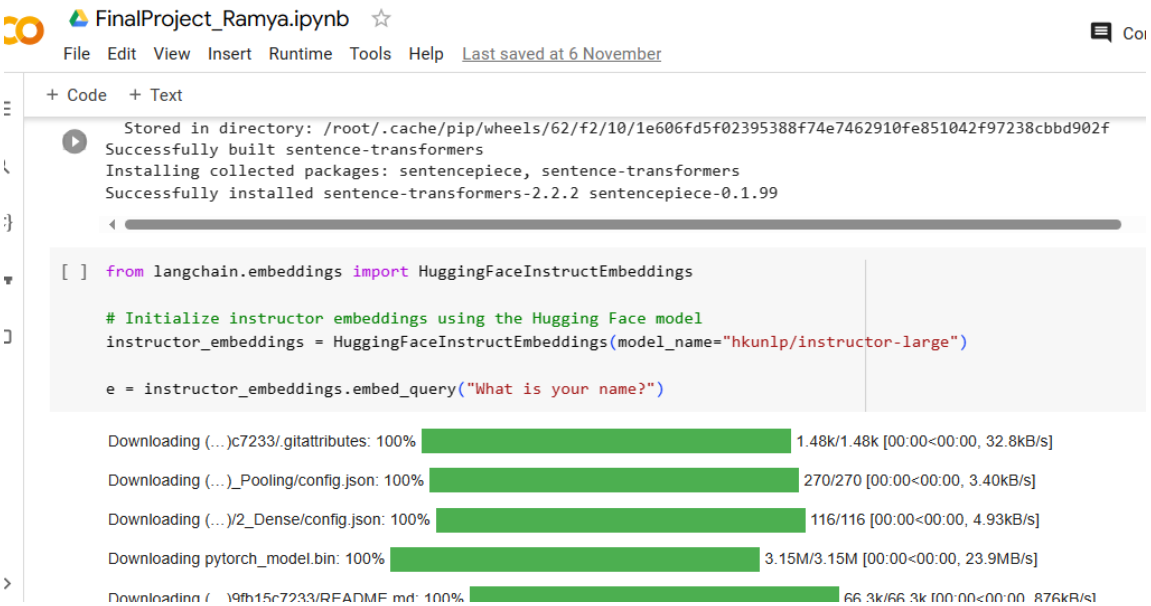
```
Colab notebook detected. This cell will run indefinitely so that you can see errors and logs. To turn off, set debug=False in launch().
Running on public URL: https://a9930d9d9cb15ac1fe.gradio.live

This share link expires in 72 hours. For free permanent hosting and GPU upgrades, run `gradio deploy` from Terminal to deploy to Spaces (https://huggingfa
```

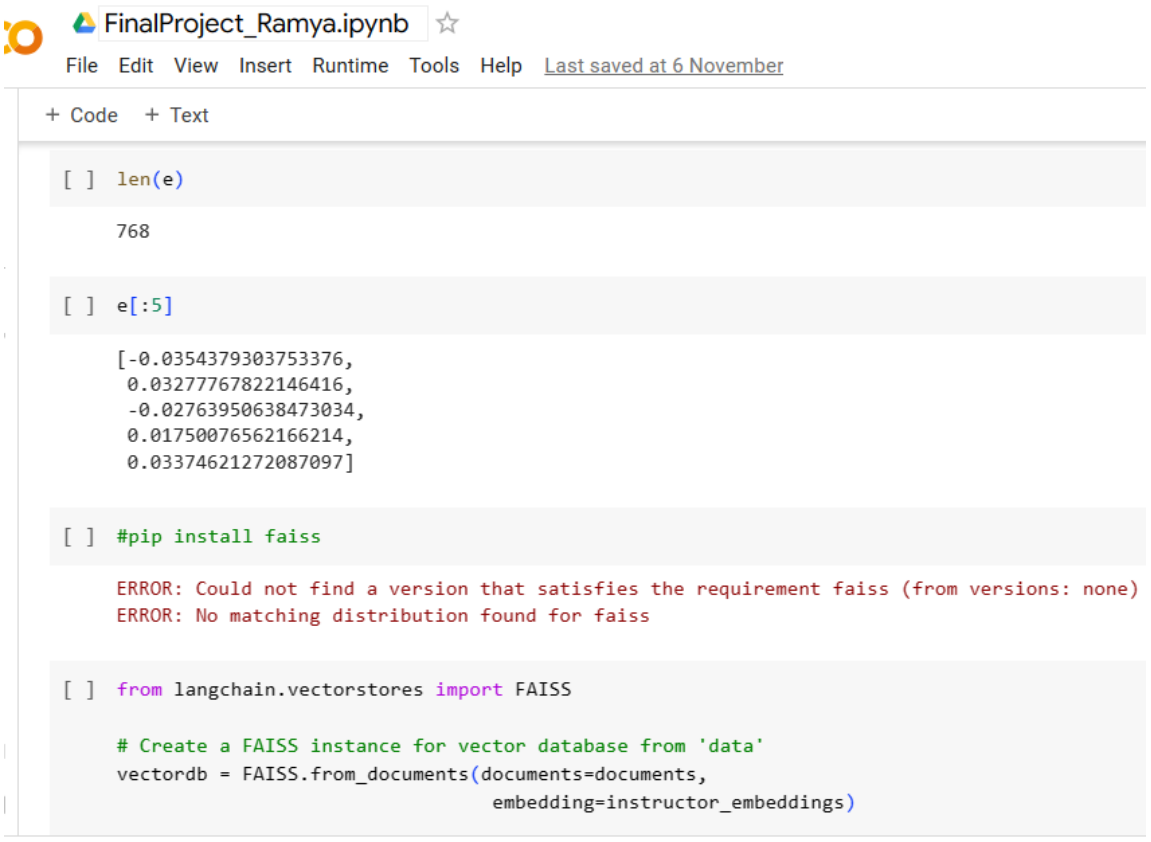




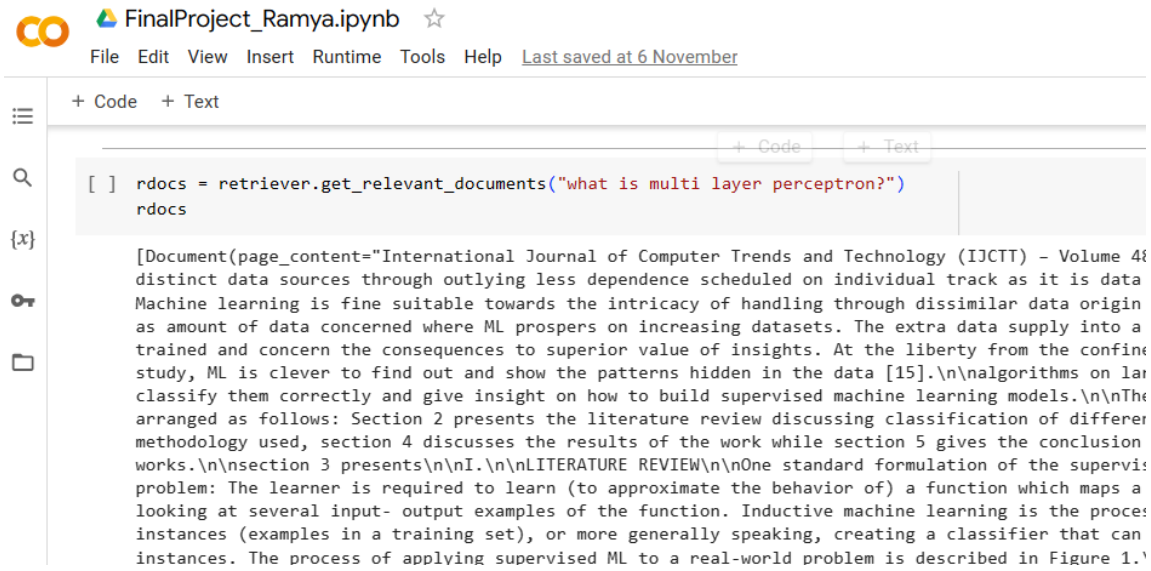

```
[7] pip install google-generativeai

Collecting google-generativeai
  Downloading google_generativeai-0.2.2-py3-none-any.whl (133 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 133.2/133.2 kB 2.7 MB/s eta 0:00:00
Collecting google-ai-generativelanguage==0.3.3 (from google-generativeai)
  Downloading google_ai_generativelanguage-0.3.3-py3-none-any.whl (267 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 267.9/267.9 kB 6.8 MB/s eta 0:00:00
Requirement already satisfied: google-auth in /usr/local/lib/python3.10/dist-packages (from google-generativeai) (2.17.3)
Requirement already satisfied: google-api-core in /usr/local/lib/python3.10/dist-packages (from google-generativeai) (2.11.1)
Requirement already satisfied: protobuf in /usr/local/lib/python3.10/dist-packages (from google-generativeai) (3.20.3)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from google-generativeai) (4.66.1)
Requirement already satisfied: proto-plus<2.0.0dev,>=1.22.0 in /usr/local/lib/python3.10/dist-packages (from google-ai-generativelanguage==0.3.3->google-g
Requirement already satisfied: googleapis-common-protos<2.0.dev0,>=1.56.2 in /usr/local/lib/python3.10/dist-packages (from google-api-core->google-generat
Requirement already satisfied: requests<3.0.0.dev0,>=2.18.0 in /usr/local/lib/python3.10/dist-packages (from google-api-core->google-generativeai) (2.31.0
Requirement already satisfied: cachetools<6.0,>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from google-auth->google-generativeai) (5.3.2)
Requirement already satisfied: pyasn1-modules>=0.2.1 in /usr/local/lib/python3.10/dist-packages (from google-auth->google-generativeai) (0.3.0)
```



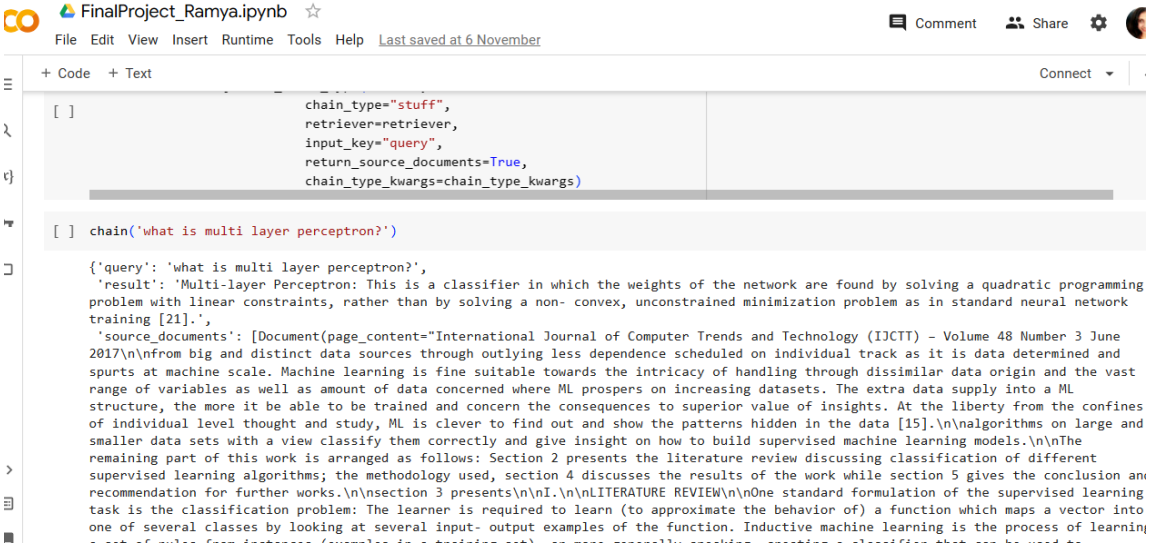

FinalProject_Ramya.ipynb
File Edit View Insert Runtime Tools Help Last saved at 6 November
+ Code + Text
Stored in directory: /root/.cache/pip/wheels/62/f2/10/1e606fd5f02395388f74e7462910fe851042f97238cbbd902f
Successfully built sentence-transformers
Installing collected packages: sentencepiece, sentence-transformers
Successfully installed sentence-transformers-2.2.2 sentencepiece-0.1.99
[ ] from langchain.embeddings import HuggingFaceInstructEmbeddings
# Initialize instructor embeddings using the Hugging Face model
instructor_embeddings = HuggingFaceInstructEmbeddings(model_name="hkunlp/instructor-large")
e = instructor_embeddings.embed_query("What is your name?")
Downloading (…)c7233/.gitattributes: 100% 1.48k/1.48k [00:00<00:00, 32.8kB/s]
Downloading (…)_Pooling/config.json: 100% 270/270 [00:00<00:00, 3.40kB/s]
Downloading (…)/2_Dense/config.json: 100% 116/116 [00:00<00:00, 4.93kB/s]
Downloading pytorch_model.bin: 100% 3.15M/3.15M [00:00<00:00, 23.9MB/s]

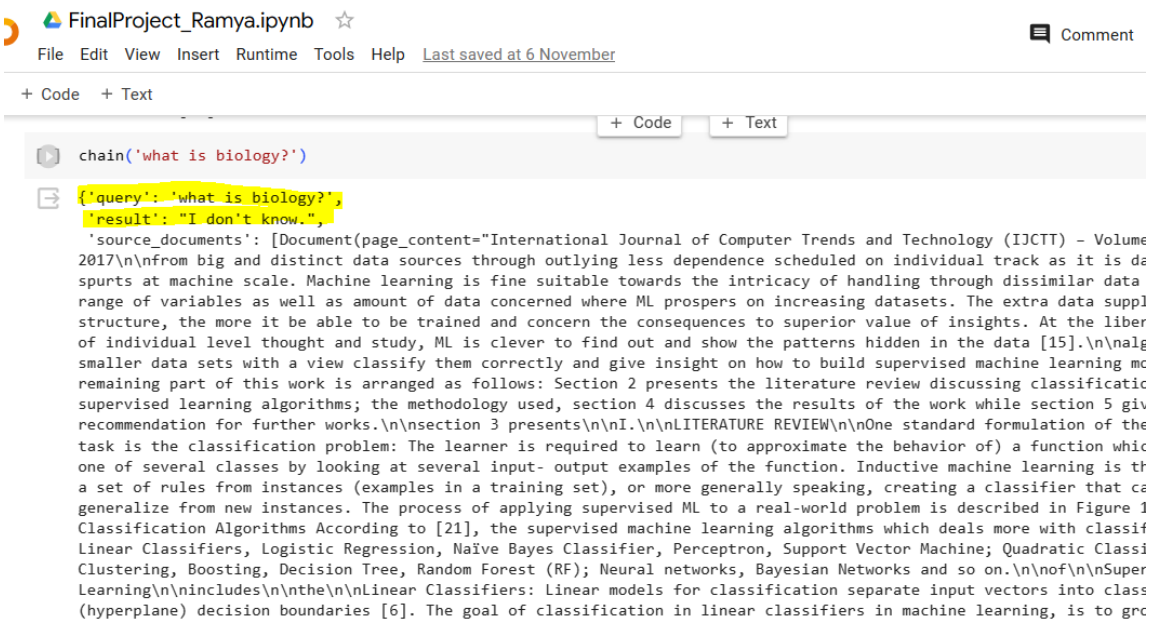

FinalProject_Ramya.ipynb
File Edit View Insert Runtime Tools Help Last saved at 6 November
+ Code + Text
[ ] len(e)
768
[ ] e[:5]
[-0.0354379303753376,
0.03277767822146416,
-0.02763950638473034,
0.01750076562166214,
0.03374621272087097]
[ ] #pip install faiss
ERROR: Could not find a version that satisfies the requirement faiss (from versions: none)
ERROR: No matching distribution found for faiss
[ ] from langchain.vectorstores import FAISS
# Create a FAISS instance for vector database from 'data'
vectordb = FAISS.from_documents(documents=documents,
embedding=instructor_embeddings)


FinalProject_Ramya.ipynb
File Edit View Insert Runtime Tools Help Last saved at 6 November
Comment
+ Code + Text
[ ] from langchain.vectorstores import FAISS
# Create a FAISS instance for vector database from 'data'
vectordb = FAISS.from_documents(documents=documents,
embedding=instructor_embeddings)
# Create a retriever for querying the vector database
retriever = vectordb.as_retriever(score_threshold = 0.7)
[ ] pip install faiss-gpu
Collecting faiss-gpu
Downloading faiss_gpu-1.7.2-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (85.5 MB)
85.5/85.5 MB 10.9 MB/s eta 0:00:00
Installing collected packages: faiss-gpu
Successfully installed faiss-gpu-1.7.2
https://python.langchain.com/docs/modules/data_connection/retrievers/ get_relevant_documents is actually extracting the meaning of th
questions given and not just giving the answer to the question.


FinalProject_Ramya.ipynb
File Edit View Insert Runtime Tools Help Last saved at 6 November
+ Code + Text
[ ] rdocs = retriever.get_relevant_documents("what is multi layer perceptron?")
rdocs
[Document(page_content="International Journal of Computer Trends and Technology (IJCTT) – Volume 48
distinct data sources through outlying less dependence scheduled on individual track as it is data
Machine learning is fine suitable towards the intricacy of handling through dissimilar data origin
as amount of data concerned where ML prospers on increasing datasets. The extra data supply into a
trained and concern the consequences to superior value of insights. At the liberty from the confine
study, ML is clever to find out and show the patterns hidden in the data [15].\n\nalgorithms on lar
classify them correctly and give insight on how to build supervised machine learning models.\n\nThe
arranged as follows: Section 2 presents the literature review discussing classification of differer
methodology used, section 4 discusses the results of the work while section 5 gives the conclusion
works.\n\nsection 3 presents\n\nI.\n\nLITERATURE REVIEW\n\nOne standard formulation of the supervis
problem: The learner is required to learn (to approximate the behavior of) a function which maps a
looking at several input- output examples of the function. Inductive machine learning is the proces
instances (examples in a training set), or more generally speaking, creating a classifier that can
instances. The process of applying supervised ML to a real-world problem is described in Figure 1.\

```
[ ] from langchain.prompts import PromptTemplate

    prompt_template = """Given the following context and a question, generate an answer based on this conte
    In the answer try to provide as much text as possible from "response" section in the source document co
    If the answer is not found in the context, kindly state "I don't know." Don't try to make up an answer.

    CONTEXT: {context}

    QUESTION: {question}"""


    PROMPT = PromptTemplate(
        template=prompt_template, input_variables=["context", "question"]
    )
    chain_type_kwargs = {"prompt": PROMPT}


    from langchain.chains import RetrievalQA

    chain = RetrievalQA.from_chain_type(llm=llm,
                                chain_type="stuff",
                                retriever=retriever,
                                input_key="query",
```




FinalProject_Ramya.ipynb
File Edit View Insert Runtime Tools Help Last saved at 6 November
Comment
+ Code + Text
+ Code + Text
chain('what is biology?')
{'query': 'what is biology?',
'result': "I don't know.",
'source_documents': [Document(page_content="International Journal of Computer Trends and Technology (IJCTT) – Volume
2017\n\nfrom big and distinct data sources through outlying less dependence scheduled on individual track as it is da
spurts at machine scale. Machine learning is fine suitable towards the intricacy of handling through dissimilar data
range of variables as well as amount of data concerned where ML prospers on increasing datasets. The extra data suppl
structure, the more it be able to be trained and concern the consequences to superior value of insights. At the liber
of individual level thought and study, ML is clever to find out and show the patterns hidden in the data [15].\n\nalg
smaller data sets with a view classify them correctly and give insight on how to build supervised machine learning mo
remaining part of this work is arranged as follows: Section 2 presents the literature review discussing classificatio
supervised learning algorithms; the methodology used, section 4 discusses the results of the work while section 5 giv
recommendation for further works.\n\nsection 3 presents\n\nI.\n\nLITERATURE REVIEW\n\nOne standard formulation of the
task is the classification problem: The learner is required to learn (to approximate the behavior of) a function whic
one of several classes by looking at several input- output examples of the function. Inductive machine learning is th
a set of rules from instances (examples in a training set), or more generally speaking, creating a classifier that ca
generalize from new instances. The process of applying supervised ML to a real-world problem is described in Figure 1
Classification Algorithms According to [21], the supervised machine learning algorithms which deals more with classif
Linear Classifiers, Logistic Regression, Naïve Bayes Classifier, Perceptron, Support Vector Machine; Quadratic Classi
Clustering, Boosting, Decision Tree, Random Forest (RF); Neural networks, Bayesian Networks and so on.\n\nof\n\nSuper
Learning\n\nincludes\n\nthe\n\nLinear Classifiers: Linear models for classification separate input vectors into class
(hyperplane) decision boundaries [6]. The goal of classification in linear classifiers in machine learning, is to gro