Final Report

Offline Chatbot for PDF Processing

Group 1: Cassandra Phung, Husandeep Kaur, Kumar Kartikeyn Arora

Course: COMP-3704 (254805) Neural Networks and Deep Learning

Date: December 5, 2024

Final Project Contributions

Cassandra Phung: 33.33%

Husandeep Kaur: 33.33%

Kumar Kartikeyn Arora: 33.33%

Introduction

Objective

This project developed a chatbot to operate locally and offline, ensuring user privacy and data security.

Key Features

- Processes and interacts with PDF files using a language model (LLM) within a Jupyter Notebook.
- By operating offline, the chatbot avoids relying on online platforms where user-uploaded data could be utilized for training generative AI models.
- Due to this being a local system, it is suitable for handling sensitive files such as financial, medical or general personal information.

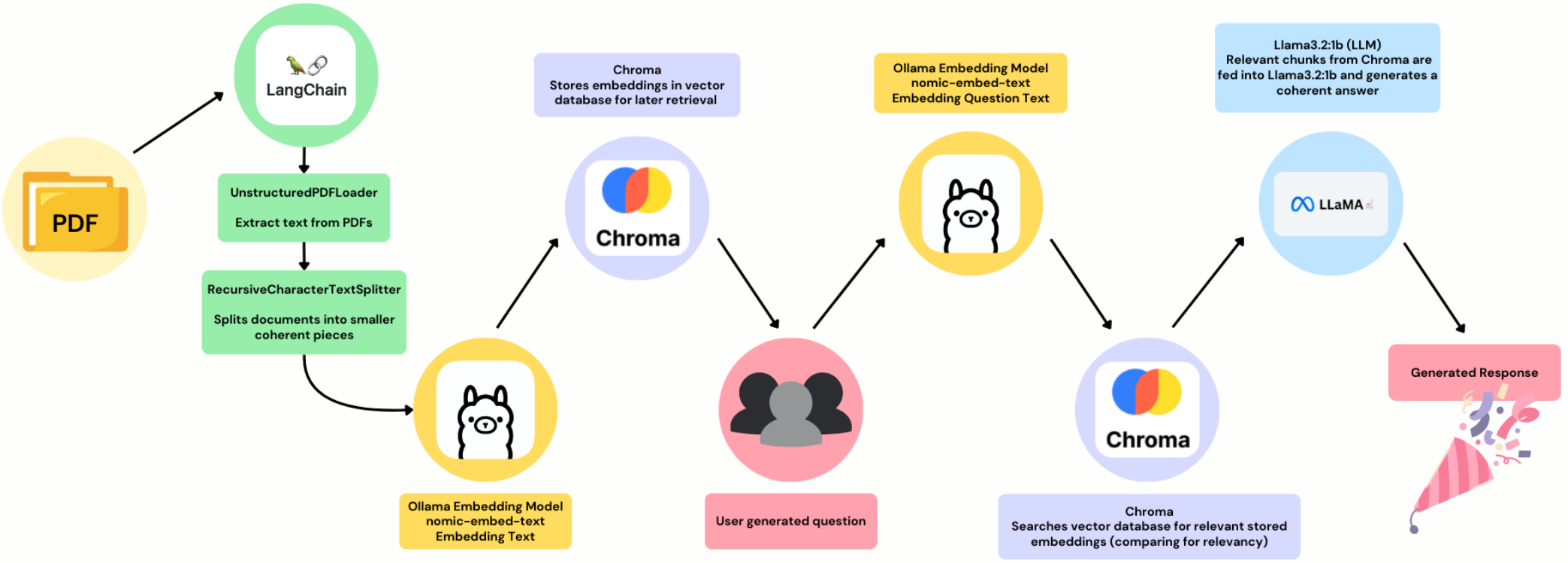
Use Case

Demonstration includes processing information from three Information Technology Programs at Red River College, sourced directly from the official RRC website.

Chatbot Project

Neural Networks & Deep Learning

Cassandra Phung
Kumar Kartikeyn Arora
Husandeep Kaur



Model Description & Overview

The local PDFs are loaded into a Jupyter Notebook, where the LangChain document loader (UnstructuredPDFLoader) is used to extract text from the documents, preparing them for further processing. The extracted text is then split and chunked using RecursiveCharacterTextSplitter, which divides lengthy documents into smaller, coherent sections, ensuring that chunks maintain readability.

The text chunks are embedded locally using the Ollama service with the nomic-embed-text embedding model. These embeddings are stored in a vector database created with Chroma, making them retrievable for answering user queries.

When a user enters a query, the system embeds the query and matches it against the stored embeddings in the vector database, identifying the most relevant chunks. The matched chunks are then processed by the large language model which generates a coherent and accurate response for the user.

Implementation

Tools and Frameworks

- **LangChain**: Framework for integrating LLMs with external sources such as PDFs.
- **Ollama**: Serviced used to download our embedding model and LLMs
- **Libraries Used**:
    - **UnstructuredPDFLoader:** Extracts text from PDF files.
    - **RecursiveCharacterTextSplitter:** Splits documents into manageable chunks for **LLM processing.**
    - **OllamaEmbeddings:** Generates embeddings for text processing.
    - **Chroma:** A vector database for efficient information retrieval.
    - **ChatOllama:** Language model integration for generating understandable responses.

Chatbot Step-by-Step

- **PDF Loading**: Text is extracted from PDFs using UnstructuredPDFLoader. And adds metadata to each document for organization and to improve the information's retrieval.

```
# Load documents and add metadata
documents = []

for local_path in pdf_paths:
    # Load PDF
    loader = UnstructuredPDFLoader(file_path=local_path)
    data = loader.load()
    print(f"PDF loaded successfully: {local_path}")

    # Add metadata to each document
    # Metadata identifying which program a document belongs to
    # Easier to filter results by program when retrieving or answering queries
    for doc in data:
        # Create metadata dictionary for each document
        doc.metadata = {
            "program": local_path.split(".")[0],  # Program identifier
            "program_name": get_program_name(local_path),
            "source": "Program PDF"
        }
        documents.append(doc)
```

```
PDF loaded successfully: Information_Security.pdf
PDF loaded successfully: Data_Science_Machine_Learning.pdf
PDF loaded successfully: Application_Development_Delivery.pdf
```

- **Text Splitting**: The text is divided into manageable chunks using RecursiveCharacterTextSplitter. With hyperparameter tuning conducted adjusting the chunk sizes and character overlap to improve context retention while keeping in mind the token limitations of LLM and the number of embeddings resulting from the chunk sizes.

```
# Changing Chunk size to a higher number, Now more content in a single chunk is taken.
# Overlap 200, to avoid losing key details
splitter = RecursiveCharacterTextSplitter(
    chunk_size=1000,
    chunk_overlap=200
)

chunks = splitter.split_documents(documents)
print(f"Total chunks created: {len(chunks)}")
```

```
Total chunks created: 43
```

- **Embeddings**: Chunks are converted into vectors by sending this text to our locally running Ollama service and using the model: nomic-embed-text for the embedding process.
- **Storage**: Embeddings are stored in a created vector database using Chroma, allowing quick retrieval of relevant chunks.

```
# Create vector database
# Embed and store chunks into vector database where it will be later retrieved
vector_storage = Chroma.from_documents(
    documents=doc_chunks,
    embedding=OllamaEmbeddings(model="nomic-embed-text"),
    collection_name="program-advisor-db"
)
print("Vector database created successfully.")
```

```
Vector database created successfully.
```

- **LLM:** Setup our LLM which is accessed locally through our Ollama service.
  LLM Selection: Tested both llama2:7b and the lighter llama3.2:1b models
  - Faster response generation due to reduced parameters (1 billion vs. 7 billion).
  - Trade-off: Slight reduction in performance and accuracy.
  - Selecting llama2:7b for the final model for better response accuracy

```
# Initialize the Llama 3.2.1b model
chat_model = ChatOllama(
    model="llama2:7b",
    base_url="http://localhost:12345"
)
```

- **Retrieval with Metadata Filtering:** Filtering mechanism to target relevant chunks based on program-specific metadata. This simplifies the retrieval process.
  Where the custom prompt template ensures that responses are accurate and context specific.

```
# Create MultiQueryRetriever
retriever = MultiQueryRetriever.from_llm(
    vector_storage.as_retriever(),
    chat_model,
    prompt=template
)
```

```
# Improved prompt template with placeholders for program
response_prompt = PromptTemplate(
    input_variables=["program_name", "context", "user_query"],
    template="""
    You are an academic advisor. Answer the user's query concisely based only on the provided context.

    Program: {program_name}
    Context: {context}
    Query: {user_query}

    Response:
    """
)
```

- **Generating Response**: Llama2:7b retrieves relevant filtered chunks from Chroma and generates human-readable answers.

```
def fetch_context_by_program(query, program_filter):
    # Prepare the inputs for the retriever
    prompt_inputs = {
        "program": program_filter,
        "query": query
    }

    # Generating relevant documents using MultiQueryRetriever
    results = retriever.get_relevant_documents(prompt_inputs)

    # Extracting and combine relevant context
    context = "\n".join([
        doc.page_content for doc in results if program_filter in doc.metadata.get("program", "")
    ])
    return context if context else "No specific information available."
```

```
from langchain.schema import SystemMessage, HumanMessage

def generate_response(query, program_filter):
    # Fetch context and program name
    context = fetch_context_by_program(query, program_filter)
    if not context.strip():
        return "No specific information is available in the provided documents for this query."

    program_name = program_name_mapper(program_filter)

    # Prepare structured messages
    messages = [
        SystemMessage(content="You are an academic advisor."),
        HumanMessage(content=f"Program: {program_name}\nQuery: {query}\nContext: {context}")
    ]

    try:
        # Generate response
        response = chat_model(messages, max_tokens=150)
        return response.content.strip()
    except Exception as e:
        return "An error occurred while generating the response. Please try again."
```

- **Chat with PDF:** Using an interactive chat loop created a better chatting experience compared to older versions where queries were entered directly in the cell.

```python
# Interactive chatbot loop for dynamic queries
def chatbot_interactive():
    print("Welcome to the Academic Advisor Chatbot!")
    print("Type 'exit' to quit the chatbot.")
    while True:
        try:
            user_query = input("Enter your question: ")
            if user_query.lower() in ["exit", "quit"]:
                print("Goodbye!")
                break
            program_filter = input("Specify the program (e.g., Data_Science_Machine_Learning): ").strip()
            if not program_filter:
                print("Please specify a valid program.")
                continue
            response = generate_response(user_query, program_filter)
            print(f"\nChatbot Response:\n{response}\n")
        except Exception as e:
            print(f"An error occurred: {e}")

# Run the chatbot
if __name__ == "__main__":
    chatbot_interactive()
```

Setup

- Created a virtual environment to isolate dependencies.
- Ensured libraries could function with the correct versions as dependencies were incredibly sensitive.
- Downloaded and installed the Ollama, changing default port from 11454 to 12345 as the default was causing errors with the endpoint for models on the local ollama service.
- Pulled models: nomic-embed-text, llama2:7b, and llama3.2:1b using ollama

Improvements

1. Large Language Model (LLM) was changed from llama2:7b to llama3.2:1b to speed up the chatbot responses (switching from 7 billion parameters to 1 billion) however this resulted in less accurate responses so we switched back to llama2:7b
2. The prompt template was improved and the chatbot was scripted to allow users to have a more interactive experience.
3. Using llama2:7b response time was slower but more accurate and relevant results.
4. The responses generated by the model earlier used a function (chat_with_pdf) which was changed to a interactive loop (chatbot_interactive) this saves time and asks user to specify question and program to generate outputs till the point user decides to quit.
5. By using MultiQueryRetriver we assured that all the PDFs are processed simultaneously, worked on creating custom template for getting more relevant results.
6. Improved readability of code and used a formatted prompt to show clear and concise output.

Testing

- Tested chatbot functionality with three IT program documents from the Red River College website.
- Testing user queries, hyperparameters and LLM versions to evaluate:
    - Accuracy of responses.
    - Speed of processing.

Accuracy, Performance & Testing Results

1. Adjusting chunk and chunk overlap sizes affected chatbot responses. Hyperparameters were tested to see for differences in generated responses & time.

```
# Split the documents into chunks
text_splitter = RecursiveCharacterTextSplitter(chunk_size=1000, chunk_overlap=100)
chunks = text_splitter.split_documents(documents)
print(f"Text split into {len(chunks)} chunks")
```

```
Text split into 50 chunks
```

```
# User Query #1
chat_with_pdf("There are 3 programs in the PDF documents, what are they?")
```

```
Response generated in 11.23 seconds.
```

Based on the provided PDF documents, there are three programs listed:

1. Application Development and Delivery (Program #1)
2. Information Security (Program #2)
3. Data Science and Machine Learning (Program #3)

```
# User Query #2
chat_with_pdf("For the Data Science Machine Learning Program, what elective courses are available in Year 2?")
```

```
Response generated in 6.12 seconds.
```

Based on the provided documents, the following elective courses are available for Year 2 of the Data Science Machine Learning program:

1. COMP-2036 Introduction to Bioinformatics
2. COMP-3702 Information and Data Architecture
3. COMP-3704 Neural Networks and Deep Learning
4. COMP-3706 Robotics and Automation
5. COOP-4001 Data Science and Machine Learning Co-op
6. PROJ-4001 Data Science and Machine Learning Industry Project

```
# User Query #3
chat_with_pdf("What Term 1 courses are available in the Application Development and Delivery program?")
```

```
Response generated in 6.11 seconds.
```

According to the provided Document (Program PDF), the following courses are available in Term 1 of the Application Development and Delivery program:

1. COMM-1173 - Communication Strategies
2. COMP-1327 - Software Development Fundamentals
3. COMP-1328 - Customer Experience and User Experience For Developers
4. COMP-1329 - IT Fundamentals
5. COMP-1332 - Service Management for Software Developers

These courses are typically taken in the first term of the program, which runs from September to December.

```
# User Query #4
chat_with_pdf("For the Data Science and Machine Learning Program: what are the costs?")
```

```
Response generated in 10.78 seconds.
```

The costs for the Data Science and Machine Learning program at Red River College can be found in the following documents:

Result

Using a **chunk size of 1000** and a **chunk overlap of 100** resulted in an average response time of **8.56 seconds.** While the results were mostly accurate, there were still minor issues:

- User Query #2: Although mostly correct, Co-op & Industry Projects were incorrectly listed as electives.

This configuration shows promise but highlights the need for further refinement to ensure complete accuracy.

```python
# Split the documents into chunks
text_splitter = RecursiveCharacterTextSplitter(chunk_size=1000, chunk_overlap=200)
chunks = text_splitter.split_documents(documents)
print(f"Text split into {len(chunks)} chunks")
```

```
Text split into 43 chunks
```

```python
# User Query #1
chat_with_pdf("There are 3 programs in the PDF documents, what are they?")
```

```
Response generated in 6.68 seconds.
```

Based on the information provided in the PDF documents, there are three programs offered by the Application Development and Delivery department at the Winnipeg location:

1. Application Development and Delivery
2. Data Science and Machine Learning
3. Information Security

```python
# User Query #2
chat_with_pdf("For the Data Science Machine Learning Program, what elective courses are available in Year 2?")
```

```
Response generated in 6.09 seconds.
```

Based on the provided program document, the following elective courses are available in Year 2 of the Data Science and Machine Learning program:

1. COMP-3702 Information and Data Architecture
2. COMP-3703 Introduction to Artificial Intelligence
3. COMP-3704 Neural Networks and Deep Learning
4. COMP-3706 Robotics and Automation

Note that these are the available elective courses in Year 2 of the program, and there may be additional or updated course offerings in subsequent years.

```python
# User Query #3
chat_with_pdf("What Term 1 courses are available in the Application Development and Delivery program?")
```

```
Response generated in 6.88 seconds.
```

Based on the document provided, the following courses are available in Term 1 of the Application Development and Delivery program:

1. COMM-1173 Communication Strategies 3 credit hours
2. COMP-1327 Software Development Fundamentals 6 credit hours
3. COMP-1328 Customer Experience and User Experience for Developers 6 credit hours
4. COMP-1329 IT Fundamentals 6 credit hours
5. COMP-1332 Service Management for Software Developers 3 credit hours

Note that the courses offered may vary depending on the academic term and the program delivery mode (part-time or full-time). It is best to consult the program's official website or contact the admissions office for the most up-to-date information.

```python
# User Query #4
chat_with_pdf("For the Data Science and Machine Learning Program: what are the costs?")
```

```
Response generated in 7.98 seconds.
```

Based on the provided documents, the costs for the Data Science and Machine Learning program are as follows:

Results

Using a **chunk size of 1000** and a **chunk overlap of 200** resulted in an average response time **of 6.91 seconds**, with significantly improved results.

- User Query #2: No longer incorrectly lists the industry project and co-op courses as electives.
- User Query #3: Now provides more detailed responses, including the credit hours for each listed course.

Given that an **overlap of 200 is producing better results**, further testing will focus on varying chunk sizes. Larger chunk sizes embed more comprehensive information per chunk, while smaller chunk sizes provide more focused and specific embeddings. With further testing, we'll determine the right balance between detail and precision in the chatbot responses.

```python
# Split the documents into chunks
text_splitter = RecursiveCharacterTextSplitter(chunk_size=800, chunk_overlap=200)
chunks = text_splitter.split_documents(documents)
print(f"Text split into {len(chunks)} chunks")
```

```
Text split into 57 chunks
```

```python
# User Query #1
chat_with_pdf("There are 3 programs in the PDF documents, what are they?")
```

```
Response generated in 7.91 seconds.
```

Based on the content of the PDF documents you provided, there are three programs listed:

1. Application Development and Delivery
2. Information Security
3. Data Science and Machine Learning

Each program has a detailed description of the program's learning outcomes, admission requirements, and costs.

```python
# User Query #2
chat_with_pdf("For the Data Science Machine Learning Program, what elective courses are available in Year 2?")
```

```
Response generated in 9.37 seconds.
```

Based on the provided program document, the following elective courses are available in Year 2 of the Data Science Machine Learning Program:

1. Information Security (ELRs option)
2. Networking Fundamentals (ELRs option)
3. Cloud Computing (ELRs option)
4. Cybersecurity Operations (ELRs option)
5. Data Science with Python (ELRs option)
6. Machine Learning with R (ELRs option)
7. Data Visualization (ELRs option)
8. Database Management Systems (ELRs option)
9. Software Development Methodologies (ELRs option)
10. Web Developmen (ELRs option)

Please note that the availability of these elective courses may vary depending on the program intake and student demand.

```python
# User Query #3
chat_with_pdf("What Term 1 courses are available in the Application Development and Delivery program?")
```

```
Response generated in 9.20 seconds.
```

According to the provided Program PDF, the following Term 1 courses are available in the Application Development and Delivery program:

1. Introduction to Computer Applications
2. Problem Solving and Algorithm Design
3. Data Structures and Algorithms
4. Software Development Life Cycle
5. Web Technologies

Results

Using a **chunk size of 800** and a **chunk overlap of 200** yielded the poorest results, with an average response time of **8.96 seconds** and completely inaccurate responses across multiple queries.

```
# Split the documents into chunks
text_splitter = RecursiveCharacterTextSplitter(chunk_size=1200, chunk_overlap=200)
chunks = text_splitter.split_documents(documents)
print(f"Text split into {len(chunks)} chunks")
```

```
Text split into 35 chunks
```

```
# User Query #1
chat_with_pdf("There are 3 programs in the PDF documents, what are they?")
```

```
Response generated in 6.79 seconds.
```

Based on the content provided in the PDF documents, the three programs mentioned are:

1. Application Development and Delivery Program
2. Information Security Program
3. Data Science and Machine Learning Program

```
# User Query #2
chat_with_pdf("For the Data Science Machine Learning Program, what elective courses are available in Year 2?")
```

```
Response generated in 5.79 seconds.
```

Based on the information provided in the Documents, the following elective courses are available in Year 2 of the Data Science Machine Learning program:

1. COMP-2036 Introduction to Bioinformatics
2. COMP-3702 Information and Data Architecture

Please note that course availability may vary depending on the academic term and the university's curriculum changes. It is recommended to consult with the university or program advisor for the most up-to-date information.

```
# User Query #3
chat_with_pdf("What Term 1 courses are available in the Application Development and Delivery program?")
```

```
Response generated in 8.89 seconds.
```

According to the provided Program PDF document, the following Term 1 courses are available in the Application Development and Delivery program:

1. COMP 101 - Introduction to Computer Science
2. COMP 102 - Problem Solving and Algorithms
3. COMP 105 - Web Technologies and Applications
4. COMP 106 - Intro to Databases and Data Modelling
5. COMP 107 - Software Development Life Cycle

These courses provide a foundation in computer science, problem-solving, web technologies, databases, and software development life cycle, which are essential for the Application Development and Delivery program.

```
# User Query #4
chat_with_pdf("For the Data Science and Machine Learning Program: what are the costs?")
```

```
Response generated in 6.86 seconds.
```

According to the document provided, the costs for the Data Science and Machine Learning program at the University of Winnipeg are as follows:

- Year 1: $15,413.00 * Year2$:9,615.00

Results

Testing with a **chunk size of 1200** and a **chunk overlap of 200** resulted in an average response time of **7.08 seconds**. However, this configuration led to issues such as missing two electives in the response to User Query #2 and providing a completely incorrect answer for User Query #3.

Further testing identified the **optimal configuration as a chunk size of 1000 with a chunk overlap of 200**, which delivered the best balance of accuracy and performance.

2. Switching from LLM llama2:7b to LLM llama 3.2:1b resulted in faster chatbot performance.

```
# Split the documents into chunks
text_splitter = RecursiveCharacterTextSplitter(chunk_size=1000, chunk_overlap=200)
chunks = text_splitter.split_documents(documents)
print(f"Text split into {len(chunks)} chunks")
```

```
Text split into 43 chunks
```

```
# Set up LLM and retrieval
local_model = "llama3.2:1b"
llm = ChatOllama(model=local_model, base_url="http://localhost:12345")
```

```
# User Query #1
chat_with_pdf("There are 3 programs in the PDF documents, what are they?")
```

```
Response generated in 4.76 seconds.
```

The three programs mentioned in the PDF documents are:

1. Data Science and Machine Learning
2. Information Security
3. Application Development and Delivery

```
# User Query #2
chat_with_pdf("For the Data Science Machine Learning Program, what elective courses are available in Year 2?")
```

```
Response generated in 1.85 seconds.
```

According to the provided document, for the Data Science Machine Learning Program, the following electives are available in Year 2:

1. COMP-2036 Introduction to Bioinformatics
2. COMP-3702 Information and Data Architecture
3. COMP-3704 Neural Networks and Deep Learning
4. COMP-3706 Robotics and Automation

```
# User Query #3
chat_with_pdf("What Term 1 courses are available in the Application Development and Delivery program?")
```

```
Response generated in 1.91 seconds.
```

According to the document, the following Term 1 courses are available in the Application Development and Delivery program:

1. COMM-1173 Communication Strategies 3
2. COMP-1327 Software Development Fundamentals 6
3. COMP-1328 Customer Experience and User Experience For Developers 6
4. COMP-1329 IT Fundamentals 6
5. COMP-1332 Service Management for Software Developers 3

```
# User Query #4
chat_with_pdf("For the Data Science and Machine Learning Program: what are the costs?")
```

```
Response generated in 1.87 seconds.
```

According to the text, there is no specific mention of costs for the Data Science and Machine Learning Program. However, I can tell you about the estimated costs mentioned:

- International applicants need to pay an International Credentials Assessment Fee, which has to be paid before their transcripts will be reviewed.
- Post-secondary transcripts must have been issued within 6 months prior to the application date.

If you're looking for a more detailed breakdown of estimated costs, it's best to check with the program administrator or a trusted online source.

Swapping from the previous large language model, **Llama2:7b** with 7 billion parameters, to the lighter-weight model **Llama3.2:1b** with 1 billion parameters results in a significant decrease in response time.

- **Average response time with Llama2:7b** (chunk size: 1000, chunk overlap: 200): **6.91 seconds**
- **Average response time with Llama3.2:1b** (chunk size: 1000, chunk overlap: 200): **2.59 seconds**

However, this improvement in speed comes with a potential trade-off in accuracy. While the chatbot successfully handled most queries, it struggled with **User Query #4**, where it had difficulty retrieving tuition-related results. This highlights the balance between model efficiency and performance.

Research and Experimentation:

Selections & Adjustments:

- Switching to a more lightweight LLM: Llama 3.2.1b recommended by Husandeep to improve chatbot response time instead of the originally selected llama 2.7b recommended by Cassandra.
- Switched back to llama2:7b for more accurate responses as the improvements in speed did not outweigh the more accurate responses
- Leveraged MultiQueryRetriver to form multiple versions of user queries, to obtain a broader range of information from the PDFs.
- Generated multiple versions of PDFs: testing the use of a single pdf vs. separate pdfs of each program.
- UnstructuredPDFLoader was the most effective way to load these PDFs
- Chroma was selected to create a vector db to store embeddings as this was the most popular choice in chatbots upon initial research.
- Created a custom prompt template to answer questions asked by user and answered by chatbot.
- Used retriever to fetch relevant documents and created a function for context by program.
- Used libraries like systemmessage and humanmessage to generate responses which sound like coming from a human.
- Scripted program filter to filter programs according to query and relevancy.
- Scripted an interactive chatbot loop for users to interact with instead of entering questions in a cell for a better user experience.

Experimentation & Additional Research:

- Found that we are getting faster response with llama3.2 but it is showing irrelevant data or output, decided to switch back to llama2:7b which was giving slower responses with less details but more relevant and accurate information.
- Hyperparameter testing for best and most accurate responses: chunk_size=1000 and overlap=200
- Husandeep conducted additional research on Docker and it's use instead of creating a virtual environment and if this would be beneficial to the final chatbot, it was not implemented in the end.
- Kartik researched Metadata and the benefits of utilizing it to retrieve more accurate information from our input data.
- Cassandra conducted initial research of the chatbot to find the most well utilized libraries, tools and LLMs for chatbot projects.
- Environment setup took a significant portion of time in the early stages of the chatbot project, as this required multiple tools, libraries and dependencies. Each group member encountered several errors in this process and troubleshooted these. These errors include dependencies for langchain, working with the correct version of python for the libraries, working with ollama and it's default automatic connection to port 11454 causing errors with local server endpoint of models used through ollama.

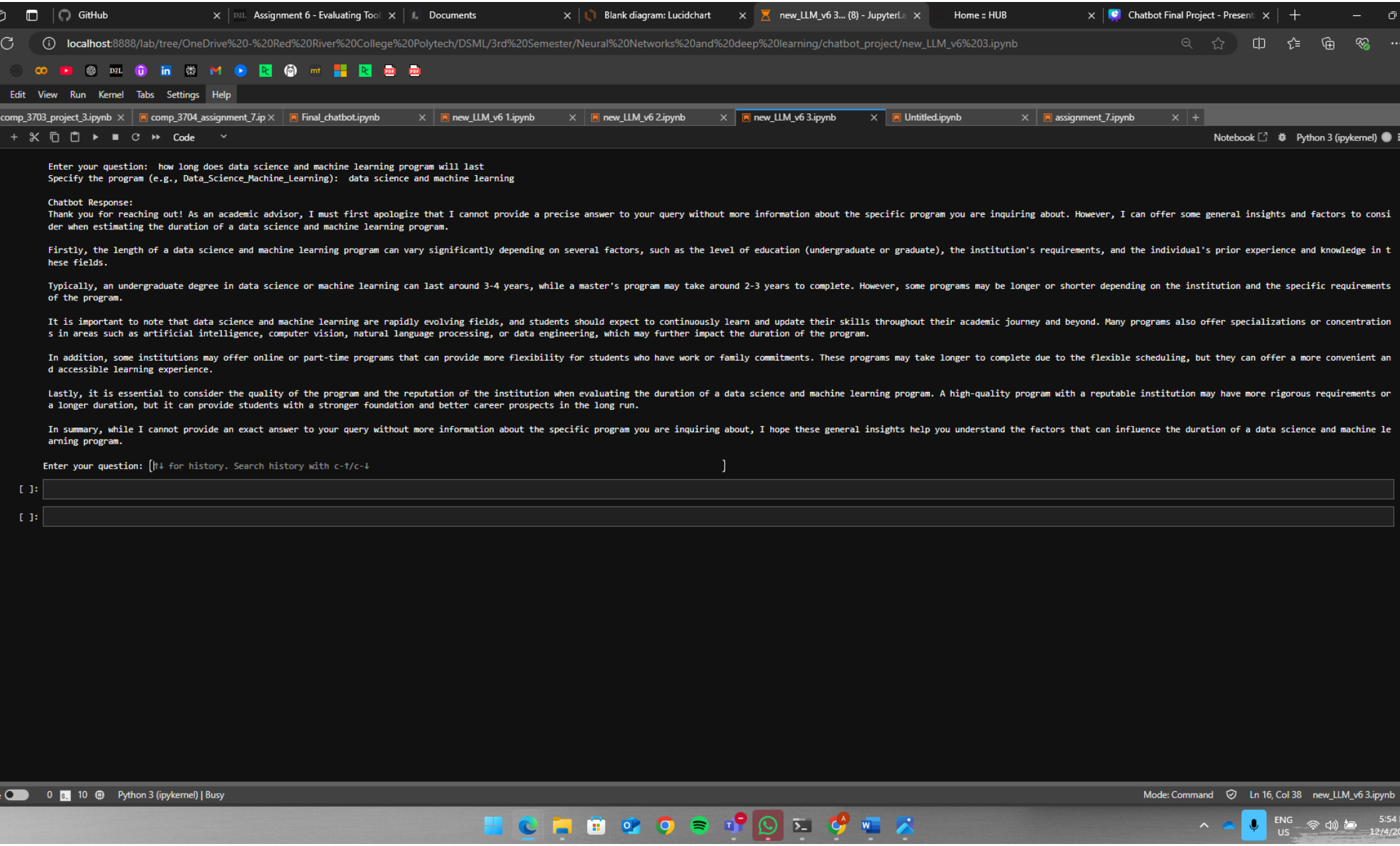
Final Code

User Query #1: Inquired about the key concepts of Data Science and Machine Learning Program and receiving an accurate human-like response.

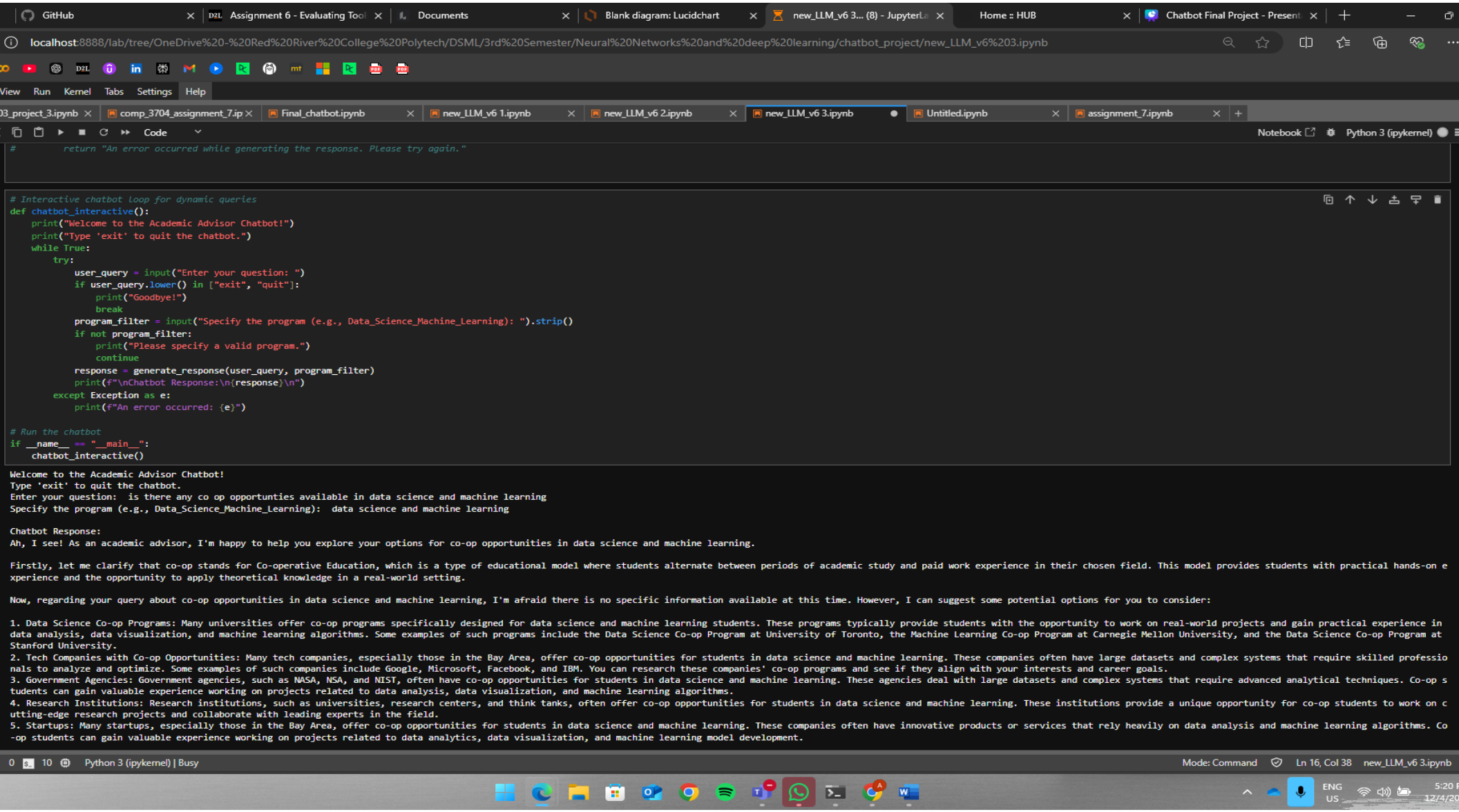
```
            continue
        response = generate_response(user_query, program_filter)
        print(f"\nChatbot Response:\n{response}\n")
    except Exception as e:
        print(f"An error occurred: {e}")

# Run the chatbot
if __name__ == "__main__":
    chatbot_interactive()
```

```
Welcome to the Academic Advisor Chatbot!
Type 'exit' to quit the chatbot.
Enter your question:  tell me about data science and machine learning program
Specify the program (e.g., Data_Science_Machine_Learning):  data science and machine learning

Chatbot Response:
As an academic advisor, I'm happy to provide you with general information about data science and machine learning programs!

Data science is an interdisciplinary field that combines elements of computer science, statistics, and domain-specific knowledge to extract insights and knowledge from data. The primary goal of data science is to uncover hidden patterns, relationships, and trends with
in large datasets, which can inform decision-making in various industries such as healthcare, finance, marketing, and more.

Machine learning is a subfield of artificial intelligence that involves training algorithms to make predictions or decisions based on data. It's a key component of data science, as it enables data scientists to build models that can learn from data and improve their p
erformance over time. Machine learning has applications in various areas, such as image recognition, natural language processing, recommendation systems, and predictive modeling.

Data science and machine learning programs typically cover a range of topics, including:

1. Programming languages: Python, R, or SQL are commonly used programming languages in data science.
2. Data structures and algorithms: Understanding data structures like arrays, lists, and dictionaries, as well as algorithms like sorting, searching, and graph traversal, is crucial for working with large datasets.
3. Statistics and probability: Statistical analysis and probability theory provide the foundation for understanding how to interpret and model data.
4. Machine learning: This includes supervised and unsupervised learning techniques, such as linear regression, logistic regression, decision trees, clustering, and neural networks.
5. Data visualization: Being able to effectively communicate insights and findings through visualizations is an essential skill in data science.
6. Database management: Understanding how to design, implement, and manage databases is important for storing and querying large datasets.
7. Big data technologies: Familiarity with big data technologies such as Hadoop, Spark, NoSQL databases, and cloud computing can be beneficial for working with large datasets.
8. Domain-specific knowledge: Depending on the industry or application area, students may need to learn domain-specific terminology and concepts to apply data science techniques effectively.

These are some of the key topics that are typically covered in data science and machine learning programs. However, it's important to note that program curricula can vary depending on the institution and the specific focus of the program. If you have any specific ques
tions or areas of interest within these fields, feel free to ask!
```

Query #2: Here we asked about the duration of a specific program like how long it takes to complete a data science and machine learning program. The chatbot provided an accurate response and also states additional information of whether the program is for a diploma or a degree.



Query #3: Here, we asked about the co-op opportunities available in a particular program and it gives us a variety of information, which could be very useful for the students.



Key Takeaways & Conclusions

Through comprehensive research, careful selection of libraries and language models (LLMs), hyperparameter testing, and effective implementation, we have successfully developed an efficient chatbot capable of processing user queries and retrieving relevant, contextually accurate information from three RRC Information Technology Programs materials. The result is a chatbot that generates human-like, coherent responses.

By utilizing tools such as LangChain, Chroma, and Ollama, the project achieved efficient document processing, robust information retrieval, and a user-friendly interaction experience. Key optimizations included hyperparameter tuning, improvements to the prompt template, and transitioning to an interactive chatbot loop, all of which contributed to enhanced response accuracy and user satisfaction.

Despite facing initial challenges with model selection and environment setup, the final implementation using the Llama2:7b model provided a reliable balance between accuracy and performance. Testing showed that a chunk size of 1000 and an overlap of 200 produced optimal results, emphasizing the importance of fine-tuning parameters for specific use cases. While the Llama3.2:1b model notably reduced response times, its accuracy trade-offs highlighted the value of prioritizing reliable outputs over speed.

Beyond the scope of this RRC program exploration, this offline chatbot for PDF processing has potential for working with sensitive information. Specifically, targeted towards individuals looking for local and offline chatbot options to accurately pull information from sensitive documents, without the worry of exposing their information online, ensuring data security and avoiding training future generative AI models with confidential data.

Future Work

Future experiments could explore the following areas:

- Optimizing the retrieval mechanism with advanced metadata filtering techniques.
- Investigating newer LLMs in hopes for faster response time without sacrificing accuracy in responses.
- Expanding the chatbot's capabilities to handle additional file formats and more complex queries.

These additions could improve the system's performance, usability, and versatility, making it a valuable tool for broader applications in secure data handling.