Semantic Spotter - RAG Project

Problem Statement

Polycystic Ovary Syndrome (PCOS) is a prevalent hormonal disorder affecting a significant number of women of reproductive age, leading to various health complications, including infertility, metabolic issues, and psychological effects. Despite its prevalence, many women lack access to clear, concise, and accurate information regarding PCOS, its symptoms, and management strategies. This project aims to develop a robust generative search system using Retrieval-Augmented Generation (RAG) techniques to effectively answer questions derived from policy documents and educational materials about PCOS. By leveraging LlamaIndex, the system will provide users with reliable, contextually relevant answers, enhancing their understanding and management of PCOS.

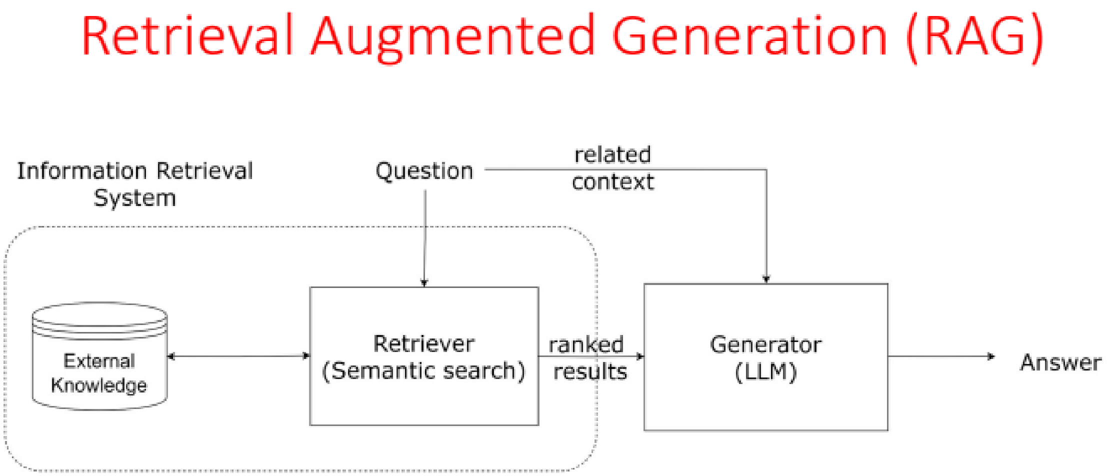
Project Summary

This project involves the creation of a generative search application designed to assist users in obtaining accurate information about PCOS. Utilizing LlamaIndex, the application will implement a semantic search bot that retrieves and processes information from two comprehensive PDF documents. The system will employ RAG techniques to enhance the quality of responses by integrating external knowledge with the generative capabilities of large language models (LLMs). This approach ensures that users receive up-to-date and authoritative information, thereby improving their ability to manage PCOS effectively.

Overview of Key Concepts

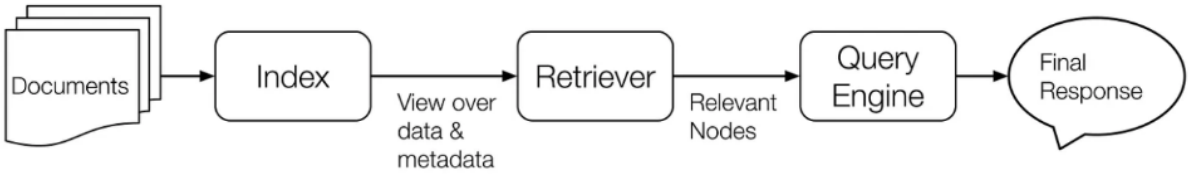
Retrieval-Augmented Generation (RAG)

RAG is a powerful framework that combines the strengths of information retrieval and generative models. It allows LLMs to access external knowledge bases, enhancing their responses with relevant, real-time information. This method addresses common challenges faced by LLMs, such as outdated knowledge and inaccuracies, by grounding their outputs in authoritative sources. RAG typically involves three main steps: retrieving relevant documents based on user queries, augmenting the query with this information, and generating a coherent response that integrates both the retrieved data and the model's internal knowledge.


Retrieval Augmented Generation (RAG)
Information Retrieval System
Question
related context
External Knowledge
Retriever (Semantic search)
ranked results
Generator (LLM)
Answer

LlamaIndex

LlamaIndex is an open-source framework designed for building context-augmented applications using LLMs. It simplifies the integration of various data sources, enabling developers to create applications that can efficiently process and respond to user queries. LlamaIndex supports data ingestion, indexing, and querying, making it an ideal choice for projects that require robust data handling capabilities. Its architecture allows for the creation of agents and workflows that can perform complex tasks, such as answering questions and synthesizing information from multiple documents.



Polycystic Ovary Syndrome (PCOS)

PCOS is a common endocrine disorder characterized by hormonal imbalances that can lead to irregular menstrual cycles, excess androgen levels, and the presence of cysts in the ovaries. Symptoms often include weight gain, acne, and hirsutism, and the condition can significantly impact fertility and overall health. Despite its prevalence, many women remain unaware of the condition's implications and management options. Early diagnosis and education are crucial for mitigating long-term health risks associated with PCOS, such as diabetes and cardiovascular disease.

This project not only aims to provide valuable information about PCOS but also seeks to empower women with the knowledge needed to navigate their health effectively.

Solution Strategy

The project focuses on building a proof of concept (POC) that meets the following requirements:

- Users receive responses from two medical PDFs about PCOS.
- The system provides citations to the original documents for reference.

Data Used

The system utilizes two medical documents sourced from authorized online resources.

Tools and Technologies

- **LlamaIndex**: Chosen for its efficient query engine and ease of implementation.
- **Python Libraries**: Includes pypdf, openai, and others for document processing and API integration.

Implementation Steps

1. **Library Installation**: Necessary libraries are installed, including LlamaIndex and OpenAI.
2. **Document Loading**: Medical documents are loaded from a specified directory.
3. **Query Engine Construction**: A vector store index is created to facilitate efficient querying.
4. **Response Generation**: A function is implemented to handle user queries and provide responses along with citations.

5. **Testing Pipeline**: A series of predefined questions are tested to evaluate the system's performance and gather user feedback.

Overview of LlamaIndex Usage

The LlamaIndex library is central to building a generative search system for answering questions about Polycystic Ovary Syndrome (PCOS). It facilitates efficient document processing, indexing, and querying, allowing users to retrieve relevant information from medical documents.

Key Components of the Code

Library Installation and Imports

```
•  get_ipython().system('pip install llama-index')
•  get_ipython().system('pip install pypdf')
•  get_ipython().system('pip install openai')
```

The necessary libraries are installed, including LlamaIndex for indexing and querying, pypdf for handling PDF documents, and OpenAI for language model integration.

Document Loading

```
•  from llama_index.core import SimpleDirectoryReader
•  reader = SimpleDirectoryReader(input_dir="/content/drive/My Drive/Colab Notebooks/PCOS_pdfs/")
•  documents = reader.load_data()
•  print(f"Loaded {len(documents)} documents/pages successfully.")
```

The SimpleDirectoryReader is used to load documents from a specified directory. This allows the system to read multiple PDF files containing information about PCOS.

Parsing Documents into Nodes

```
•  from llama_index.core.node_parser import SimpleNodeParser
•  parser = SimpleNodeParser.from_defaults()
•  nodes = parser.get_nodes_from_documents(documents)
```

The documents are parsed into nodes, which are smaller, manageable pieces of information. This step is crucial for building an index that can be queried efficiently.

Building the Index

```
•  from llama_index.core import VectorStoreIndex
•  index = VectorStoreIndex(nodes)
•  query_engine = index.as_query_engine()
```

A VectorStoreIndex is created from the parsed nodes. This index allows for fast similarity searches, enabling the system to find relevant information based on user queries.

Querying the Index

```
•  response = query_engine.query("What are the tests recommended to diagnose PCOS?")
```

The query engine is used to process user questions. The response includes relevant information extracted from the indexed documents.

Response Handling

```
•  def query_response(user_input):
•      response = query_engine.query(user_input)
•      file_name = response.source_nodes[0].node.metadata['file_name'] + " Page No " +
   response.source_nodes[0].node.metadata['page_label']
•      final_response = response.response + "\nCheck further at " + file_name
•      return final_response
```

This function formats the response by including citations to the original documents, enhancing the reliability of the information provided.

User Interaction Loop

```
•  def initialize_conv():
•      print("Feel free to ask questions related to PCOS. Enter exit once you are done!")
•      while True:
•          user_input = input("Enter your question: ")
•       if user_input.lower() == "exit":
•           print("Exiting the program. Bye!!!")
•           break
•       else:
•           response = query_response(user_input)
•           display(HTML(f'<p style="font-size:20px">{response}</p>'))
•           input("Press Enter to continue...")
```

This loop allows users to interact with the system, asking questions and receiving answers in real-time.

Future Enhancements

The code also includes recommendations for improving the system based on user feedback, such as refining the dataset and employing more effective data preprocessing techniques. Additionally, it suggests using customized nodes and language models to enhance response accuracy.

Conclusion

The LlamaIndex library plays a crucial role in this project by enabling efficient document handling, indexing, and querying. This setup not only provides users with accurate information about PCOS but also ensures that they can trace the information back to its source, enabling trust and reliability.