Semantic Spotter

Objectives

- **Develop a Generative Search System**: Create a system capable of effectively and accurately answering questions from various insurance policy documents.
- **Utilize LLaMA Index Framework**: Leverage Llama Index components to build a robust and scalable solution.
- **Enhance Data Accessibility**: Improve the accessibility and usability of insurance data for both customers and internal stakeholders.

Design

- **Data Sources**: Insurance datasets in PDF format.
- **Framework**: Llama Index, chosen for its modularity and ease of integration with large language models (LLMs).
- **Components**:
    - **Document Loaders**: CSVLoader for loading data.
    - **Embeddings**: OpenAIEmbeddings for semantic understanding.
    - **Query Engine**: Llama Index's query engine for processing queries.
    - **Load_data**: For loading the processed documents.

Implementation

1. **Data Preparation**:
    - Collect and clean insurance policy documents.
2. **Embedding Creation**:
    - Use OpenAIEmbeddings to create semantic embeddings of the documents.
3. **Chain Construction**:
    - Build an LLM Chain to handle user queries and generate responses.
4. **Storage**:
    - Store processed documents for quick retrieval.
5. **Integration**:
    - Integrate all components using Llama Index to form a cohesive system.

Challenges

The following challenges were faced while creating the project-

- **Data Quality**: Ensuring the accuracy and completeness of insurance documents.
- **Semantic Understanding**: Achieving high accuracy in understanding and processing complex insurance terms.
- **Performance**: Optimizing the system to handle large volumes of data and queries efficiently.
- **User Experience**: Designing an intuitive interface for users to interact with the system.

Lessons Learnt

The following lessons were learnt from the project.

- **Importance of Data Quality**: High-quality data is crucial for the accuracy of the system.
- **Modularity of Llama Index:** The modular nature of Llama Index significantly simplifies the development process.
- **Continuous Improvement**: Regular updates and fine-tuning of the model are necessary to maintain high performance.
- **User Feedback**: Incorporating user feedback helps in refining the system and improving user satisfaction.