PROJECT REPORT: GENERATIVE SEARCH SYSTEM FOR LIFE INSURANCE POLICY DOCUMENTS

1. INTRODUCTION

This project develops a Retrieval-Augmented Generative (RAG) search system to provide accurate answers to user queries about life insurance policy documents. The system consists of three core components:

- **System Architecture**
  - **Embedding Layer:** Breaks down the document into smaller chunks and converts them into numerical embeddings for efficient processing.
  - **Search Layer:** Utilizes a vector database to retrieve relevant chunks based on user queries and reranks them for optimal relevance.
  - **Generation Layer**: Leverages a language model to generate concise and accurate answers from the retrieved chunks.

2. SYSTEM DESIGN

The generative search system architecture comprises three primary components:

- **Embedding Layer**
  - **Document Processing:** Converts PDF documents to plain text using PyPDF2, followed by text cleaning to normalize whitespace and remove unnecessary formatting.
  - **Chunking Strategies:**
  - **Fixed-size chunks:** Splits text into predefined token lengths.
  - **Sentence-based chunking:** Breaks text into individual sentences.
  - Semantic chunking: Planned logic for semantically coherent text splitting.
  - **Embedding Generation:** Utilizes a pre-trained Sentence Transformer model (all-MiniLM-L6-v2) to create embeddings for each chunk.
- **Search Layer**
  - **Vector Database:** Indexes embeddings using ChromaDB or FAISS for efficient similarity searches.
  - **Query Embedding**: Converts user queries into compatible embeddings using the same model.
  - **Similarity Search:** Retrieves relevant chunks from the vector database based on query similarity.

- **Re-ranking:** Refines retrieved chunk relevance using a cross-encoder model (e.g., Sentence-BERT).

- **Generation Layer**
  - **Prompt Engineering:** Constructs prompts including the user query, relevant chunks, and guiding instructions for the language model.
  - **Language Model:** Leverages advanced models (e.g., OpenAI GPT) to generate final answers based on the engineered prompts.

3. IMPLEMENTATION DETAILS

- **Embedding Layer**
  - **Document Processing:** Utilizes PdfReader for text extraction, followed by cleaning and normalization.
  - **Text Chunking:** Implements fixed-size and sentence-based strategies, with potential for future semantic enhancements.
  - **Embedding Generation:** Employs a pre-trained SentenceTransformer model.
- **Search Layer**
  - **Vector Database:** Initializes ChromaDB or FAISS for efficient similarity searches.
  - **Similarity Search:** Retrieves relevant document chunks based on user queries.
  - **Re-ranking:** Enhances relevance using a cross-encoder model.

- **Generation Layer**
  - **Prompt Crafting:** Combines user queries, retrieved chunks, and instructions for the language model.
  - **Answer Generation:** Produces final answers based on crafted prompts.

4. CHALLENGES

- **Data Preprocessing:** Required extensive cleaning to ensure quality embeddings from PDF-extracted text.
- **Chunking Strategies:** Balancing chunk size and coherence was crucial for effective retrieval and generation.
- **Search Accuracy:** Fine-tuning similarity search and reranking processes was necessary for high precision.
- **Generation Quality:** Output quality depended on prompt design and model parameters.

5. RESULTS AND EVALUATION

Test queries were created to evaluate the system, such as:

- "What are the eligibility criteria for member life insurance?"
- "What benefits are provided under accidental death and dismemberment coverage?"
- "Define a 'Qualifying Event' as per the policy."

Performance evaluation criteria:

- **Relevance of Retrieved Chunks:** Assessed search layer effectiveness.
- **Coherence and Accuracy of Generated Answers:** Evaluated response clarity and correctness.

6. LESSONS LEARNED

- **Data Quality Matters:** Thorough preprocessing and structuring significantly enhance retrieval and generation effectiveness.
- **Chunking Strategy is Crucial:** A combined approach of sentence-based and semantic chunking yields optimal results.
- RAG Architecture is Powerful: The retrieval-generation combination proves highly effective for complex question-answering.
- **Scalability is Essential:** Efficient storage and retrieval mechanisms are vital for scaling the document corpus.

7. CONCLUSION

This project showcases the potential of generative search systems for querying policy documents. The layered architecture seamlessly integrates retrieval and generation, providing relevant, accurate, and concise answers. Despite preprocessing and chunking challenges, the RAG approach successfully addresses knowledge-intensive queries, demonstrating its practical value in the insurance domain.

-