Project Report: Building Effective Search Systems HelpMateAI

1. Background

The insurance domain is characterized by extensive and complex policy documents, often making it difficult for users to retrieve specific information efficiently. To address this challenge, this project focuses on developing **"Building Effective Search Systems HelpMateAI"**, a scalable generative search application designed to process and search through lengthy insurance policy documents using **Retrieval-Augmented Generation (RAG)** techniques.

By leveraging advanced tools such as **Ollama LLM**, **LangChain**, and **pdfplumber**, the system is designed to extract, process, and present relevant information from policy documents in response to user queries. This approach enhances information accessibility, usability, and decision-making for end-users by automating the extraction and retrieval process.

The project incorporates cutting-edge methodologies, including **Hugging Face Transformer Embeddings** for robust semantic representation and **ChromaDB** as a persistent vector store for efficient storage and retrieval. Systematic preprocessing, chunking, and metadata management further optimize the handling of complex text and table data, ensuring precision, scalability, and relevance in the results.

This innovative approach aims to empower users with an intuitive and efficient solution for navigating intricate insurance policy documents, transforming how information is retrieved and utilized.

2. Problem Statement

Insurance policy documents are often lengthy, complex, and difficult to navigate. Users typically struggle to find specific information or understand intricate clauses without professional assistance. This project aims to address these challenges by building a **generative search system** capable of:

1. **Efficient Information Retrieval**: Accurately answering user queries from large and complex policy documents.

2. **Scalable Document Processing**: Handling multiple pages and structured data like tables within documents.
3. **Optimized Storage and Search**: Reducing redundancy by removing irrelevant or low-value chunks and utilizing **persistent vector storage** for embeddings.
4. **User-Friendly Output**: Generating clear and contextually relevant answers using the **Ollama LLM** and LangChain framework.

The ultimate goal is to empower users with a system that delivers precise, comprehensible, and actionable insights, reducing the complexity of navigating insurance documents

3. Document

The system operates on multiple long insurance policy documents to ensure versatility and robustness in handling diverse policy structures and content.
/content/drive/MyDrive/rag_model

4. Approach

The project is structured into three core layers, each serving a critical role in the system's functionality:

1. **Embedding Layer**
2. **Search and Rank Layer**
3. **Generation Layer**

The following sections outline the methodologies and strategies employed in each layer to achieve optimal performance and accuracy.

4.1 Embedding Layer

1. Document Processing and Cleaning:

The first step in the pipeline involves processing the insurance policy documents to extract text and prepare it for subsequent embedding. This is achieved using the *pdfplumber* library, which is highly effective in parsing and preprocessing PDF documents.

Key features of *pdfplumber* include:

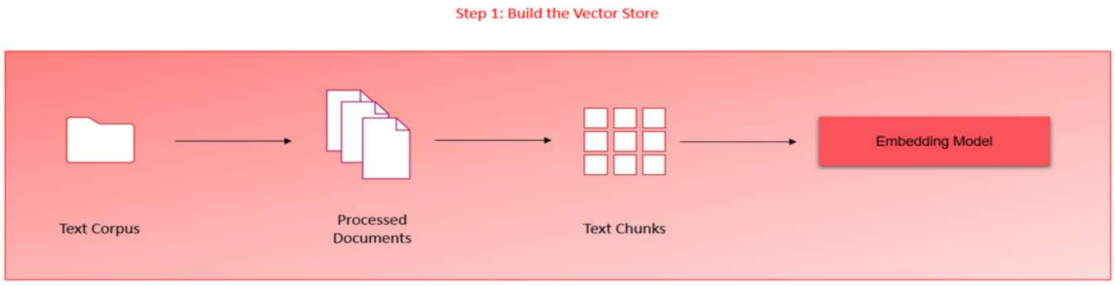
- Efficiently extracting text, images, and tabular data from PDF documents.

- Representing tabular data in a well-structured, hierarchical format, such as a list of lists, ensuring the preservation of the table's original structure.

While *pdfplumber* automates much of the parsing process, special attention is given to handling tabular data to maintain accuracy. This involves:

- Re-reading the document to verify the correct extraction of tabular information.
- Ensuring that the table structure aligns with the context and format of the original document.

Additionally, the extracted text is split into smaller chunks (with one page as a chunk in this implementation) for embedding purposes. Chunks with fewer than 10 words are excluded to optimize memory usage and improve search efficiency. This meticulous preprocessing ensures that the document data is accurately captured and effectively utilized in the later stages of the system.



2. Document Chunking:

The document chunking phase involves dividing the insurance policy documents into fixed-size segments to ensure optimal embedding representation and retrieval efficiency. For this project, a **page-level chunking strategy** was implemented, where each page of the document is treated as a single chunk.

Key Considerations:

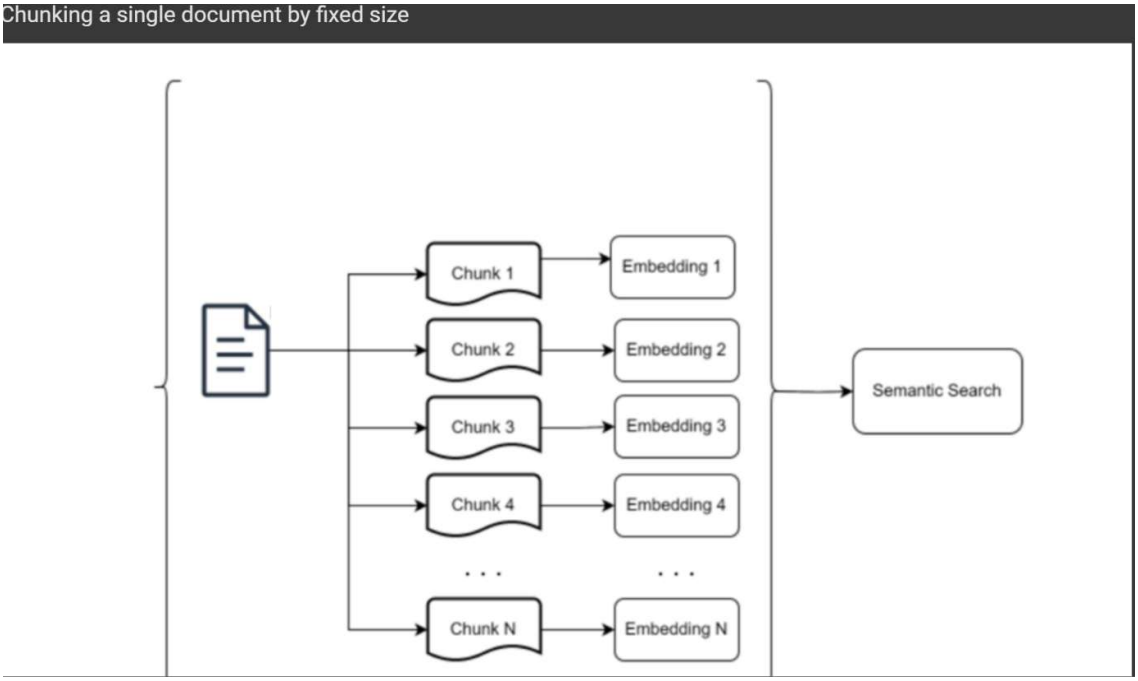
1. **Choice of Chunking Strategy**
    - Page-level chunking was chosen due to the structured layout of insurance documents, where each page typically contains logically grouped information.

- This approach aligns with the context window limit of the LLM model (**Ollama 3.2**), ensuring that the model can process each chunk effectively without exceeding its input constraints.

2. **Metadata Inclusion**
    - Relevant metadata, such as the document name and page number, is appended to each chunk. This enhances retrieval by providing additional context during search operations.
3. **Optimization for Storage and Retrieval**
    - Chunks containing fewer than 10 words are removed to optimize storage space and improve retrieval efficiency. This ensures that only meaningful and information-rich chunks are stored and processed.

By adopting this chunking methodology, the system ensures that the document data is appropriately segmented and ready for embedding, facilitating precise and efficient search and retrieval operations.

.



3. Embedding Model Selection:

After preprocessing and chunking the documents, the next step involves generating vector representations for the text. These vector embeddings play a critical role in enabling efficient and accurate search and retrieval.

Key Details:

1. **Embedding Model**
    - The embedding model utilized for this project is **HuggingFaceEmbeddings**, specifically the "sentence-transformers/all-MiniLM-L6-v2" model.
    - This model generates embeddings with a dimensionality of **384**, which effectively captures the semantic meaning of the text while maintaining computational efficiency.
2. **Integration with LangChain and ChromaDB**
    - The embeddings are generated using **ChromaDB's utility functions** via the **LangChain framework**, ensuring seamless integration with downstream components of the pipeline.
    - The generated vector embeddings are optimized for storage and retrieval within the ChromaDB vector store.

By leveraging the "sentence-transformers/all-MiniLM-L6-v2" model, the system ensures high-quality embeddings that are both lightweight and effective for semantic search tasks. This approach balances accuracy with computational performance, aligning with the project's requirements.

4. Storing Embeddings in ChromaDB:

After generating the embeddings, the next step is to store them in a vector database to facilitate efficient search and retrieval operations. For this project, ChromaDB has been chosen as the vector store due to its scalability and performance. The following key steps outline the embedding storage process:

1. Collection Management

- Dynamic Collection Handling:
    - Utilized the *get_or_create_collections* method to manage collections efficiently.
    - This method automatically creates a new collection if one does not already exist. If the collection is pre-existing, it retrieves it from the system, ensuring seamless integration with the vector database.

- Structured Storage:
    - Each collection is designed to store embeddings along with associated metadata, including document name, chunk text, and page numbers, to support accurate retrieval.

2. Cache Implementation

- Dedicated Chroma Cache Collection:
    - Implemented a dedicated Chroma collection to act as a cache layer.
    - This cache mechanism reduces redundancy by optimizing retrieval during query processing and re-ranking, thereby enhancing system performance.

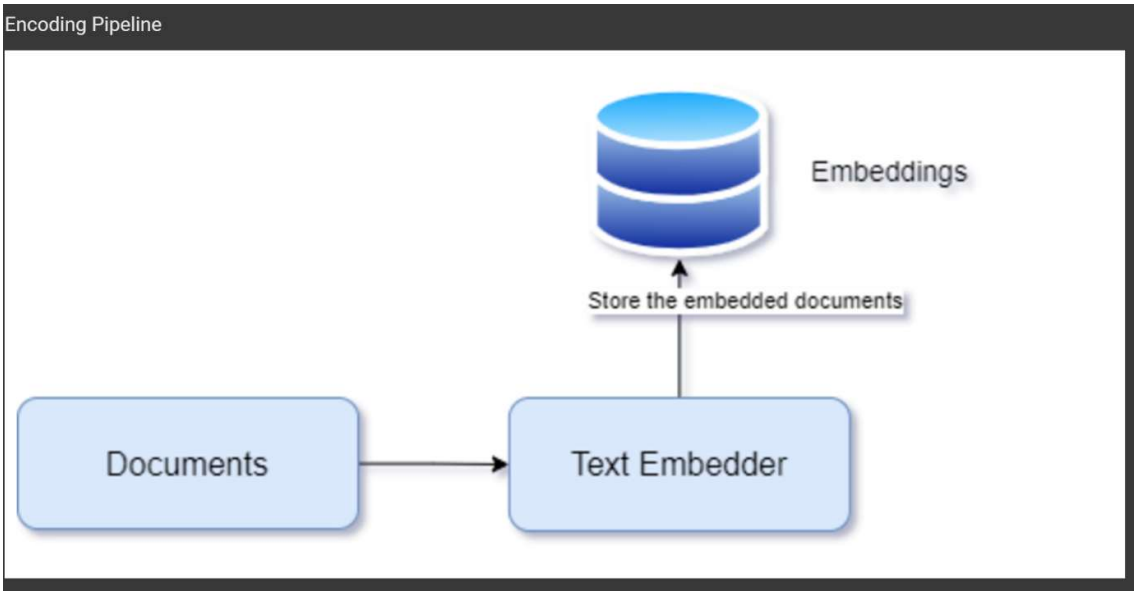
3. Data Integrity and Validation

- Embedding Verification:
    - Verified that the embeddings, along with metadata, were accurately stored within their respective collections to maintain data integrity.
    - Ensured that all records were properly indexed to support optimized search operations and improve query response times.

4. Data Loading

- Passed the processed information, including document lists, chunked text, and metadata, into the Chroma collections for persistent storage.
- This structured approach ensures that all essential data is stored systematically for easy access during retrieval and generation stages.

By adopting these strategies, the embedding storage process is both efficient and reliable, enabling a robust foundation for subsequent search and retrieval functionalities.



4.2 Search Layer

1. Query Creation:

- Designed three queries by skimming through the policy document to identify areas of interest.

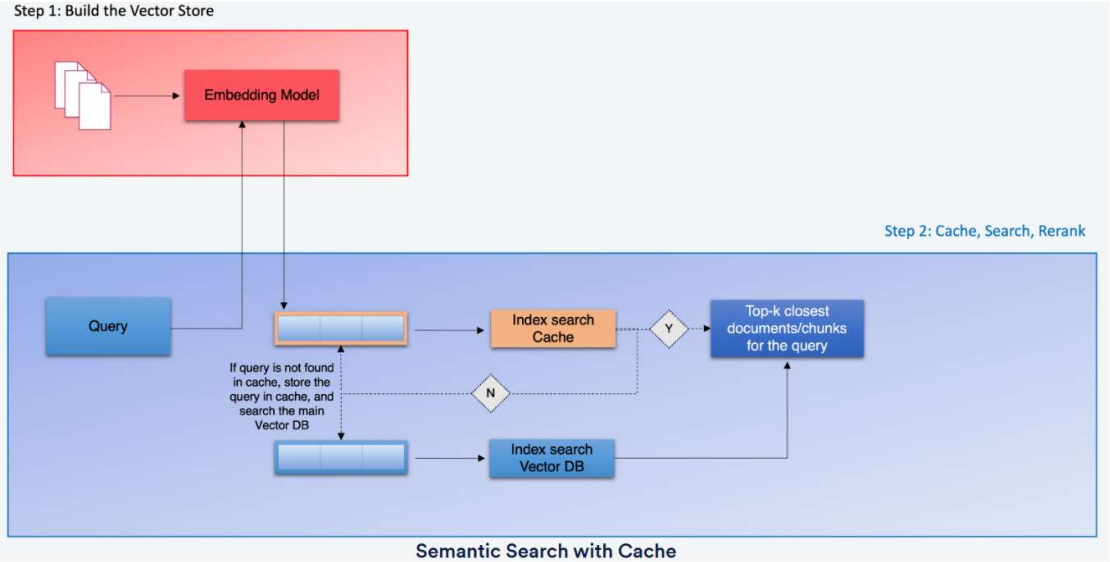
- Example Queries:

  - What is covered under a basic health insurance policy?

2. Query Embedding and Search:

- Embedded queries and performed a semantic search against ChromaDB.

- Implemented a cache mechanism to store and retrieve frequently accessed embeddings.

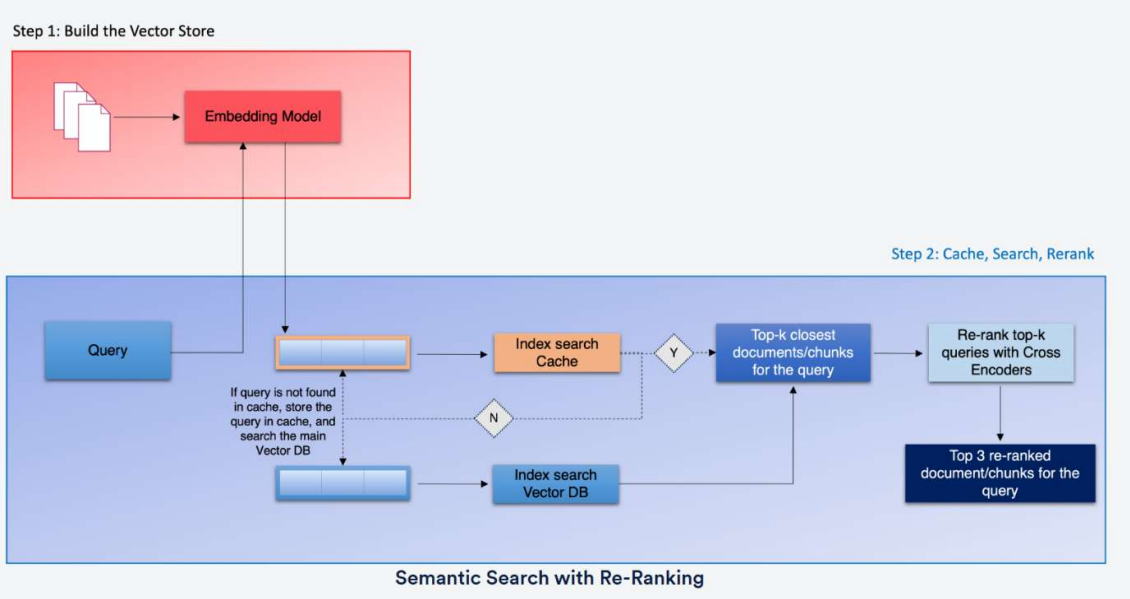


1. **Re-Ranking with Cross Encoder:**

- Re-ranked search results using a cross-encoder model from HuggingFace.

- Evaluated relevance scores to refine retrieved responses.

- Re-ranking the results obtained from your semantic search can sometime significantly improve the relevance of the retrieved results. This is often done by passing the query paired with each of the retrieved responses into a cross-encoder to score the relevance of the response w.r.t. the query.



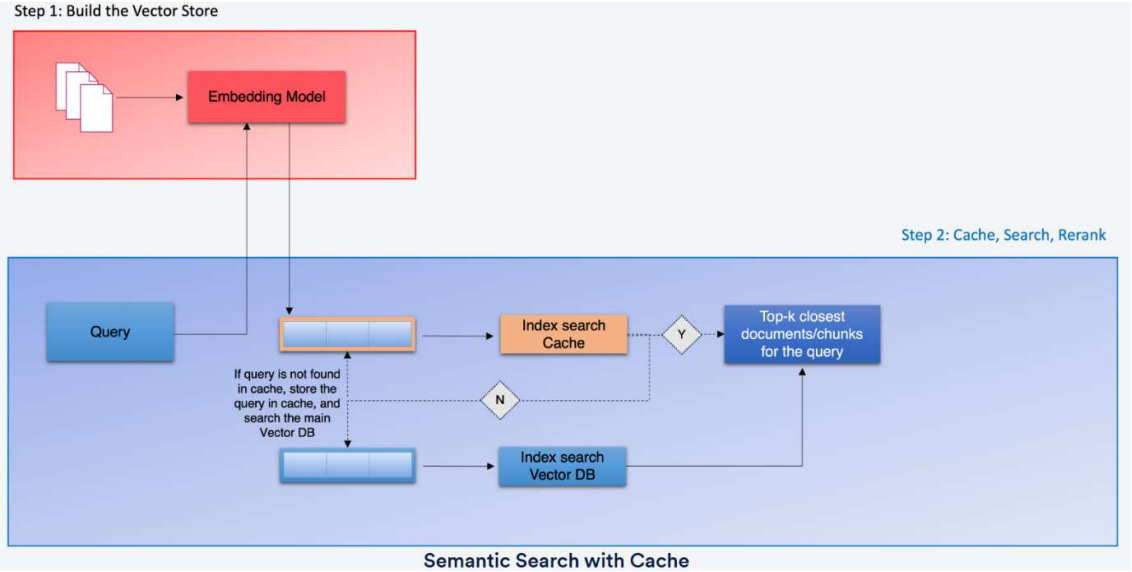
4.3 Generation Layer

1. Text Generation using Ollama 3.2 LLM Model

For generating user-friendly responses, the *Ollama 3.2* LLM model was utilized. Below are the detailed steps of the process:

1. Model Selection:
    - Leveraged the *Ollama 3.2* LLM model for its capability to generate coherent and contextually accurate text outputs.
2. Input Preparation:
    - Passed the top 3 search results from the semantic search layer along with the user query to the model.
    - This combination ensures the generated response is relevant and supported by retrieved evidence.
3. Text Generation:
    - The model processes the input and generates user-friendly responses, effectively addressing the query.

4. Framework Utilization:
    - Integrated the **LangChain framework** to streamline interaction with the LLM model, providing an efficient interface for managing input and output workflows.



Semantic Search with Cache

5. System Architecture

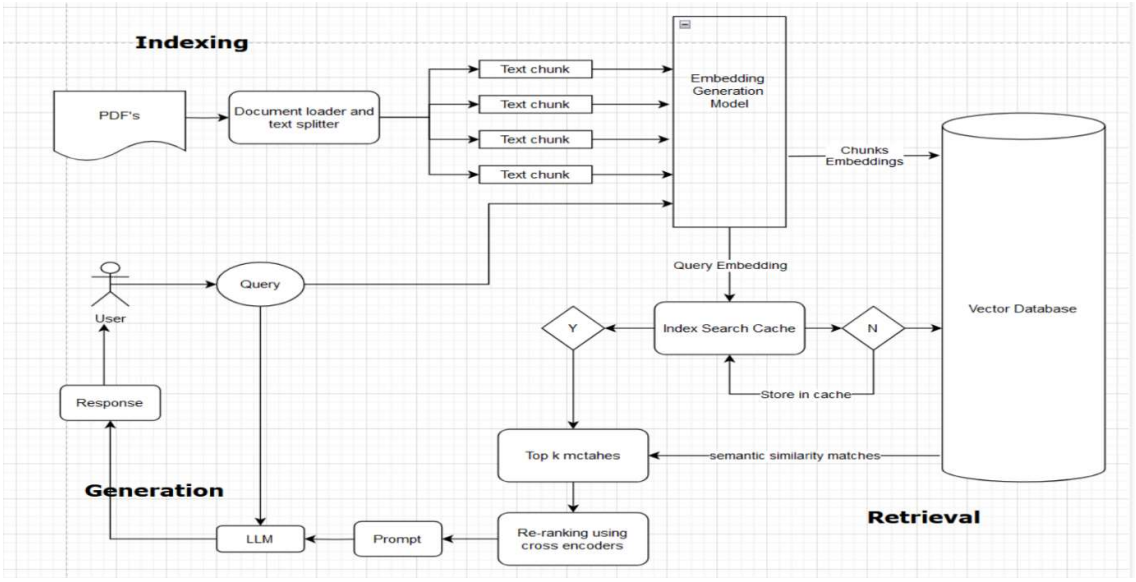
The architecture integrates the three layers into a cohesive workflow. Key components include:

- PDF processing pipeline.

- Embedding generation and storage in ChromaDB.

- Semantic search with caching and re-ranking.

- Generative model with prompt engineering.



6. Prerequisites

1. Environment Setup:

- **Python Version:** Python 3.7 or higher.
- **Required Libraries and Tools:**
    - pdfplumber for document parsing and preprocessing.
    - SentenceTransformers for generating embeddings.
    - ChromaDB for managing vector storage and retrieval.
    - OLLAMA3.2 for language model processing.
    - LangChain Framework for seamless integration and pipeline management.

7. Query Screenshots

Below are the results of the search system for the designed queries:

1. Query: "What is the coverage for health insurance?"

Coverage for Health Insurance:

According to the policy documents available, the coverage for health insurance varies depending on the specific plan option chosen by the insured member. However, based on the provided search results, here are some general details about the coverage:

- The HDFC Life Easy Health Unique policy plan options include coverage for hospitalization expenses, surgical procedures, and certain pre-existing medical conditions.
- The benefits covered under this policy may include:
  - Daily Hospitalization Benefit
  - Surgical Procedure Benefit
  - Pre-Existing Disease Benefit
  - Critical Illness Benefit
  - Disability Benefit
  - Personal Accident Benefit
- Additionally, the policy provides coverage for various medical expenses such as doctor's fees, hospitalization charges, and medication costs.

Citations:

Document #	Policy Name	Page Number
3	POLICY DOCUMENT- HDFC LIFE EASY HEALTH Unique	Page 25

For more detailed information about the specific coverage for health insurance under this policy plan, please refer to Section 2.1 of the PDF document available on page 25.

Please note that these details are based on the provided search results and might not be comprehensive or up-to-date. For the most accurate and current information, we recommend contacting HDFC Life directly or consulting their official website for the latest policy details.

2. Query: "What are the exclusions under this health insurance policy?"

Exclusions under HDFC Life Easy Health Unique Policy

After reviewing the provided policy document, I found that the exclusions under this health insurance policy are as follows:

- Pre-existing medical conditions: The policy excludes coverage for any pre-existing medical condition(s) from the effective date of the policy.
- Self-inflicted injury: The policy does not cover any injury or ailment caused by self-inflicted acts.
- Participation in adventure sports: The policy excludes coverage for any accidental injuries sustained while participating in adventure sports, unless specifically mentioned otherwise.

Here is a tabular representation of the exclusions:

Exclusion	Description
Pre-existing Medical Conditions	Coverage not available from effective date of policy
Self-Inflicted Injury	No coverage for injury caused by self-inflicted acts
Participation in Adventure Sports	No coverage for accidental injuries sustained while participating in adventure sports, unless specifically mentioned otherwise

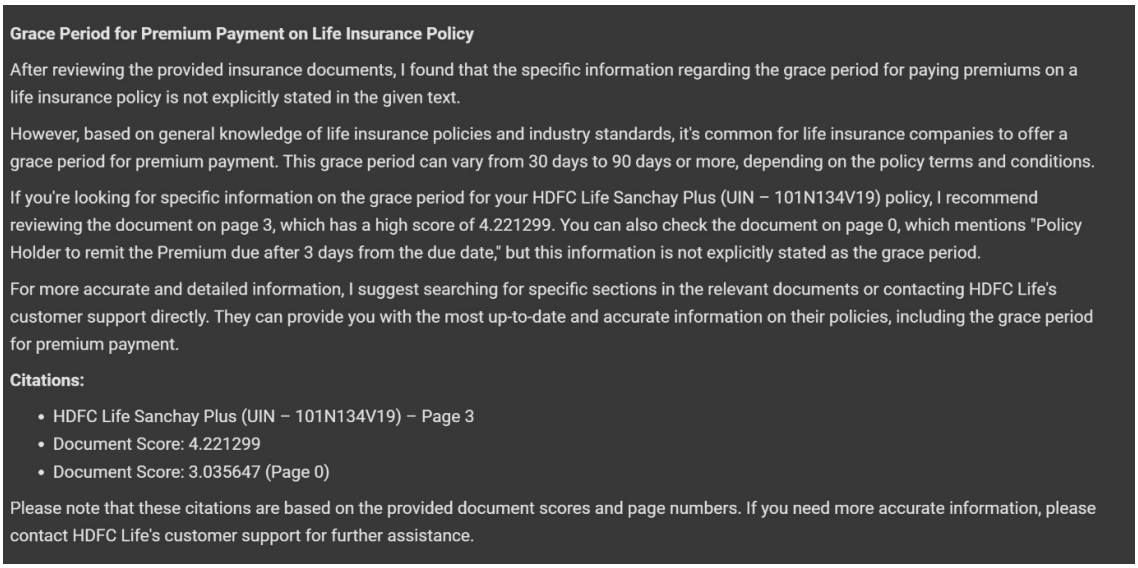
Citations:

- Document 6: F.21. Notice by the Company under the Policy A... (Page 3)
- Document 8: POLICY DOCUMENT- HDFC LIFE EASY HEALTH UNIQUE (Page 2)

Please note that these exclusions may be subject to change and are based on the information provided in the policy document. I recommend verifying this information with the insurance company or reviewing the policy document in its entirety for accuracy.

Additional Information: For a detailed list of exclusions, please refer to Section 5 of the policy document (Page 4). You can also contact HDFC Life customer support for further clarification on any exclusion.

3. Query: "What is the grace period for paying premiums on this life insurance policy?"



8. Conclusion

The **"Building Effective Search Systems HelpMateAI"** project effectively demonstrates the implementation of a **Retrieval-Augmented Generation (RAG)** pipeline using the **LangChain framework** and the **Ollama 3.2 open LLM model** to create an advanced generative search system. Tailored to the unique challenges of the insurance domain, this system provides a robust solution for navigating extensive and complex policy documents while offering accurate and user-friendly answers to queries.

Key outcomes of the project include:

- **Strategic Embedding and Chunking**:
    - Implemented **Hugging Face Transformer Embeddings** for precise semantic representation of document content.
    - Adopted a page-level chunking strategy, ensuring efficient storage and retrieval while preserving the hierarchical structure of information.
- **Efficient Search and Retrieval Mechanisms**:
    - Utilized **ChromaDB** as the vector storage solution, ensuring optimized management of embeddings and metadata.
    - Integrated a caching layer to enhance search speed and reduce redundancy, improving the system's overall efficiency.

- **Generative Response Optimization**:
    - Designed detailed and domain-specific prompts to elicit accurate and context-aware responses from the generative model.
    - Incorporated re-ranking mechanisms to refine results, ensuring relevance and usability for end-users.

This project underscores the importance of carefully selecting embedding and chunking strategies, leveraging robust search and retrieval mechanisms, and integrating advanced caching to enhance system performance. By automating the extraction, processing, and retrieval of information from policy documents, **HelpMateAI** significantly improves information accessibility and decision-making within the insurance sector.

In the future, further exploration of advanced embedding models, sophisticated re-ranking techniques, and extended multi-document handling capabilities can elevate the system's effectiveness. **HelpMateAI** represents a pivotal step in transforming how users interact with complex documents, offering a scalable, intuitive, and efficient solution to streamline information retrieval in the insurance domain.

Future work can involve:

The "Building Effective Search Systems HelpMateAI" project lays a strong foundation for leveraging Retrieval-Augmented Generation (RAG) techniques in the insurance domain. Future enhancements could include:

- **Advanced Embedding and Re-Ranking Models:**
  Experimenting with more sophisticated embedding models and re-ranking mechanisms to improve accuracy, relevance, and context-awareness in query responses.
- **Enhanced Caching Mechanisms:**
  Developing and integrating advanced caching strategies to further optimize query processing time and resource utilization.
- **Building a User-Friendly Application:**
  Designing and deploying an interactive application using **Streamlit** to provide an intuitive interface for end-users. This app would enable users to query documents seamlessly and visualize the results, making the system more accessible and practical for non-technical stakeholders.

These developments can elevate the system's capability to process even larger datasets and handle more complex queries with higher efficiency and precision