Project Report

📄 MR. HELPMATE AI – INSURANCE POLICY QNA SYSTEM

❓ OVERVIEW

Mr. HelpMate AI is an intelligent assistant designed to help policyholders query and understand complex insurance policy documents using natural language. Leveraging **Large Language Models (LLMs)**, **semantic search**, and **retrieval-augmented generation (RAG)**, the system provides users with precise, contextually grounded answers directly from their policy content.

In an industry riddled with dense legal terminology and inaccessible information, this project bridges the gap between end-users and their policy knowledge by offering a conversational layer over traditionally static documents.

🎯 PROJECT GOALS

The primary objective of this project is to build an intelligent, document-aware QA (Question-Answering) system that can answer user questions accurately and transparently. The key goals are:

1. **Document Understanding:** Develop a system that can semantically parse and understand policy documents regardless of formatting or length.
2. **Accurate Question-Answering:** Enable policyholders to ask questions in natural language and receive precise, contextually relevant, and non-hallucinatory answers.
3. **Retrieval-Augmented Generation (RAG):** Incorporate an effective semantic search mechanism that retrieves relevant content before generating responses.
4. **Scalability and Adaptability:** Build a modular and scalable architecture that can easily extend to other domains like legal contracts, financial agreements, or technical documentation.
5. **Explainability and Grounding:** Ensure that every generated answer can be traced back to the original document context, enhancing user trust and reducing misinformation.

💼 DATA SOURCES

The project is designed to work with a variety of **insurance policy documents**, which can be obtained from:

- **Customer-uploaded PDF documents** (e.g., health, motor, or travel insurance)
- **Sample policy documents** provided by insurers or publicly available online
- **Synthetic documents** generated for testing edge cases and different formats

To process these documents effectively, the system uses:

- `pdfplumber` for PDF extraction
- A custom chunking algorithm that splits the text into semantically meaningful segments
- Metadata tagging (e.g., section headers, clause numbers) for enriched retrieval

---

🛠 SYSTEM DESIGN & ARCHITECTURE

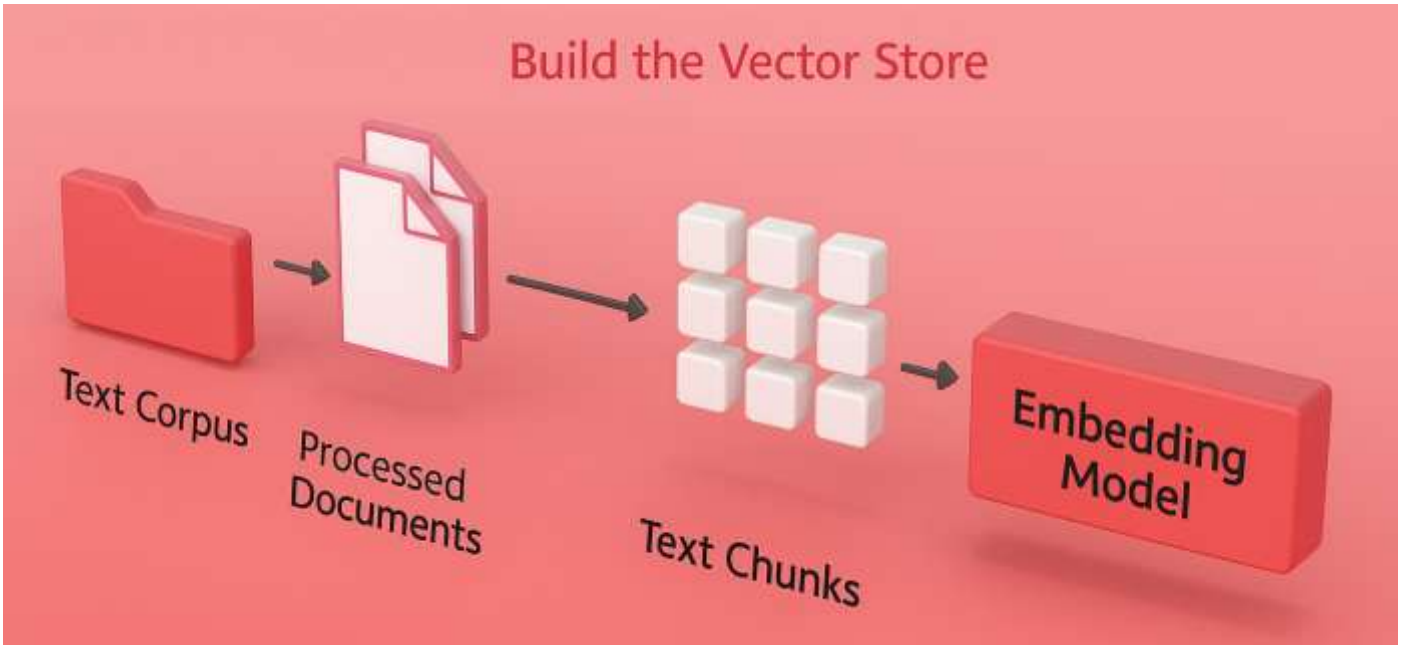
The architecture of Mr. HelpMate AI is based on a **three-layered pipeline** that ensures semantic understanding, contextual retrieval, and generative answering:

1. EMBEDDING LAYER

- Converts documents and user queries into **dense vector embeddings** using models such as:
  - `text-embedding-3-small` (OpenAI)
  - `all-mpnet-base-v2` (Sentence Transformers)
- Embeddings capture the **semantic meaning**, not just keywords, enabling similarity search based on meaning.

Design Choices:

- Chose Sentence Transformers for local use and OpenAI for cloud-based precision.
- Employed a configurable chunking strategy (by tokens, sentences, or sections) to preserve semantic integrity.



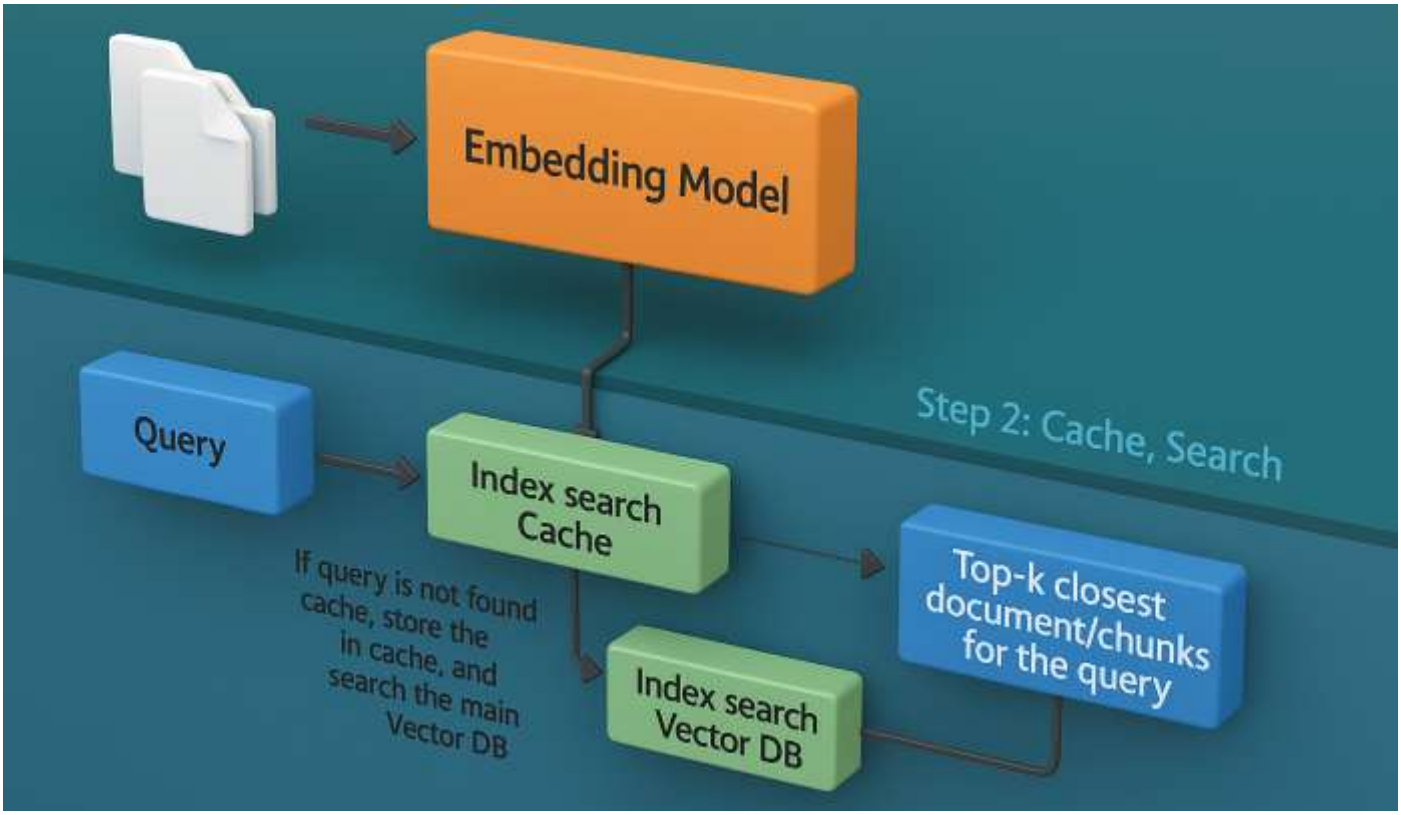
2. SEARCH LAYER

- Uses a vector database (like **FAISS** or **Chroma**) to store and query embeddings.
- When a user submits a query, the system computes its embedding and retrieves top-k relevant chunks based on cosine similarity.

The search layer is responsible for retrieving the most relevant content from insurance policy documents based on a user's natural language query. It utilizes a **vector database**—in our case, **Chroma DB**—to store and index dense vector embeddings of document chunks. These embeddings capture the semantic meaning of each chunk and are created using a transformer-based embedding model.

Design Choices:

- **Chroma DB** was selected for its **lightweight design**, **easy integration with Python workflows**, and **support for metadata filtering**, which allows fine-tuned retrieval from specific policy sections.
- We evaluated **hybrid retrieval** (BM25 + vector search), but in our use case, **semantic vector search alone yielded better accuracy**, especially for complex insurance-related queries.

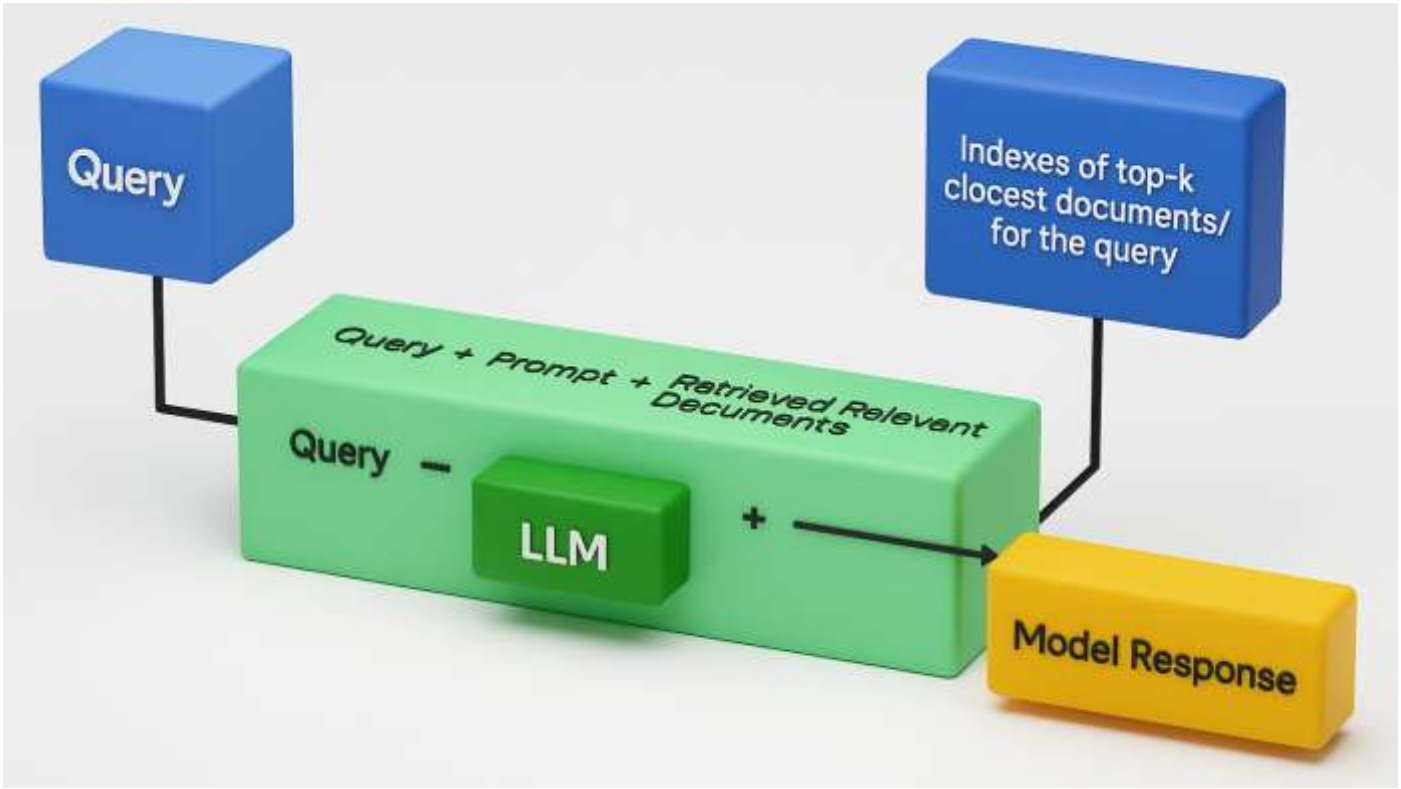


3. GENERATION LAYER

- The retrieved chunks are passed to an LLM (e.g., GPT-4 or GPT-3.5) along with the user's query.
- The model uses this contextual prompt to generate a grounded, coherent answer.

Prompt Engineering Techniques:

- System prompts emphasize factual accuracy and instruct the model to respond only using the provided context.
- Includes fallback responses like “Based on the document, this information is not available” for unanswerable queries.



📈 EVALUATION CRITERIA

To ensure effectiveness, the system is evaluated using the following metrics:

1. **Answer Relevance** – Does the answer directly address the user's question?
2. **Factual Accuracy** – Is the answer grounded in the source document?
3. **Completeness** – Does it include all relevant information from the retrieved context?
4. **Clarity** – Is the response understandable to a non-technical user?
5. **Handling Uncertainty** – Does the model avoid hallucinations and appropriately flag unanswerable questions?

A mix of **automated evaluations** (e.g., BLEU/ROUGE metrics) and **human judgment** (manual review of Q&A pairs) was used to validate the system.

🔍 KEY DESIGN DECISIONS

✓ USE OF RETRIEVAL-AUGMENTED GENERATION (RAG)

RAG allows for accurate answers grounded in real documents. It addresses one of the biggest weaknesses of LLMs: hallucination. By conditioning generation on real-world context, RAG ensures verifiability and legal defensibility—crucial in domains like insurance.

✓ CHUNKING STRATEGY

Chunk size dramatically affects both retrieval accuracy and generation quality. A chunk that's too large risks irrelevant content; too small, and it misses context. A chunk size of ~200–300 words balanced context and relevance effectively.

✓ LIGHTWEIGHT INFRASTRUCTURE

The initial implementation avoids complex frontends or microservices. A single Python script using modular functions allows for quick experimentation and deployment.

⚠ CHALLENGES FACED

1. PDF PARSING INCONSISTENCIES

Insurance policy PDFs vary greatly in layout. Tables, footnotes, and multi-column layouts posed extraction issues. `pdfplumber` worked better than `PyPDF2` or `fitz`, but post-processing was often required to clean up the text.

2. FACTUAL HALLUCINATION

LLMs can fabricate answers when context is incomplete. This was addressed by:

- Strictly grounding the prompt
- Limiting model creativity
- Returning fallback messages for unanswerable queries

3. LATENCY AND COST

Running large models like GPT-4 with retrieval introduces latency and API costs. Using smaller embedding models locally (e.g., `MiniLM`) and offloading generation to the cloud balanced performance and efficiency.

4. EVALUATION COMPLEXITY

Quantitatively evaluating semantic answers is challenging. A custom annotation tool was briefly prototyped for manual evaluation of relevance and accuracy, and future work may involve crowd-sourced evaluations.

EXTENSIBILITY AND FUTURE WORK

Mr. HelpMate AI is designed with extensibility in mind. Future enhancements could include:

- **User Feedback Loop:** Allow users to rate answers and fine-tune retrieval or prompts accordingly.
- **Web-based Interface:** A simple UI or chatbot for interaction, accessible via desktop or mobile.
- **Multilingual Support:** Use translation + embeddings to support policyholders in multiple languages.
- **Fine-tuned LLMs:** Train domain-specific models on insurance language for even better accuracy.

CONCLUSION

Mr. HelpMate AI demonstrates the powerful synergy between **document retrieval** and **generative AI** in making legal and technical documents more accessible to the average person. By focusing on real-world documents, factual accuracy, and user trust, the project provides not just a proof of concept, but a foundation for scalable, domain-specific AI assistants.

As insurance continues to grow more customer-centric, systems like Mr. HelpMate AI will be pivotal in delivering transparency, self-service, and cost efficiency—making understanding your policy as easy as asking a question.

CONTRIBUTIONS

This project is actively maintained by **Sagar Maru**, an experienced automation testing professional with a deep interest in AI-powered solutions for real-world challenges.

Contributions are welcome!
Whether it's **bug reports**, **feature suggestions**, or **code improvements**—every bit helps.

PROJECT LINKS

- **Kaggle Notebook:** Mr. HelpMate AI on Kaggle
- **GitHub Repository:** Mr. HelpMate AI on GitHub

☺ Thank-you ☺