Hybrid Retrieval-Augmented Generation for Structured Financial Data Analysis

Malak Ismail

May 7, 2025

Abstract

This whitepaper presents a semantic information retrieval system built upon a pipeline that leverages modern natural language processing (NLP) techniques to query structured client data in a human-interpretable manner. By integrating dense vector embeddings, semantic search with a FAISS index, reranking using a CrossEncoder, and text generation via a large language model, the system transforms tabular records into an interactive question-answering framework. The result is a pipeline capable of handling natural language queries over financial and demographic datasets, returning answers in natural text with high semantic relevance.

1 Introduction

As organizations increasingly generate vast quantities of structured data, traditional querying methods such as SQL often fall short of accommodating non-technical users. This whitepaper introduces a modern pipeline designed to bridge this gap by enabling natural language interaction with tabular datasets. The pipeline supports intelligent querying of client demographic and financial information, offering a retrieval-augmented generation (RAG) mechanism that combines semantic search and generative models. This approach simplifies data access while maintaining high fidelity in result relevance and interpretability.

2 System Architecture

2.1 Data Preprocessing and Text Generation

The pipeline begins by ingesting a CSV file containing client records. Critical columns such as income bounds are sanitized and cast to appropriate numeric formats. Missing values in first names are replaced with a placeholder to avoid null-value errors during embedding. Each client record is then synthesized into a descriptive sentence using a custom templating function, which aggregates attributes such as age, marital status, education level, and buying potential. These textual summaries serve as the foundation for subsequent embedding and retrieval stages.

2.2 Sentence Embedding

To enable semantic search, each client summary is encoded into a dense vector representation using the SentenceTransformer model 'all-mpnet-base-v2'. This model translates high-dimensional text into a low-dimensional embedding space, allowing for meaningful comparisons based on vector similarity. These embeddings are computed in batch and converted to NumPy arrays to facilitate efficient indexing.

2.3 FAISS-Based Semantic Search

The vectorized data is stored in a FAISS (Facebook AI Similarity Search) index, which is configured using L2 distance for fast nearest-neighbor search. At query time, the user's natural language input is encoded using the same sentence transformer and searched against the FAISS index to retrieve the top-k semantically similar records. This forms the retrieval phase of the pipeline, enabling scalable similarity-based lookup from thousands of records.

2.4 Cross-Encoder Re-Ranking

Although the FAISS index efficiently narrows down candidate records, it lacks nuanced contextual understanding. To improve accuracy, the retrieved candidates are reranked using the 'ms-marco-MiniLM-L-6-v2' CrossEncoder. This model takes the query and each candidate record as input pairs, scoring their semantic alignment with fine-grained contextual comparison. The candidates are then sorted based on these scores, ensuring that the most contextually relevant records are prioritized.

2.5 Response Generation

The final step of the pipeline employs a sequence-to-sequence language model, 'LaMini-Flan-T5-783M', to generate user-friendly answers. The top reranked records are formatted into a structured bullet list containing key client attributes. This context is then supplied along with the user query to the text generation model, which synthesizes an answer in natural language. This retrieval-augmented generation (RAG) approach ensures that the answer reflects both factual data and linguistic coherence.

3 Conceptual Design Principles

3.1 Interoperability Between Structured and Unstructured Data

At its core, the pipeline is designed to bridge structured tabular data with unstructured natural language interaction. By transforming records into descriptive sentences, the system leverages the power of large language models without losing the integrity of original tabular data.

3.2 Modular Semantics and Retrieval

The design emphasizes modularity between embedding-based retrieval and cross-encoder reranking. This dual-phase design allows for fast initial screening followed by precise reordering, balancing performance and accuracy. Each stage operates independently, making the pipeline both scalable and extensible.

3.3 Natural Language Grounding for Explainability

The choice to summarize records as natural language text enhances model interpretability. Rather than working on abstract numerical vectors or database identifiers, every component of the pipeline operates on human-readable sentences, ensuring traceability and facilitating debugging or auditing.

3.4 System Design Considerations

Several engineering decisions underpin the robustness and flexibility of the pipeline. The use of FAISS ensures scalability for large datasets, while batch encoding accelerates the embedding process. The reranking phase, though computationally heavier, significantly enhances precision and is optimized by limiting the number of candidate pairs. The QA model is deployed on a GPU to enable real-time inference, which is essential for practical deployment. The input prompt is carefully crafted to guide the generative model toward concise and structured answers, improving consistency across outputs.

4 Limitations

Despite its effectiveness, the system has limitations. First, it is memory-intensive, especially during the embedding and reranking phases. Second, generative models, while fluent, may hallucinate or misinterpret context under ambiguous conditions. Additionally, the pipeline does not include fine-tuning mechanisms for domain-specific adaptation, relying solely on pre-trained models. The FAISS index also assumes static data; updates require re-embedding and rebuilding the index. Lastly, while the prompt-based generation is powerful, it can be sensitive to slight changes in wording, potentially affecting output stability.

5 Future Improvements

Several avenues exist for enhancing the pipeline. Incorporating dynamic indexing with approximate nearest neighbor search would allow for real-time updates to the data. Fine-tuning the generative and cross-encoder models on domain-specific examples could improve output accuracy. A hybrid query parser could allow users to mix natural language with structured filters, enhancing expressiveness. Additionally, integrating user feedback loops for continual learning would allow the system to improve over time based on usage patterns. Finally, deploying the entire pipeline as a service with REST or GraphQL endpoints would improve accessibility for real-world applications.

6 Conclusion

This technical whitepaper has presented a comprehensive semantic retrieval and generation pipeline capable of transforming structured client data into an intelligent, interactive system. By combining embedding models, similarity indexing, cross-encoding, and large language generation, the system allows users to ask complex natural language questions and receive accurate, interpretable answers. Through careful design and modular integration, the pipeline demonstrates the feasibility and power of combining structured data with modern NLP techniques. While limitations exist, the architecture provides a strong foundation for further development and practical deployment in data-driven environments.

7 Models

1. Sentence Embedding: all-mpnet-base-v2 model from the SentenceTransformers library (Hugging Face).

2. Cross-Encoder: The cross-encoder/ms-marco-MiniLM-L-6-v2 model was used for reranking (Hugging Face).

3. Text Generation: The LaMini-Flan-T5-783M model was used for response synthesis (Hugging Face).

4. FAISS Indexing: We leveraged the FAISS library for efficient similarity search (GitHub).