**SEMANTIC SPOTTER**

**Project Objective:**

Build a project in the insurance domain, similar to the project you saw in the ‘Retrieval Augmented Generation’ session. The goal of the project will be to build a robust generative search system capable of effectively and accurately answering questions from various policy documents. You may use LangChain or LlamaIndex to build the generative search application.

**Solution Strategy:**

Build a solution which should solve the following requirements using LlamaIndex:

● Users would responses from insurance policy knowledge base.

● If user want to perform a query system must be able to response to query accurately.

**Goals:**

Solving the above two requirements well in and would ensure that the accuracy of the overall model is good.

**Data Used:**

HDFC various insurance policy documents.

Tools Used:

LlamaIndex, ChatGPT, disc cache has been used due to its powerful query engine, fast data processing using data loaders and directory readers as well as easier and faster implementation using fewer lines of code.

**Why LlamaIndex?**

LlamaIndex is an innovative data framework specially designed to support LLM-based RAG framework application development. It offers an advanced framework that empowers developers to integrate diverse data sources with large language models.

LlamaIndex includes a variety of file formats, such as PDFs and PowerPoints, as well as applications like Notion and Slack and even databases like Postgres and MongoDB.

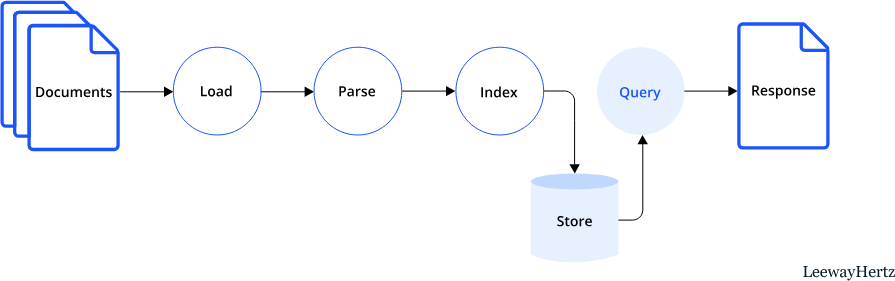
The framework brings an array of connectors that assist in data ingestion, facilitating a seamless interaction with LLMs. Moreover, LlamaIndex boasts an efficient data retrieval and query interface.

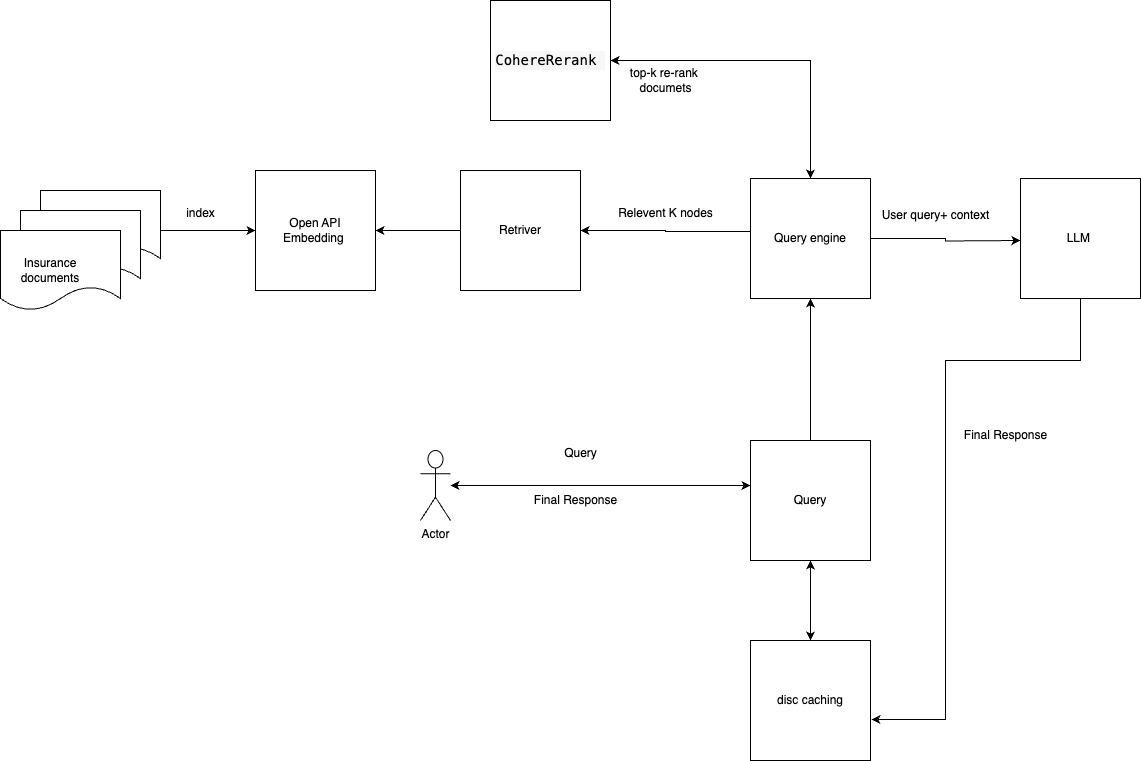
LlamaIndex enables developers to input any LLM prompt and, in return, receive an output that is both context-rich and knowledge-augmentation.

**Key Feature of LlamaIndex:**

● Data connectors allow ingestion from various data sources and formats.

● It can synthesize data from multiple documents or heterogeneous data sources.

● It provides numerous integrations with vector stores, ChatGPT plugins, tracing tools, LangChain, and more.

**High Level Design of Semantic Spotter:** 

**Architecture Descriptions:**

* Documents: We will be using list of HDFC insurance documents provides inside a single folder.
* Open API embedding: We are using OpenAPI embedding as Vector DB for indexing insurance documents in the form of embedding.
* Query Engine: We are using Query Engine Module of Llammaindex for performing
* semantic search. Query Engine will use internally retrieve and sentence transformer re-rank to retrieve top-k relevant nodes from embedding.
* LLM : top k-documents along with user query will be passed to LLM to generate the accurate response. We are using ChatGPT LLM.
* Caching:" Caching is being used to improve the read operation. Recent similar search will be store in Caching and user query first will be served from caching. If user query

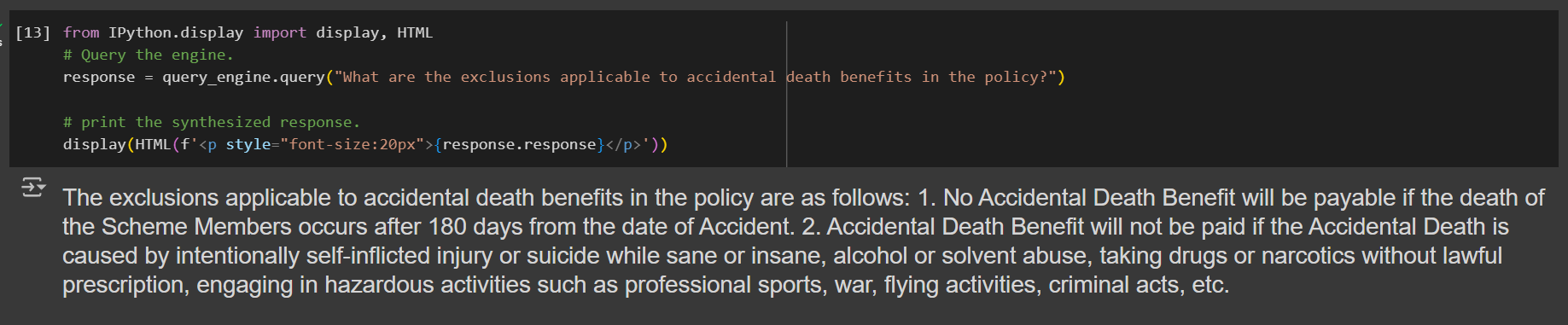
not found in cache then query will be forwarded to query engine and then LLM to generate

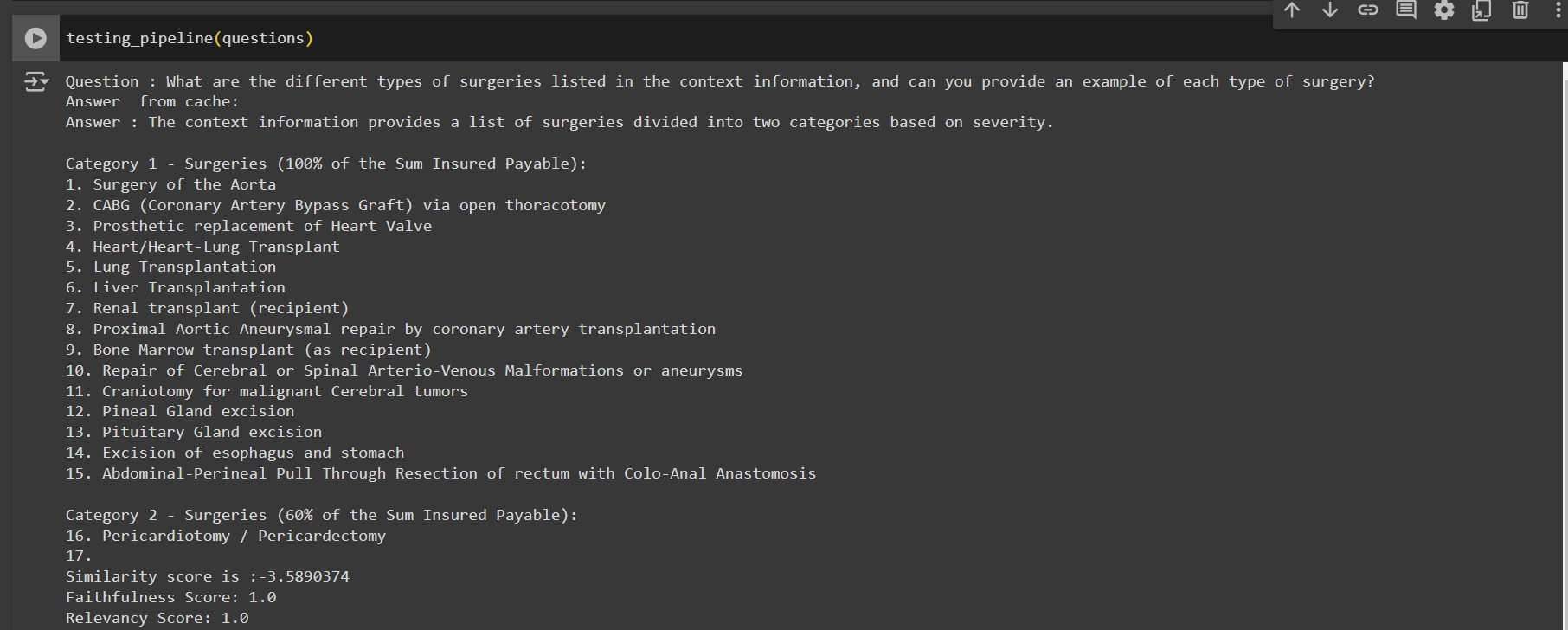
response. User query and generated response will be cached in in cache and will be served from there based ttl.

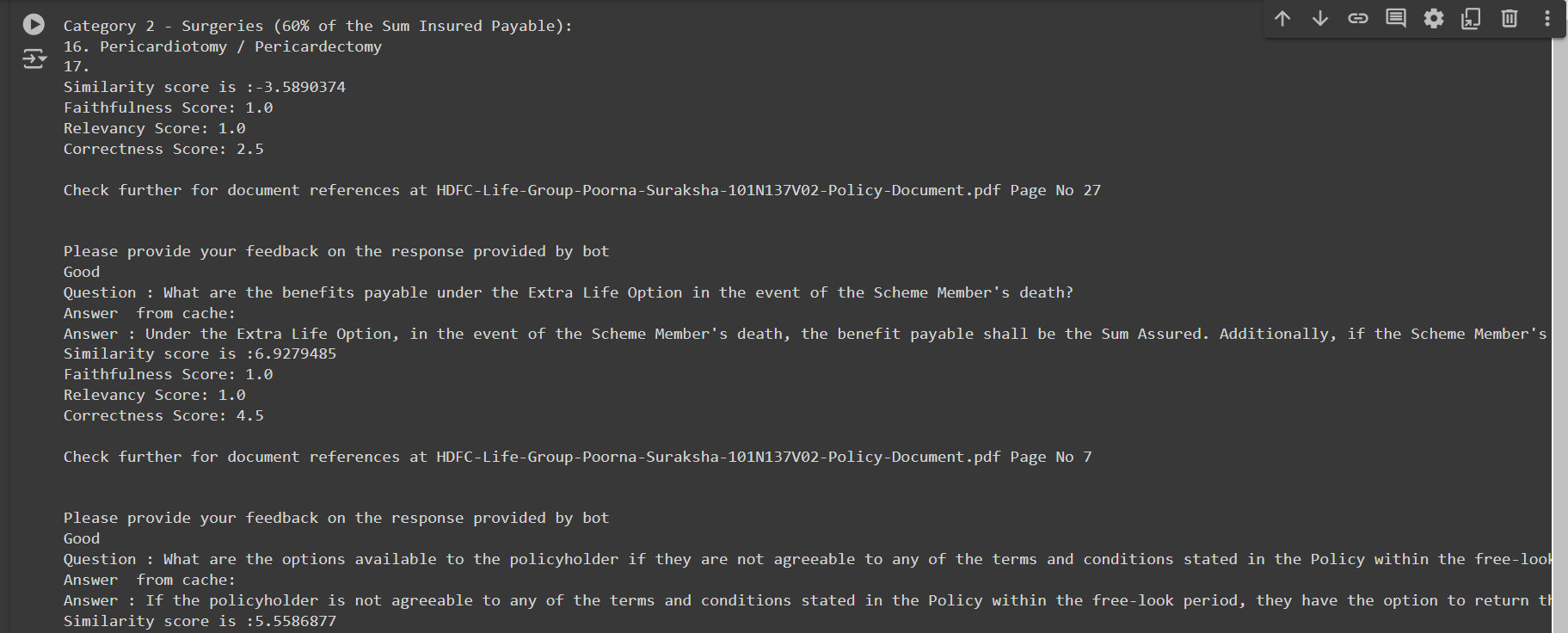
* Meta data: Along with response we are also returning docs reference and similarity score, faithfulness, relevancy and correctness score to improve the user confidence towards the implemented RAG system.
* Sentence Transformer Re-rank: Is being used to re-rank the query based on semantic score.

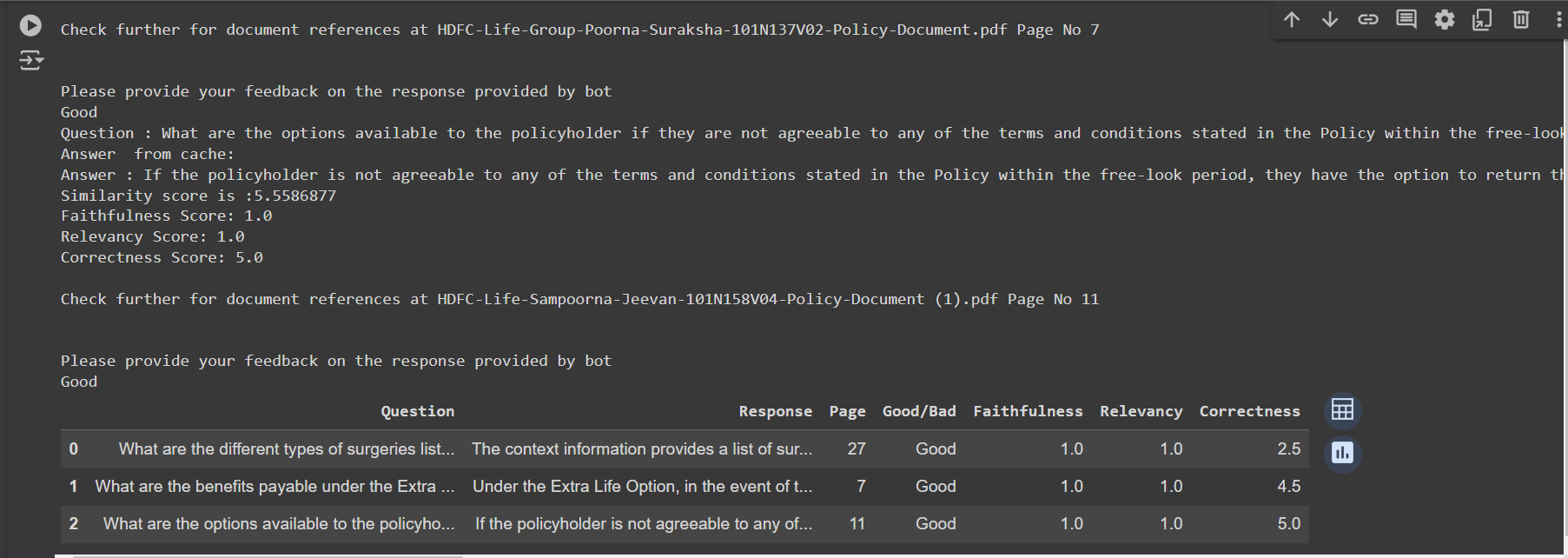
**Generative Search Response from Insurance documents**:

**1. Using a single Query Response:**



**2. Multiple Query Response:**





AUTHOR: Gaurav Dhuri.