Semantic Search

Table of Contents

[1. Problem Statement 1](#_Toc170853928)

[2. Overall System Design 2](#_Toc170853929)

[3. Code Implementation 3](#_Toc170853930)

[3.1 Checking response and response parameters 3](#_Toc170853931)

[3.2 Creating a response Pipeline 3](#_Toc170853932)

[3.3 Build a Testing Pipeline 3](#_Toc170853933)

[4. Documentation 4](#_Toc170853934)

# Problem Statement

The goal of the project will be to build a robust generative search system capable of effectively and accurately answering questions from a list of policy documents. Our task is to build a proper Q/A bot using RAG, that the users can interact with and get answers from.

Solution Strategy

Build a POC which should solve the following requirements:

Users would responses from list of policy documents

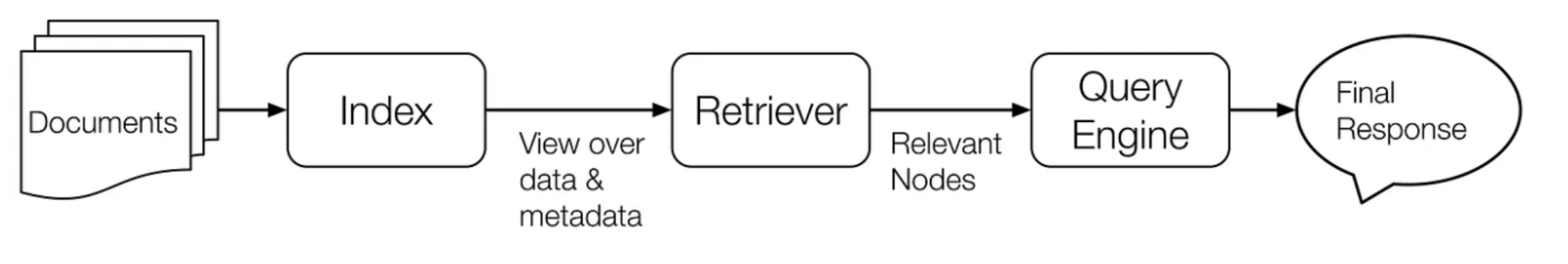
Data Used

List of multiple policy document PDFs

Tools used

LlamaIndex 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.

# Overall System Design



The key building blocks of LlamaIndex, a library designed for efficient and scalable indexing of large language models, can be summarized as follows:

1. **Nodes**: Nodes are the basic unit of data representation in LlamaIndex. Each node contains a piece of text or data, along with metadata and other attributes.
2. **Indices**: Indices are structures that organize nodes to enable efficient querying and retrieval. There are different types of indices, such as keyword-based indices, embedding-based indices, and hybrid indices.
3. **Documents**: Documents are collections of nodes. A document can be a single large piece of text, such as an article or a book, that is broken down into smaller nodes for indexing and retrieval.
4. **Query Engines**: Query engines are responsible for processing queries against the indices. They use various algorithms and strategies to retrieve the most relevant nodes based on the query.
5. **Indexers**: Indexers are responsible for creating and updating indices. They take documents, break them down into nodes, and organize these nodes into indices for efficient retrieval.
6. **Retrieval Models**: Retrieval models determine how queries are matched with nodes in the indices. Common models include vector space models, probabilistic models, and neural network-based models.
7. **Scalability Components**: These components ensure that the indexing and querying processes can handle large-scale data efficiently. They include techniques for distributed computing, parallel processing, and data partitioning.
8. **APIs and Interfaces**: LlamaIndex provides APIs and interfaces for developers to interact with the indexing system. These interfaces allow for easy integration of the indexing capabilities into various applications.
9. **Optimization Modules**: These modules focus on optimizing the performance of the indexing and retrieval processes. They include techniques for query optimization, index compression, and caching.
10. **Security and Access Control**: Components that manage access control, authentication, and data security to ensure that only authorized users can access and modify the indices.

These building blocks work together to provide a robust and scalable indexing solution for large language models, enabling efficient retrieval of relevant information from vast datasets.

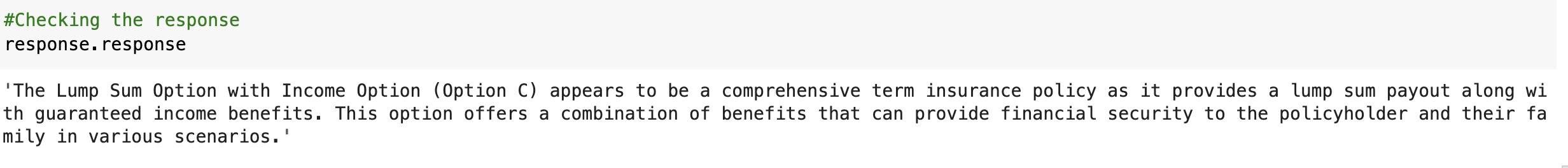
# Code Implementation

## 3.1 Checking response and response parameters

Question

“which is the best term insurance policy?”

Response

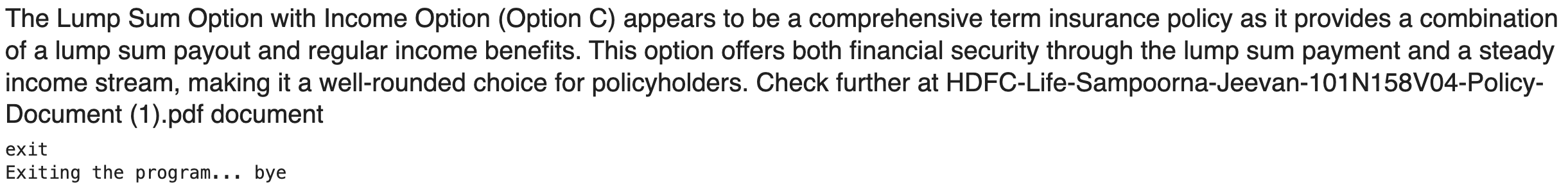


## 3.2 Creating a response Pipeline

Question

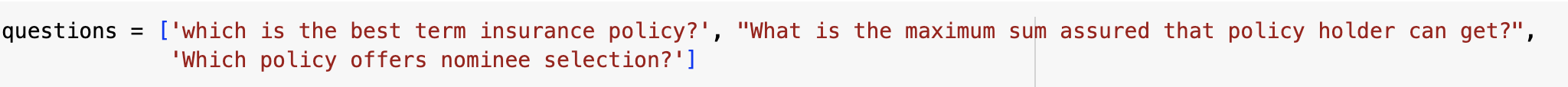
“Which is the best term insurance policy?”

Response

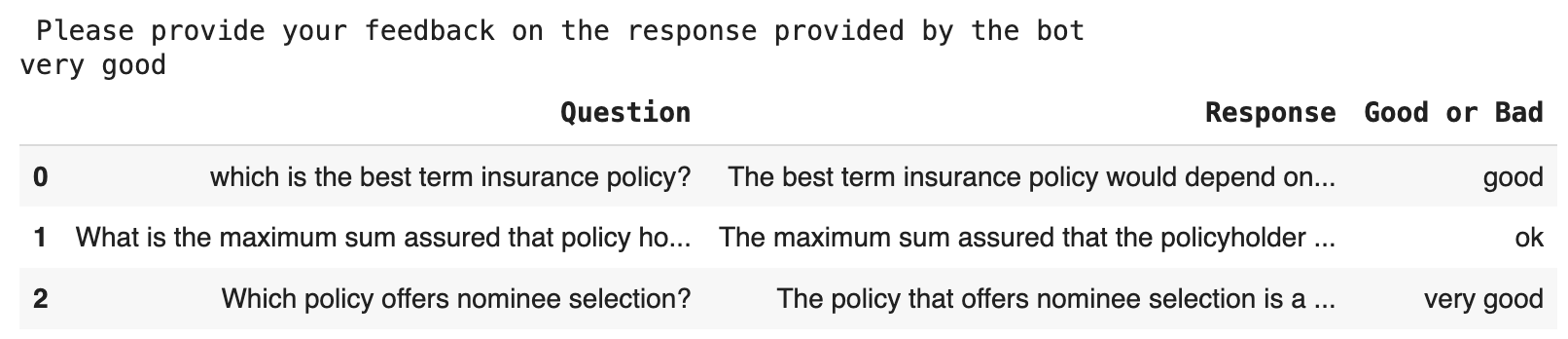


## 3.3 Build a Testing Pipeline

Questions



Responses

****

# Documentation

