# Question-Answering System Using Language Models and Document Retrieval

## Introduction

This document outlines the development of a question-answering (QA) system that leverages language models and document retrieval techniques. The system's primary goal is to provide accurate and contextually relevant answers to user questions based on a collection of documents. It combines the power of transformer-based language models with efficient document indexing and retrieval methods.

## Objective

The main objective of this code is to create an effective QA system that can answer user questions using a pre-trained language model and a database of documents. The system must be capable of handling a wide range of questions and retrieving relevant information from the documents to formulate responses.

## Libraries Used

The code relies on several Python libraries and frameworks to achieve its objectives:

- **langchain**: Langchain is a library for building natural language processing pipelines and workflows. It provides tools for document loading, text splitting, embeddings, and more.

- **unstructured**: This library extends langchain to support document formats like DOCX and PDF, making it easier to load and process unstructured data.

- **pytesseract**: PyTesseract is used for optical character recognition (OCR) to extract text from images.

- **sentence_transformers**: Sentence Transformers provides pre-trained models for generating sentence embeddings, which are useful for semantic similarity calculations.

- **llama_index**: Llama Index is a vector indexing library used for efficient document retrieval.

- **faiss-cpu**: FAISS is used for fast similarity search on large-scale vector data.

- **pypdf**: PyPDF is used to extract text from PDF documents.

- **transformers**: This library provides pre-trained transformer-based language models.

- **openai**: The OpenAI API is used for natural language generation and chatbot capabilities.

## Steps Overview

The code can be divided into several key steps:

١. **Installation**: In this step, the required Python libraries and packages are installed using the `pip` tool. These libraries provide essential functionalities for text processing, embeddings, and document retrieval.

٢. **Document Loading**: The code loads a collection of documents from a specified directory using the `DirectoryLoader` provided by langchain. The documents are prepared for further processing.

٣. **Text Splitting**: The text content of the documents is split into smaller chunks to optimize processing. The `TokenTextSplitter` is used to split the text into manageable segments.

۴. **Embeddings Generation**: Sentence embeddings are generated for the text chunks using the OpenAI GPT-3.5 Turbo model. These embeddings capture the semantic meaning of the text, enabling efficient similarity calculations.

۵. **Document Indexing**: The text chunks and their corresponding embeddings are indexed using the Chroma vector store from llama_index. This indexing enables fast and efficient retrieval of relevant documents based on user queries.

۶. **Question-Answering Chain**: A retrieval-based question-answering chain is constructed using the indexed documents and the GPT-3.5 Turbo model. This chain is capable of accepting user questions, retrieving relevant documents, and generating answers.

۷. **Question-Answering Function**: A function is defined to accept user questions, pass them through the QA chain, and return the generated answers.

۸. **Main Function**: The main function demonstrates the system's functionality by posing a sample question and retrieving an answer.

## Details in Each Step

Let's delve into each step in more detail:

### Installation

- Required Python libraries are installed using `pip`.

### Document Loading

- Documents are loaded from a specified directory using the `DirectoryLoader`.

### Text Splitting

- The text content of the documents is split into smaller chunks to optimize processing. Chunks are generated using the `TokenTextSplitter`.

### Embeddings Generation

- Sentence embeddings are generated for the text chunks using the GPT-3.5 Turbo model from OpenAI.

### Document Indexing

- The text chunks and their embeddings are indexed using the Chroma vector store, enabling efficient document retrieval.

### Question-Answering Chain

- A retrieval-based question-answering chain is constructed using the GPT-3.5 Turbo model and the document retriever.

### Question-Answering Function

- A function is defined to accept user questions, pass them through the QA chain, and return the generated answers.

### Main Function

- The main function demonstrates the system's functionality by posing a sample question and retrieving an answer.