Submitted By: Vivek Singh

Data Scientist – Deutsche Bank

Question Answer Based Chat Bot Documentation

Overview

This is a **Question-Answer (QA) Based Chat Bot** that provides answers based on the input PDFs provided to the system. The bot uses **ElasticSearch** for storing data, **LlamaIndex** for document parsing and dense retrieval, and **FastAPI** for creating endpoints to handle user queries and evaluate the system's performance. The system can answer user queries strictly based on the content of ingested PDF documents and evaluate the correctness of its responses using custom-defined metrics inspired by the **RAGAS** research paper.

Input PDF Files:

1. **GPT - Expense Reimbursement Policy.pdf**
2. **GPT - Grievance and Disciplinary Policy.pdf**
3. **GPT - Leave Policy.pdf**
4. **GPT - Travel Reimbursement Policy.pdf**
5. **GPT - ESOPS Policy.pdf**
6. **GPT - Information Security and IT Policy.pdf**
7. **GPT - POSH Policy.pdf**
8. **GPT - Recruitment and Onboarding Policy.pdf**
9. **GPT - Remote Work Policy.pdf**
10. **GPT - Parental Leave Policy.pdf**

Key Components

1. ElasticSearch Vector Database

- **Purpose:** Used to store parsed document data in vector format.
- **Index:** Named `policy_documents_dense`. It stores dense vector embeddings of PDF documents for faster retrieval based on cosine similarity.

2. LlamaIndex Framework

- **Document Parsing:** Utilizes the **LLAMA Parser** to extract content from the input PDF files.
- **Dense Retrieval:** Uses a dense retriever from the **LlamaIndex** to extract relevant document chunks based on the user's query.

3. FastAPI Endpoints

- **/ask_question:** Accepts a user question and retrieves the answer strictly from the parsed PDF documents.
- **/evaluate:** Evaluates the QA pipeline based on predefined golden questions and answers extracted from the input PDF documents.

System Workflow

1. Data Ingestion Process

- **ElasticSearch Index Creation:** An ElasticSearch index named `policy_documents_dense` is created if it does not already exist.
- **Duplicate Prevention:** The system calculates the **SHA-256 hash** for each PDF file to avoid duplicate ingestion. If the hash already exists, the document is not ingested again.
- **Metadata Handling:** The title of the PDF document is taken as the document name in the metadata.
- **Parsing and Storing:** PDF files are parsed using the **LLAMA Parser** and ingested into the ElasticSearch index.

2. Query Handling (/ask_question)

- **Process:**
    - A user submits a question to the `/ask_question` endpoint.
    - The system searches for relevant chunks in the ElasticSearch index using cosine similarity.
    - A dense retriever from **LlamaIndex** retrieves relevant document chunks.
    - The system generates and returns the answer to the user, along with the retrieved context.
- **Response Structure:**
    - **Answer:** The generated answer from the ingested documents.
    - **Context:** Relevant chunks (document context) used to derive the answer.

3. RAG Evaluation (/evaluate)

- **Purpose:** Evaluates the QA system based on predefined golden questions and golden answers.
- **Process:**
    1. A dictionary of questions and their corresponding golden answers is provided.
    2. The system evaluates each answer generated for the given questions.
    3. **Custom Evaluation Metrics** are used to compute scores based on the **RAGAS** research paper.

Evaluation Metrics

1. Faithfulness

- Evaluates how well the generated answer aligns with the retrieved context.
- **Formula:** Ratio of statements in the generated answer that are supported by the context.
- **Returns:** `faithfulness_score` (float)

2. Answer Relevance

- Measures the relevance of the generated answer to the input question.
- **Formula:** Average cosine similarity between the original question and questions generated from the answer.
- **Returns:** `answer_relevance_score` (float)

3. Context Relevance

- Evaluates the relevance of the context retrieved to the input question.
- **Formula:** Ratio of relevant sentences in the context to the total number of sentences.
- **Returns:** `context_relevance_score` (float)

Running the QA System

1. Setup Instructions:

1. Extract the ZIP file.

Open the terminal and navigate to the directory:
bash

```
cd QA_System/app
```

2.

Run the Docker command to build and start the system:
css

```
docker-compose up --build
```

3.

2. Components Running:

- **ElasticSearch** will be triggered through the Docker Compose file (default port: `9200`).
- **FastAPI Server** will be launched to handle API requests (default port: `8000`).

3. Accessing Endpoints:

You can access the FastAPI **Swagger UI** for testing:
bash
Copy code
http://localhost:8000/docs#/

Endpoint Details

1. /ask_question

- **Description:** Answers user questions by retrieving relevant chunks from the documents.

Example Query:
arduino

```
How many personal leaves do I have?
```

2. /evaluate

- **Description:** Evaluates the RAG pipeline based on golden questions and answers.

Input Example:
json:

```
{
  "questions": [
    "What is the eligibility requirement for paid sick leave at
Simpplr, and how many days of paid sick leave are employees entitled
to?",
    "What is the time limit for submitting travel expense reports at
Simpplr?",
    "What is Simpplr’s policy on reporting sexual harassment
incidents?",
    "What is the eligibility requirement for employees to request
remote work at Simpplr?",
    "What is the eligibility requirement for parental leave at
Simpplr?",
    "What is the timeframe for submitting expense claims at Simpplr?",
    "What is Simpplr’s process for internal transfers?"
  ],
  "golden_answers": [
    "Employees are eligible for up to 10 days of paid sick leave per
year after six months of continuous employment.",
```

```
    "Employees must submit travel expense reports within 14 days of completing their trip.",
    "Employees must report sexual harassment incidents promptly to their supervisor, manager, or HR representative. Reports are treated confidentially.",
    "Employees must have completed at least six months of continuous employment and demonstrate the ability to work independently.",
    "Employees must have been employed for at least six months continuously to be eligible for parental leave.",
    "Employees must submit expense claims within 30 calendar days of incurring the expense.",
    "Simpplr encourages internal transfers, and job postings are communicated regularly. Selection is based on qualifications, skills, and experience."
  ]
}
```

- **Notes:** This endpoint takes approximately 30-40 seconds to run the evaluation and return the metrics.

Important Notes

- Ensure **port 9200** is free for ElasticSearch.
- Do not change the provided API keys to ensure the tool runs as expected.
- The system may take some time for processing large data or evaluations—please be patient.

Other Approaches:

Approach 1: Using a Pre-trained Document Retrieval and QA Model (Haystack with FARM/BERT)

In this approach, we can leverage pre-trained language models designed for document retrieval and QA tasks.

Components:

- **Haystack Framework**: Utilising it for document search, QA, and evaluation.
- **BERT Models**: Using it for document embedding and QA tasks
- **Document Store**: **Postgres Vector Database** for storing document vectors.
- **Retrieval-Augmented Generation (RAG)**: Can use combination of denser and spare retrieval techniques

Approach 2: Knowledge Graph and QA System Using Neo4j and GPT

In this we can build **knowledge graph** from the policy documents and then using a **QA model** to extract and infer knowledge from the graph.

Components:

- **Neo4j**: Graph database to build knowledge graph for policy documents
- **Entity and Relationship Extraction**: Can use classical NLP techniques to extract the entities
- **QA Model**: A fine-tuned GPT-based mode to perform question answering over the knowledge graph.