Technical Documentation for the Question-Answering System

1. Introduction

This is a technical documentation for the question-answering system. The system is designed to answer user questions about a given corpus of text. The system leverages SentenceTransformers models for generating embeddings, ElasticSearch as a vector store, Docker for containerization and deployment, and Flask for the API.

This document covers the following topics:

System overview
Installation and setup
Directory structure
System components
Usage instructions
API documentation
Conclusion

2. System Overview

2.1 Architecture

The system consists of the following components:

- Parsing component: Parses the given corpus of text and extracts passages.
- Embeddings generation component: Generates passage embeddings using SentenceTransformers models.
- ElasticSearch integration component: Indexes the passage embeddings in ElasticSearch.
- Document retrieval component: Retrieves relevant passages from ElasticSearch based on the user query.
- Docker containerization: Containerizes the system for easy deployment.
- API deployment: Deploys a Flask API for user interaction.
- Evaluation component: Evaluates the performance of the system using a given evaluation dataset.

2.2 Key Components

SentenceTransformers: A Python library for natural language processing and machine learning.

ElasticSearch: A distributed search and analytics engine.

Docker: A containerization platform.

Flask: A Python web framework.

3. Installation and Setup

3.1 Prerequisites

Python 3.6 or higher SentenceTransformers ElasticSearch Docker Flask

3.2 Installation Steps

- 1. Clone the repository.
- 2. Navigate to the project directory.
- 3. Install the required Python packages using pip.

pip install -r requirements.txt

4. Start the ElasticSearch server.

./bin/elasticsearch

5. Build the Docker image.

docker build -t question-answering.

6. Run the Docker container.

docker-compose up

3.3 Configuration

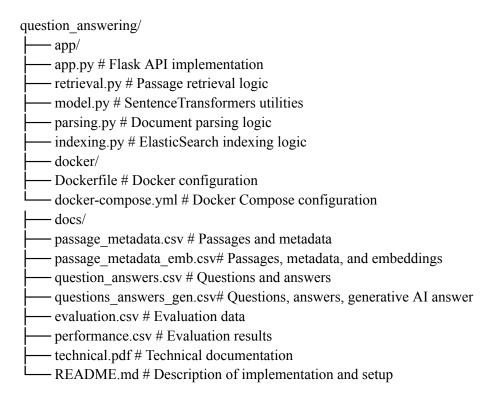
You can configure the system by editing the following files:

`config.py`: This file contains the configuration parameters for the system.

'Dockerfile': This file contains the Docker configuration for the system.

4. Directory Structure

The project directory structure is as follows:



5. System Components

5.1 Parsing Component

5.1.1 Description

The parsing component parses the given corpus of text and extracts passages. The component uses a regular expression-based approach to extract passages.

5.1.2 Usage

To use the parsing component, run the following command:

python parsing.py

This will parse the input file and save the extracted passages to the output file.

5.2 Embeddings Generation

5.2.1 Description

The embeddings generation component generates passage embeddings using SentenceTransformers models. The component uses the `SentenceTransformer` class from SentenceTransformers to generate the embeddings.

5.2.2 Usage

To use the embeddings' generation component, run the following command:

python model.py

This will generate passage embeddings from the given passage metadata file and save them to an output file

5.3 ElasticSearch Integration

5.3.1 Description

The ElasticSearch integration component indexes the passage embeddings in ElasticSearch. The component uses the `elasticsearch`

5.3.2 Usage

To use the ElasticSearch integration component, run the following command:

python indexing.py

This will index the passage embeddings from the given passage metadata embeddings file into ElasticSearch.

- 5.4 Document Retrieval
- 5.4.1 Description

The document retrieval component retrieves relevant passages from ElasticSearch based on the user query. The component uses the 'elasticsearch' library to query ElasticSearch and retrieve the relevant passages.

5.4.2 Usage

To use the document retrieval component, run the following command:

python retrieval.py

This will retrieve the relevant passages from ElasticSearch based on the given query.

5.5 Docker Containerization

5.5.1 Description

The Docker containerization component containerizes the system for easy deployment. The Dockerfile defines the Docker configuration for the system.

5.5.2 Usage

To build the Docker image, run the following command:

docker build -t question-answering .

To run the Docker container, run the following command:

docker-compose up

5.6 API Deployment

5.6.1 Description

The API deployment component deploys a Flask API for user interaction. The Flask API provides endpoints for submitting queries and retrieving the results.

5.6.2 Usage

To start the Flask API, run the following command:

python app.py

...

- 6. Usage Instructions
- 6.1 Running the System

To run the system, follow these steps:

- 1. Start the ElasticSearch server.
- 2. Build the Docker image.
- 3. Run the Docker container.

Once the Docker container is running, you can access the system through the Flask API.

7. Conclusion

The question-answering system can be used to answer user questions about a given corpus of text.