School of Engineering & Technology

Department of Computer Science & Technology

Agentic AI Lab (CSCR3214)

Lab File (2025-2026)

for

B. Tech. (CSE)

6th Semester

Lab Faculty details:

Ayush Kumar Singh

Department of Computer Science & Engineering
School of Engineering & Technology Sharda University

Student details:

Mayank Raj

B. Tech. CSE [6th Semester] 2023482186
2023 - 27

RAG-Based Transit Information System

1. Project Overview

This project implements a **Retrieval-Augmented Generation (RAG)** system designed to provide intelligent, context-aware answers regarding a public transportation monitoring system.

The core functionality revolves around processing specific documentation or knowledge bases related to bus transit operations—specifically focusing on **real-time bus tracking, estimated arrival times (ETA), and passenger occupancy detection**. By leveraging RAG, the system retrieves relevant chunks of information from these source documents to augment the responses of a language model, ensuring accuracy and grounding the answers in the provided data.

Key Capabilities:

- **Information Retrieval:** Semantically searches a knowledge base for relevant query context.
- **Contextual Answering:** Answers user queries about specific transit metrics like "Bus location," "Passenger count," and "Seat availability".
- **Source Attribution:** The system is designed to reference specific information sources (e.g., "Source 1," "Source 2") to validate its answers.

2. Tools & Libraries Used

The project utilizes a modern Python-based Natural Language Processing (NLP) stack.

- **Development Environment:**
  - **Jupyter Notebook / Google Colab:** The primary interface for development, utilizing GPU acceleration (Tesla T4 detected) for efficient model inference.
  - **Python 3:** The underlying programming language.
- **Core Libraries:**
  - **`sentence-transformers`:** Used for generating dense vector embeddings of the text data. The specific model identified is `all-MiniLM-L6-v2`, a highly efficient model optimized for semantic search and clustering.
  - **`transformers` (Hugging Face):** Provides the foundational architecture for loading and managing pre-trained language models.
  - **`huggingface-hub`:** Facilitates the downloading and management of model weights and configurations from the Hugging Face Hub.
  - **`LangChain`:** The code references `langchain-community` and `langchain-core`, indicating this framework is likely used for chaining the retrieval and generation steps.

3. Instructions to Run the Notebook

Follow these steps to successfully execute the project code.

Prerequisites

- **Python 3.8+** environment.
- **GPU Access (Recommended):** While the code can run on a CPU, using a GPU (like in Google Colab) is significantly faster for generating embeddings.

Installation

If running locally, you must install the required dependencies. Run the following command in your terminal or a notebook cell:

Bash

```
pip install sentence-transformers transformers huggingface-hub langchain-community
```

Execution Steps

1. **Open the Notebook:** Load `RAG_PROJECT-2.ipynb` into Jupyter Lab, Jupyter Notebook, or upload it to Google Colab.
2. **Environment Setup:** Ensure your runtime type is set to **GPU** if available (in Colab: *Runtime > Change runtime type > T4 GPU*).
3. **Run Cells Sequentially:**
    - **Import & Install:** Execute the initial cells to install libraries and import necessary modules.
    - **Model Loading:** Allow the `sentence-transformers` model (`all-MiniLM-L6-v2`) to download. This may take a moment (approx. 90MB).
    - **Data Ingestion:** The subsequent cells define the source text (documents regarding bus tracking) and create embeddings.
    - **Querying:** Go to the final section of the notebook to input your questions (e.g., "*What type of data is collected in the DTC bus tracking system?*") and view the generated responses.
4. **Troubleshooting:**
    - *Hugging Face Token Warning:* You may see a warning about unauthenticated requests. This is generally harmless for public models like `all-MiniLM-L6-v2`, but setting a `HF_TOKEN` environment variable can resolve it if you hit rate limits.