**Project Documentation: SIT RAG Chatbot**

This project is a Retrieval-Augmented Generation (RAG) system designed to answer questions related to university knowledge. It combines:

**FAISS**: Semantic vector-based document retrieval

**BM25**: Keyword-based retrieval using token matching

**Gpt-4.1-mini**: Generates final answers grounded in retrieved context

**Project Structure:**

├── combined\_faiss/

│ ├── index.faiss # FAISS vector index (binary format)

│ └── index.pkl # Document metadata for FAISS

├── data/

│ ├── vector-index-lancedb # Location of LanceDB

│ └── bm25\_index.lance #folder containing tables and data of bm25\_index

└── faiss\_index.lance #folder containing tables and data of faiss\_index

├── bm25\_index.pkl # Prebuilt BM25Okapi index with texts and metadata

├── bm25\_helper.py # BM25 search function

├── query\_test.py # Main script to run interactive queries

├── lancedb\_bm25\_migration.py # Main script to store or migrate FAISS/Bm25 data

│ to LanceDB

├── abbreviation\_cache.json # Stores abbreviation expansions (auto-generated)

├── .env # Environment variables (OpenAI key)

└── requirements.txt # Python dependenies

**How to Set Up and Run**

**Step 1: Install required packages:**

***\*Run this code in your terminal\****

pip install -r requirements.txt

Make sure requirements.txt contains at least the following:

openai

langchain

rank\_bm25

nltk

tiktoken

python-dotenv

For first-time setup, you may also need to download the NLTK resources:

import nltk

nltk.download("punkt")

nltk.download("stopwords")

**Step 2: Set up Your OpenAI API Key**

Create a .env file with the following line:

OPENAI\_API\_KEY=your\_openai\_key\_here

Replace “your\_openai\_key\_here” with your actual OpenAI key.

**Step 3: Run the application**

python query\_test.py

You will see this in your terminal:

Your question:

Type your question in and get the generated reply

**Explanation of each file:**

**query\_test.py**

* Main script for interacting with the system
* Rewrites queries, retrieves relevant documents from FAISS and BM25
* Truncates results to token limit and sends them to GPT
* Measures response time and shows debug info

**bm25\_helper.py**

* Loads and searches the pre-built BM25 index (bm25\_index.pkl)
* Performs keyword-based ranking using rank\_bm25
* Efficient, minimal implementation with optional caching

**combined\_faiss/**

* Stores:
  + index.faiss: binary FAISS vector index
  + index.pkl: accompanying metadata (document text, program info, etc.)
* Required for FAISS vector search

**bm25\_index.pkl**

* Serialized file containing:
  + BM25Okapi model
  + List of documents
  + Metadata dictionaries
* Used in bm25\_helper.py for fast retrieval

**data/**

* File and folder location of LanceDB tables and data
* Stores vector-index-lancedb folder containing:
  + Bm25\_index.lance tables and data
  + Faiss\_index.lance folder tables and data
* Required for FAISS vector and BM25 retrieval using LanceDB as the database storage

**lancedb\_bm25\_migration.py**

* Script to store or migrate vector data of FAISS Index and BM25 into a LanceDB database.
* Chunk-sized batch processing for faster operations.
* Requires the root folder to have the FAISS index and bm25 files.

**abbreviation\_cache.json**

* Auto-generated file that stores full forms of abbreviations
* Prevents repeated user prompts

**.env**

* Contains your OpenAI API key
* Used by dotenv to load environment variables