Data Aggregation Platform

1. Installation (libraries, Ollama, DB)

1. Create new Environment
   - Created a env named **data_aggregation** with conda
   - Activate the env: `conda activate data_aggregation`

   If working with venv use this to create the environment - python(3) -m venv data_aggregation
2. Install the libraries with the command: `pip install -r requirements.txt`
3. Create a database (name = data_aggregation_platform) in the Postgres or database of choice and add tables using the **CREATE TABLES.sql** file or **create_tables.py** file
4. Download ollama from ollama.org and pull models
   - `ollama pull bge-m3 -> embedding model`
   - `ollama pull orca-mini:7b-q2_K -> summarizer`
   - `ollama pull gemma:2b -> chat model`

main.py

Endpoint: **/rag/process_query** - Handles POST requests with a query string. This query is processed using the RAGSystem module.

rag_system/rag_module.py

The **RAGSystem** class processes user queries through a pipeline of web scraping (scraper module), relevance checking (releveance module), and database storage (storage module). It is designed to scrape relevant content from the web, filter it for relevance, and store query results in a Postgres database and vector db.

web_scraping/scraping.py

The **WebScraper** class handles both searching for links based on a query and scraping the content from those links.

Currently the news type identifier, scraping from specific sites is in process.

All the site specific scripts are present at scraping_utils.py

relevance_check/relevance_model.py

The **RelevanceModel** class performs the relevance checking between user queries and scraped articles by embedding text, chunking content, and storing

relevant results in a vector database (ChromaDB). It also handles database interactions for tracking embeddings and query results.

The embedding model used here is **bge-m3**.

Remember to change the Chroma Persistent client path

storage/database.py

The **PostgresDB** class handles the storage of query data, embeddings, chunk relations, and summaries. This class establishes a connection to the PostgreSQL database and allows for seamless integration with the relevance model by storing necessary information for future reference and analysis.

summarization/summarizer.py

The **Summarizer** class generates concise summaries of given text chunks.

The text generation model used here is **orca-mini:7b-q2_K**.

Dashboard

It is a streamlit applitcation which shows certain graphs about queries, chunks and query-chunk relation

Run the app and access the Swagger UI

Run the command in the terminal: `python main.py` and access the swagger at **http://127.0.0.1:8000/docs**