Title: Global AI Colab For Good

Member:
Rama Edlabadkar (ramasandeepedlabadkar@g.harvard.edu),
Shanzeh Batool (shanzehbatool@g.harvard.edu),
Labdhi Gandhi (labdhigandhi@g.harvard.edu),
Hinal Jajal (hjajal@g.harvard.edu)

Project: The scope of this project is to build a global platform that links AI research groups with organizations aiming to solve social issues using AI. The platform will have a search interface for organizations to look for AI research papers relevant to their social cause. A dashboard will provide a curated list of relevant research to the user prompt, the research groups, and how the research work relates to the user's problem prompt. The platform will be designed to support a growing number of research groups and global organizations. We process a large corpus of AI research papers & social issue descriptions and train LLMs for information retrieval and matching between research and real-world problems.

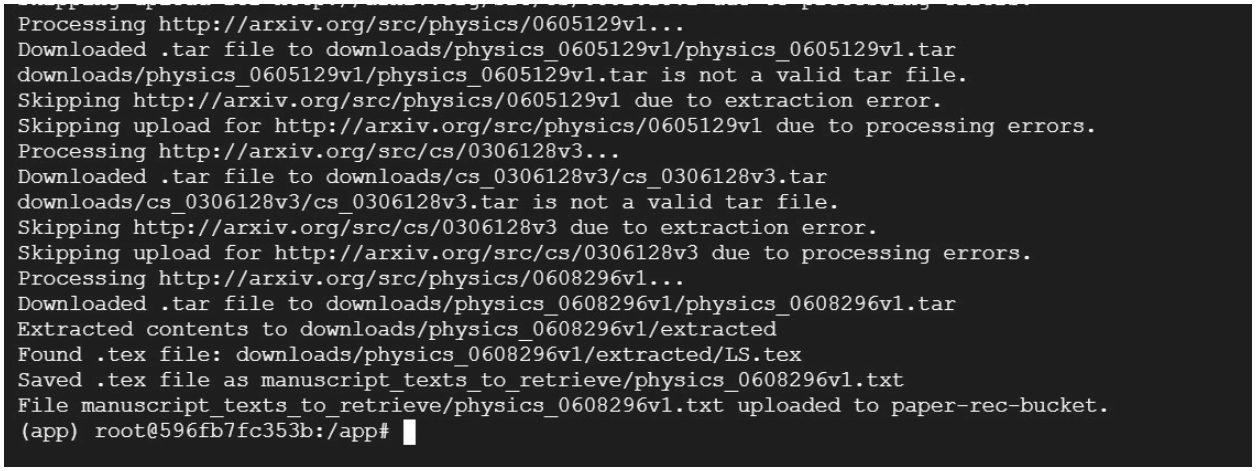
Milestone 2:

DataSet:

We utilized textual data fetched from social impact related papers obtained using the ArXiv Api, then embedded it using the Hugging Face library and stored in the vector database.

Virtual Environment Setup Process:

We use Docker containers to separate tasks like data scraping, preprocessing, embedding generation, and user prompt processing via the RAG model. This setup allows for easy management and scalability of different services. Each container runs specific processes, which can be updated, modified, or scaled independently based on requirements. We have set up the virtual environment using Docker to support containerized components. This will ensure that all project elements, from data scraping to large language model (LLM) processing, remain isolated and easy to integrate.



Purpose of different containers used (for data scraping, data preprocessing, RAG model) and instructions for running them:

This milestone covers the development and setup of a virtual environment using containers for different stages of the project, including data scraping, data preprocessing, and running the Retrieval-Augmented Generation (RAG) model. Each container is designed to handle specific tasks efficiently, ensuring modularity, scalability and ease of deployment.

1. **Container for data scraping/retrieval:**

Purpose:

- This container handles getting academic research papers from various sources like ArXiv, IEEE and Google Scholar. It retrieves AI-related research papers based on predefined categories or search queries provided.
- It also scrapes and processes social issue descriptions from non-profit organizations and research databases.
- Currently we are only extracting the social impact related research papers from ArXiv for the first baseline iteration.

Key Components:

- Libraries/Frameworks: Requests for API-based data retrieval (ArXiv API), Python's BeautifulSoup, Selenium or Scrapy for scraping.
- Tasks performed:
    - Initiate api calls and web scraping based on research domain keywords (e.g., AI and social impact).
    - Store the scraped data in a raw format, ready for preprocessing.
    - Handle rate limiting, retries and error handling for robust scraping.

Instructions for Running:

- docker build -t retrieve_papers .
- docker run --rm -ti -v "$(pwd)":/app retrieve_papers

2. **Container for data embedding, storing and other preprocessing:**

Purpose:

- This container is responsible for processing the collected data and converting text into embeddings that can be used for matching research papers to social problems.
- It also handles tasks like text cleaning, tokenization and storing embeddings in a vector database.

Key Components:

- Libraries/Frameworks: all-MiniLM-L6-v2 embedding from sentence-transformers HuggingFace library, ChromaDB for vector storage, pandas and NumPy for data manipulation.
- Tasks performed:
    - Preprocess raw text by removing irrelevant information, normalizing and cleaning.
    - Generate embeddings using all-MiniLM-L6-v2 embedding from sentence-transformers HuggingFace library, since it is lightweight and scores well on known benchmarks.
    - Store the embeddings in a vector database for later retrieval.

Instructions for Running:

- docker build -t embed_papers .
- docker run --rm -ti -v "$(pwd)":/app embed_papers

3. **Container for user prompt processing through a RAG model:**

Purpose:

- This container manages the retrieval of relevant research papers and generates responses for user queries using a Retrieval-Augmented Generation (RAG) model.
- It integrates the stored embeddings and research papers, processes user prompts, retrieves relevant information and generates human-readable output for users.

Key Components:

- Libraries/Frameworks: LangChain for RAG models, HuggingFace for embedding search, Flask/FastAPI for serving user prompts.
- Tasks performed:
    - Accept user queries, process them and retrieve relevant research papers.
    - Use the RAG model to combine the retrieved documents with user queries and generate a coherent, well-structured response.
    - Provide a dashboard for displaying relevant research to the user prompt, the research groups, and how the research work relates to the user's problem prompt. Currently we do not have a front-end dashboard but we plan to do this in the coming milestones.

Orchestration Using Docker-Compose

Since these containers need to work together in sync, we created a docker-compose.sh file to orchestrate them.

Shell Script: A shell script is included to manage the startup of all containers sequentially.

Versioned Data Strategy (Planned, not yet done):

To ensure reproducibility and portability of data pipelines, we will implement a data versioning strategy using DVC (Data Version Control). This will allow us to track the data from different versions of research papers and social problems, ensuring consistency when embedding and storing data and fine tuning models.

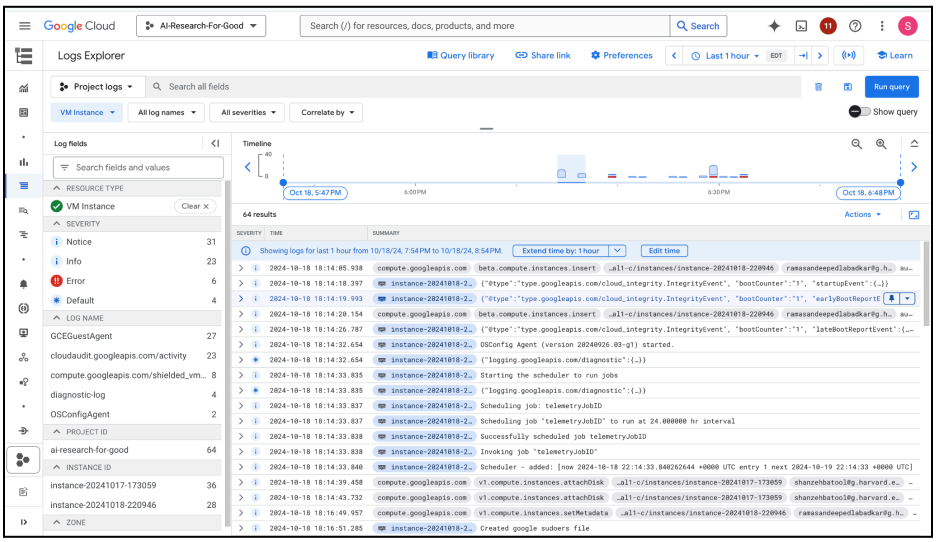
- **Strategy:** We will choose DVC for its seamless integration with Git, enabling tracking of data files alongside model code. This also provides an efficient way to manage large datasets. The versioned data pipeline will be containerized, ensuring easy integration into any environment.
- **Version Control History:** We will track dataset versions, commits and logs to ensure the exact same datasets can be reproduced at any point.

Utilizing LLM (Large Language Model) :

We utilize LLMs in our RAG model to match AI research with societal issues. To do so, we implemented a workflow that chunked the research papers and stored the vectors in a vector database. When a user enters a prompt, we retrieve relevant research papers and generate responses for user queries using the RAG model.

1. **RAG Pipeline:**
    - **Data Chunking:** Research papers are chunked into manageable text blocks to ensure efficient retrieval.
    - **Vector Database Integration:** We used ChromaDB to store the chunked research data for quick retrieval.
    - **Fine-tuning:** We will fine-tune our vector search/retrieval strategy by creating a synthetic dataset of user queries and then for each query identify and rate top relevant & top irrelevant papers. This synthetic dataset will then be utilized for fine-tuning.

2. **Experiment details/Logs:**



Mock-up of the Application:

We created a wireframe of the final platform, integrating the search interface, dashboard and backend components for matching research papers to social issues.

- **Prototype:** The user interface will allow organizations to input prompts and receive relevant research papers / suggestions. The dashboard shows research groups working on similar problems and provides detailed insights.
- **Wireframe:** The wireframe below illustrates how the UI looks. Through this the user's prompt can interact with back-end components, displaying user prompts, search results and research matches.