Page Recommender Project

Overview:

This project is a Website Similarity Recommender System.
It extracts website titles from WARC files (web archives) from Common Crawl, computes cosine similarity between them, stores everything in Neo4j, and allows users to search for websites and get recommendations through a Flask web app with a Bootstrap frontend.

Backend:

Language & Tools: Python, Flask, Neo4j AuraDB, Warcio, BeautifulSoup, scikit-learn

Components

1. **Main Data Loader (main.py)**
    - Processes WARC files to extract websites
    - Parses .warc.gz web archive files
    - Extracts:
        - Website URL
        - Page title (from <title> or meta tags)
        - Timestamp
    - Uses embedding_cosine_similarity to compute similarity between titles
    - Uploads websites as nodes (:Website {url, title, timestamp}) to Neo4j
    - Uploads relationships (:SIMILAR_TO {score}) representing similarity scores

2. **Cosine Similarity (cosine_similarity_file.py)**
    - Converts titles into TF-IDF vectors
    - Computes cosine similarity matrix
    - Filters similarities between 0.4 < score < 1
    - Returns pairs of similar websites with scores

3. **Neo4j Handler (neo4j_handler.py)**
    - Manages all database interactions
    - Functions:
        - save_websites → adds website nodes
        - save_similarities → creates bidirectional similarity relationships
        - exists_website → checks if a website exists
        - recommend_similar → fetches top-K (in this case k=5) most similar websites
        - get_all_websites → retrieves all stored websites

4. **Flask Application (app.py)**
    - Provides the backend web server
    - Workflow:
        - User enters URL → app fetches title → checks Neo4j → computes similarities if new → queries Neo4j for recommendations → sends results to frontend template

5. **Data Input & Graph Statistics**
    - Input File: WARC file (web archive).
    - File Sizes: 50 MB and 32 MB.
    - Processed Output in Neo4j:
        - Nodes (Websites): 4,484
        - Relationships (SIMILAR_TO): 32,798

Frontend:

Language & Tools: HTML, Bootstrap, Flask Jinja templates

Features

- Input form for user to enter a website URL
- Displays current website and recommendations
- Shows each recommendation with:
    - URL
    - Similarity score
    - Embedded preview (<iframe>)
    - "Open in New Tab" button that opens current URL

File: frontend.html

- Built with Bootstrap 5 for styling
- Responsive card layout for recommendations
- Uses Flask Jinja ({% ... %}) for dynamic rendering

Workflow:

1. **Data Preparation**
    - WARC files are processed with main.py
    - Websites + similarity scores are uploaded to Neo4j

2. **User Search**
    - User enters URL in frontend form
    - Flask (app.py) checks Neo4j for the website
    - If missing → fetch title + compute similarity + update Neo4j

3. **Recommendation Query**
    - Flask queries Neo4j for top-K similar websites
    - Results returned to frontend

4. **Frontend Display**
    - Recommendations shown as cards with URL, similarity score, and embedded preview