Enhanced ExpertSearch System Project Report

CS 410 - Text Information System Course Project

This is the project progress report for the CS410 Fall 2023 semester. The team chose the **System Extension** topic to improve the existing **ExpertSearch System.**

The **ExpertSearch System** is a web-based service that allows users to search for faculty members with the appropriate areas of expertise. Users can interact with the User Interface to input the query terms or chat with the chatbot to search for the information.

Overview of Functions

Web Crawler - We have implemented a new web crawler for the ExpertSearch System which crawls and scrapes data from faculty directory web pages. We have added additional checks to ensure that only directory web pages are crawled. We have also improved efficiency by adding checks to ensure that no page is visited twice, which improved the runtime significantly.

Data Extraction - As per plan, we now extract additional information from the faculty biography page. We have improved the scraper, by crawling much more information about each faculty member as compared to the previous faculty data that existed. In addition to research areas that were scrapped by the existing system, we have added functionality to scrape name, education, research areas and research interests, to give more detailed information about each faculty member, and thus improve accuracy of the overall system. This required text parsing and processing since the information in these pages were likely unstructured data. We scrape this faculty data and store them in individual text files, and also generate a metadata file for the data and the text files. The metadata stores information such as text file name, and faculty name, university, location, department associated with the specific text file.

Ranker - We aimed to improve the ranking function of the existing ExpertSearch System. The code uses KLDivergence as the primary ranker and defaults to BM25. We found that the two rankers give optimal performance, however, we fine tuned some parameters of the ranking functions to achieve slightly better performance than the existing system.

Chatbot - The chatbot in the ExpertSearch system offers enhanced functionalities, integrating directly with the system's expert search capabilities while retaining its ability to provide Wikipedia-based general information. This dual-function chatbot allows users to query specific areas of faculty expertise or research interests. It redirects the user to the search results based on the query. Additionally, for queries beyond UIUC professors, the chatbot fetches and presents summaries from Wikipedia. This integration of expert search assistance with general information retrieval makes the chatbot a valuable feature, significantly improving user interaction and the overall utility of the ExpertSearch system.

Implementation

This is an extension of the existing ExpertSearch system which is a web-based system to help users find the experts in specified areas of expertise. The platform is implemented in the Python Flask framework in combination with HTML, CSS, and Javascript. The modules added in this system are also implemented in their programming language.

The web crawler is implemented in python with the use of requests and beautifulsoup packages. The requests package allows for the crawler to reach each individual page that we are trying to scrape. BeautifulSoup gives the ability to interact with the different css and html elements on the page in order to scrape data. The crawler has a queue in order to determine what next page it needs to visit but also keeps track of the pages visited as well to prevent an infinite loop. Upon reaching a new page the crawler scrapes anything with an *h* tag because these words have hyperlinks to other pages. Since we are looking for a specific page that holds the directory for professors we only add pages with the following path in the url *'/about/people/department-faculty'* and *'/about/people/all-faculty/department-faculty'*. This specific condition helped increase efficiency and reduce runtime because it prevented the crawler from randomly searching everything on *'cs.illinois'* website and narrowed the search radius.

All the professors' information related to their department, research interests, research areas, and education is held on the page *'https://cs.illinois.edu/about/people/department-faculty/'*. So the crawler stops anytime a url has this full path in it and starts scraping the professors' information into a specific data structure. As mentioned, we increased the number of topics that we extract for each faculty, which resulted in significantly more comprehensive data files for each faculty member. Data extraction was also done using beautifulsoup and its methods. We also implemented three different storing mechanisms for the scraped data, including json objects, CSV files and text files, however, given the existing structure of the system, text files worked best to store the scraped data. However, just the text file by itself was not sufficient, therefore, we added functionality to generate a metadata file for each text file, for the search results to be retrieved efficiently.

In order to use the existing ranker system professor's education, research interests, and research areas are all stored in individual files. In order to use the updated dataset the .mat file the config.toml where updated to know where the dataset was. The ranker goes through the whole dataset and scores each file and then highest scores are mapped to the .mat file where we store the key information about the professor e.g their university, url link. The FacultyDataset-idx folder had to be updated also so that the binary files there could be recreated because in order to improve efficiency and not scan through the whole dataset again it stores a binary version in the FacultyDataset-idx folder. For the ranker implementation it was kept the same; however, in order to improve results we changed the hyperparameters.

The chatbot is seamlessly integrated into the existing Flask web application. The chatbot is programmed to handle POST requests with user queries. It interprets these queries to determine whether the user is seeking information from the ExpertSearch system or general knowledge from Wikipedia. For expert search-related queries, the chatbot redirects the user to the search results page which utilizes the web crawler and ranker. When a query is identified as a general information request, the chatbot uses the Wikipedia API to fetch summaries or articles. On the frontend, JavaScript along with HTML is used to create an interactive and user-friendly chat interface.

Usage

Server Side

Install the System - Firstly, we will install the project and its requirements:

- *git clone git@github.com:r-pongchairerks/CS410CourseProject-EnhancedExpertSearch.git*
- *cd src*
- *python3.9 -m pip install -r requirements.txt*

All required python packages have been listed in the requirements.txt and they must be installed before the server can be started. This project uses docker container so users need to ensure that the environment can run Docker.

Web Crawler - The web crawler is run on its own to generate all the data files and the metadata file. The webcrawler can be run as a python script:

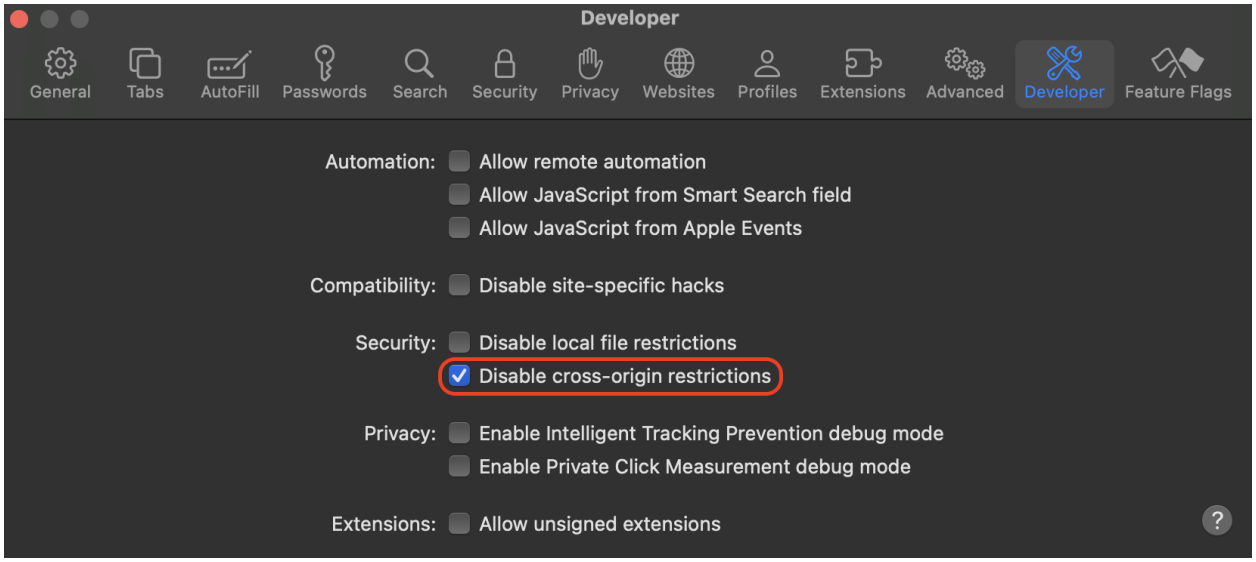
- In the src directory, run: *python3.9 webcrawler.py*
- The generated data files and metadata file will be stored in: *src/data/compiled_bios*

ExpertSearch Server Side - To start the ExpertSearch server, users require a terminal and docker-compatibility system. On Terminal, run the following command: *`gunicorn server:app -b 127.0.0.1:8095`* to start the server listening to port 8095. Localhost IP should be replaced by the actual domain name.

When starting the server, the system starts the indexing process to cache the indexing text data for fast response. The indexing data is stored in the src/FacultyDataset-idx.

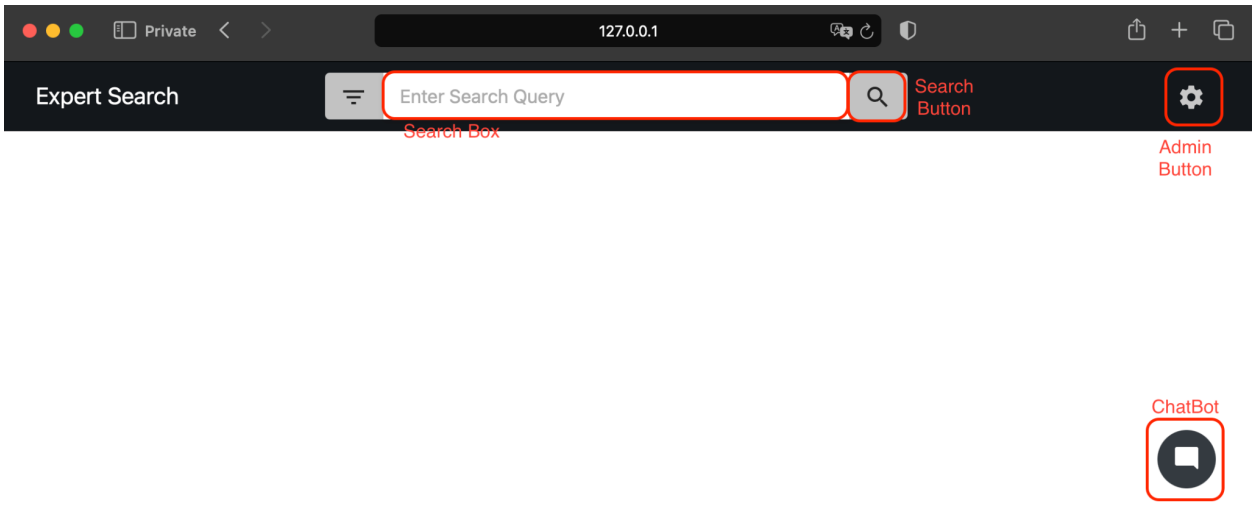
Client Side

ExpertSearch Client Side - Since this is a web-based application, users can access the system through a web browser. The system has been tested against Google Chrome, Safari, and Microsoft Edge. Currently, users can access the system by browsing to *http://localhost:8095/* . Please note that users need to ensure to disable cross-origin restrictions.



Below screenshot shows the main webpage of the ExpertSearch system. At the center of the top bar locates the “Search Box” where users can specify the search term for the research topic of interest. On the right side of the search box is the “Search Button” which triggers the search and ranking algorithm. The outcomes will be displayed in the whitespace areas based on the ranking of the ranking scores of the similarity.

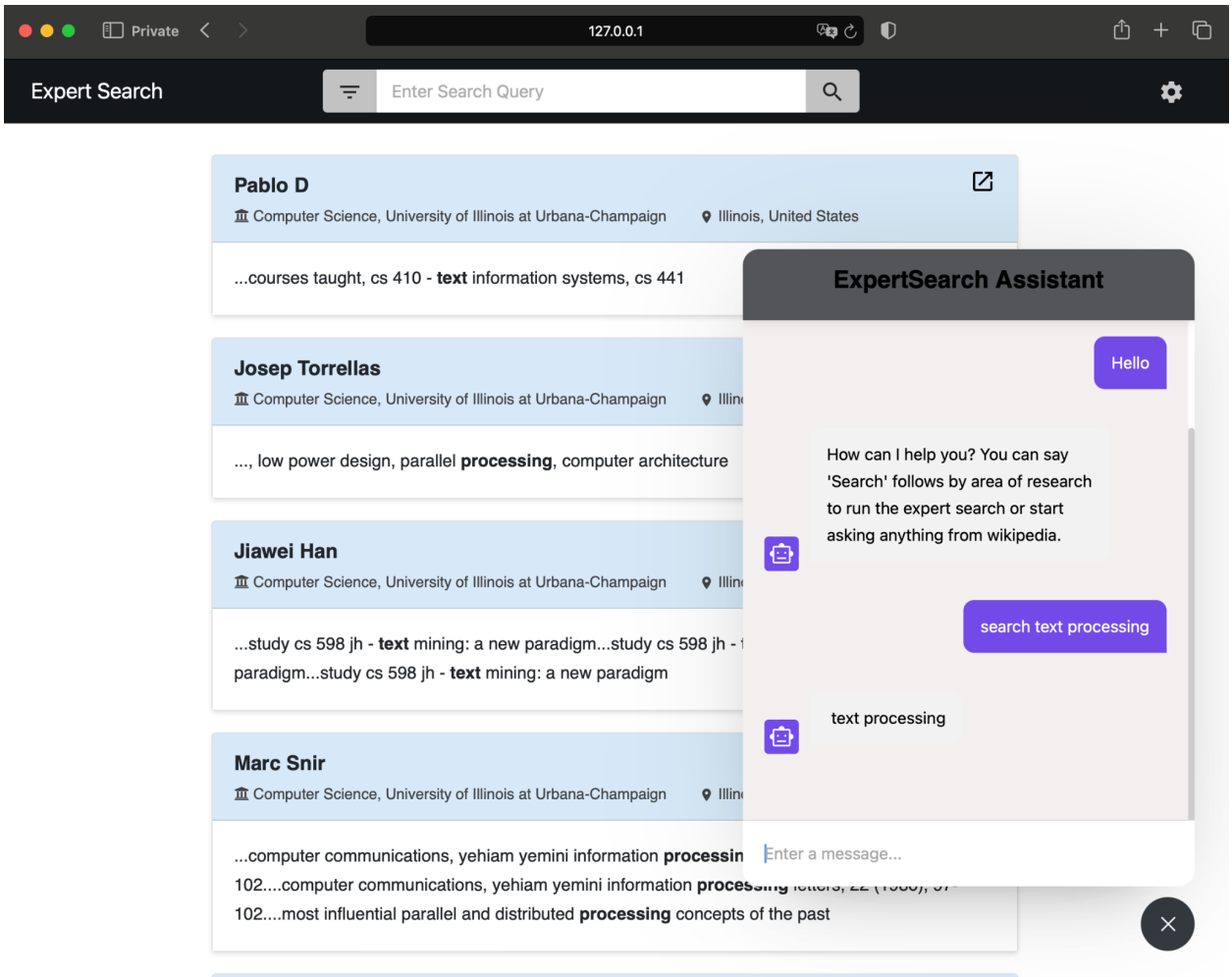
At the top right corner, you will find the “Admin Button” where you can change/modify the ranking algorithm. On the bottom right corner, you will find the “ChatBot” button that will start the ChatBot feature.



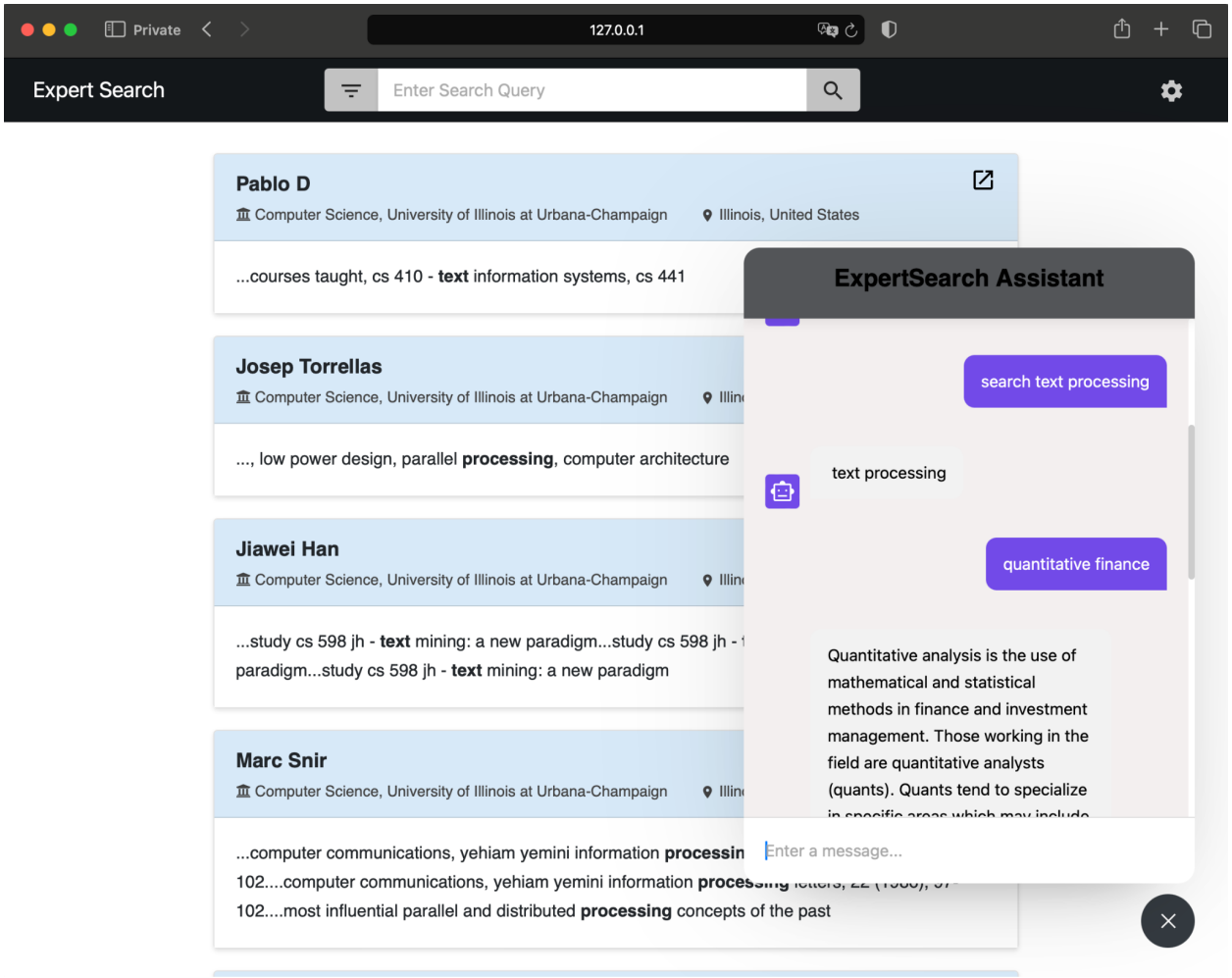
The screenshot below illustrates the examples of the search outputs from the ExpertSearch system.

Chatbot - The chatbot button is located on the bottom right corner of the page, as shown in above screenshot. Clicking on the button makes the chat box appear. Users can start off by saying “hi”, “hello”, or “what’s up” to get instructions on how to search. Queries that start with “search” followed by the research area will redirect users to the search results page with the corresponding query. Omitting “search” will make the chatbot do a general search using Wikipedia, which it then returns a summary of the information provided by Wikipedia.

The screenshot below shows an example of searching for the faculty who has research specialized in text processing. The ExpertSearch Assistant ChatBot redirects the search to the /Search page, similar to how users would input the search query in the Search Box.



The screenshot shows an example of the wikipedia search of the terms that do not start with “search”. It redirects the search to the wikipedia api and responses in the chatbot.



Contributions

Aksh Gupta (ag26) - Web Crawler, Data Extraction, Ranker, Testing and Debugging, Documentation and Presentation
Pakhi Gupta (pakhig2) - Web Crawler and Data Extraction, Testing and Debugging, Documentation and Presentation
Rasinee Pongchairerks (rasinee2) - ChatBot, Testing and Debugging, Documentation and Presentation
Woo Sung Jeon (woojeon2) - ChatBot, Documentation and Presentation
Xingsi Zhang (xingsiz2) - Data Extraction and Ranker, Documentation and Presentation