Build a Web or Social Network Search Engine

Part B: A Reddit Post Collector

Team 18 Members:
Vishal Gondi
Venkata Sai Vineeth Gudela
Alan Ngo
Rithvik Vukka
Rajeswari Pedaballi

Demo Link

1. Collaboration Details

The project was completed by the collaborative efforts of our five-member team. Vineeth and Rithvik worked on writing the backend code and setting up the indexing system. Rajeswari handled the data preprocessing, getting the Reddit data ready for indexing and creating scripts to clean the data. Alan and Vishal focused on designing the user interface and making sure everything worked well by developing test cases to check the robustness of the system. They also tested the web interface to make sure it functioned well on different platforms.

2. Detailed Overview of the System

a. Architecture

The system is structured to process, index, and provide query capabilities for Reddit data through a web interface. It integrates several components:

Web Interface: Utilizing Flask, this component handles user interactions through HTTP requests, rendering HTML content dynamically.

- ***retrieve_data.py***: Acts as the data access layer, using Apache Lucene to fetch and process data for indexing and query responses. It ensures efficient retrieval of relevant Reddit posts based on user queries.
- ***views.py***: Orchestrates data flow between the web interface and the backend logic, ensuring data is correctly passed to templates for rendering. It manages the routes and controls the display of search results to the user.
- ***static/***: Contains CSS files for web interface styling, ensuring a consistent and responsive design.
- ***templates/***: Houses HTML files (home.html, search.html) that define the structure and layout of the web pages. Using Jinja2 for dynamic content rendering, it integrates data into the HTML to display search results and other information.

Lucene Backend: Manages indexing and searching of data using Apache Lucene, a robust search library optimized for handling large datasets and complex queries.

- ***create_index.py***: Creates and configures the Lucene index, defining fields such as username, timestamp, and content, allowing for efficient search operations. By organizing data in a structured manner, it allows for efficient and rapid search operations.
- ***test.py***: Contains tests to validate the indexing and search functions, ensuring system reliability and accuracy .
- ***lucene_index/***: Directory where Lucene's index files are stored, facilitating quick data retrieval during search operations. By maintaining a well-structured index, it supports fast and accurate querying of the Reddit data.

reddit_data/: The Reddit data folder contains numerous text files, collectively exceeding 500MB, which serve as the data source for Lucene indexing. Each text file includes raw Reddit post data, such as title, body, link to the reddit post, upvotes, and comments, which are processed and indexed to facilitate efficient search and retrieval operations in the web application.

main.py: The main entry point for the Flask application, setting up the server and initializing all routes and configurations necessary for the web application. It imports and initializes the app via the create_app function from the website package. When run directly, it starts the Flask development server in debug mode, providing detailed error messages and reloading capabilities.

b. Index Structures

The Lucene index utilizes an inverted index for efficient word lookup and stored fields for retrieving complete documents. This setup supports full-text search across multiple fields such as Title, Selftext, and Comments, which are indexed to support various search functionalities.

Indexed	Stored	Field
Yes, Not Tokenized	Yes	ID
Yes, Tokenized	Yes	Title
Yes, Tokenized	Yes	Body
Yes, Not Tokenized	Yes	Upvotes
Yes, Not Tokenized	Yes	URL
Yes, Tokenized	Yes	Comments
Yes, Not Tokenized	Yes	Comments count

Analyzers play an important role in text data processing and are indexed in Lucene. The choice of analyzer can significantly impact the efficiency and accuracy of search results:

- ***StandardAnalyzer***: Used for most textual data, it includes tokenization, lowercasing, and removes stop words, making it suitable for general text fields like Title and Comments.
- ***EnglishAnalyzer***: This is specifically useful for content-rich text fields like Body, where advanced processing such as stemming and stop word removal helps in enhancing search relevancy by focusing on the root forms of words.
- ***Numeric Fields*** (Upvotes, Comments Count): These fields use Lucene's numeric indexing capabilities to handle range queries efficiently. No textual analysis is needed.

```
cs172@class-058:~/Flask-Web-App/lucene_code$ python3 create_index.py
Total time taken to index data: 15.37 seconds
cs172@class-058:~/Flask-Web-App/lucene_code$
```

The total time taken for indexing the reddit data of size (~500 MB) is 15.37 seconds.

c. Search Algorithm

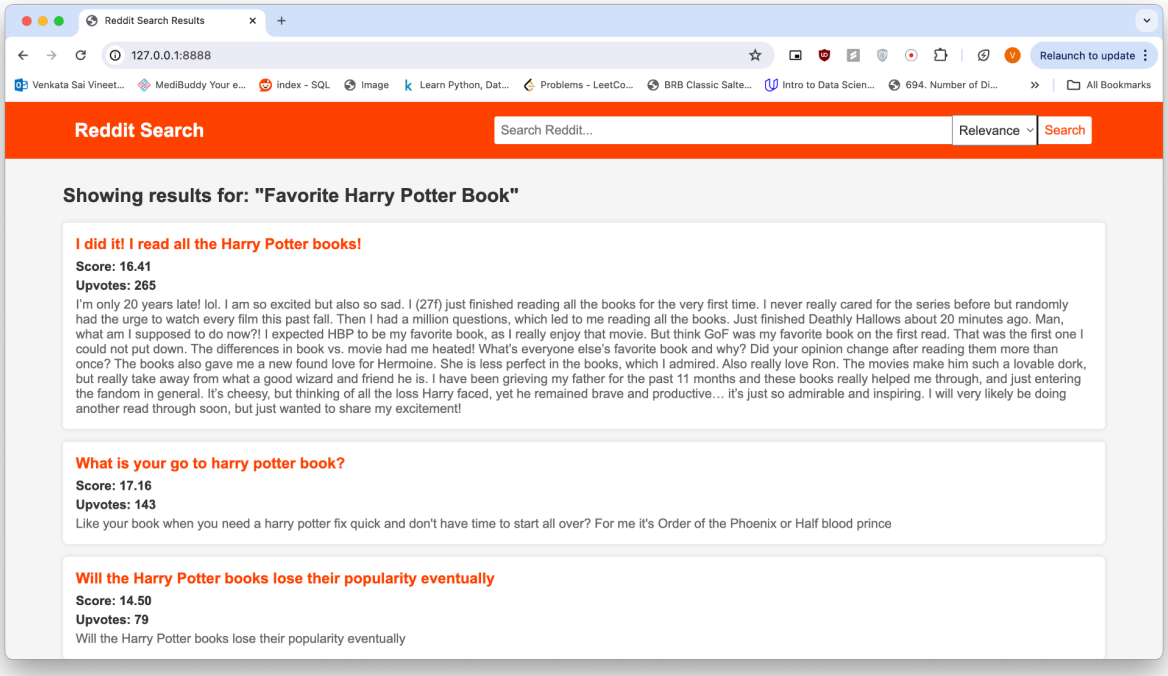
The search functionality is built upon Apache Lucene's robust search mechanisms, enhanced by customized sorting capabilities tailored for a great user experience:

- **Query Parsing**: Transforms user input into Lucene-understandable query objects, efficiently handling searches across multiple indexed fields.
- **Search Execution**: Leverages Lucene's indexing system to perform rapid and accurate searches.
- **Ranking and Sorting**: Results are initially ranked using the Lucene match score algorithm to get the relevant reddit posts across the entire dataset. The system then incorporates extra credit features by allowing dynamic sorting based on several criteria:
    - **Relevance**: Default sorting by a custom ranking function that measures how closely each post matches the search query, ensuring the most relevant content appears first. This introduces a combined weighting mechanism for match score, upvotes and comment count, enabling a customized blend of these factors to refine search results further.
    - **Upvotes**: Users can choose to sort results by the number of upvotes, prioritizing the most popular and highly endorsed content.
    - **Popularity**: Allows sorting by the number of comments, highlighting posts with the most community engagement and discussion.

d. Web Framework

We built this app in Flask and the code is written in Python, for both the backend and frontend components of our web application.

```
cs172@class-058:~/Flask-Web-App$ python3 main.py
 * Serving Flask app 'website'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:8888
 * Running on http://169.235.31.63:8888
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 133-388-392
127.0.0.1 - - [07/Jun/2024 17:39:24] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [07/Jun/2024 17:39:24] "GET /static/styles.css HTTP/1.1" 304 -
127.0.0.1 - - [07/Jun/2024 17:39:36] "POST / HTTP/1.1" 200 -
127.0.0.1 - - [07/Jun/2024 17:39:36] "GET /static/styles_search.css HTTP/1.1" 304 -
```



3. Limitations of the System

- **Scalability**: While efficient for current dataset sizes, handling extremely large datasets might require more advanced indexing algorithms.
- **Complex Queries**: Advanced search features like natural language processing or semantic search are not supported, which might limit the depth of query understanding.
- **Maintenance and Upgrades**: Regular updates are needed to ensure the new data as the current data can search only from the existing database.

4. Instructions on how to deploy the system.

The following *indexer.sh* can be used to install all the required dependencies (assuming ***pylucene*** is already installed in the environment). It will execute the required python scripts and then add the flask app variable to the environment and start the flask server.

```
#!/bin/bash

output_dir=$1

if [ -z "$output_dir" ]; then
    echo "Usage: $0 "
    exit 1
fi

sudo apt update
sudo apt install -y default-jdk
which python3
if [ $? -ne 0 ]; then
    sudo apt install -y python3 python3-pip
fi

pip3 install flask
mkdir -p "$output_dir"
python3 lucene_code/create_index.py "$output_dir"

export FLASK_APP=main.py
export FLASK_ENV=development
flask run

```

Usage:
Run the script by specifying the output directory for the Lucene indices:
chmod +x indexer.sh
./indexer.sh <output directory>

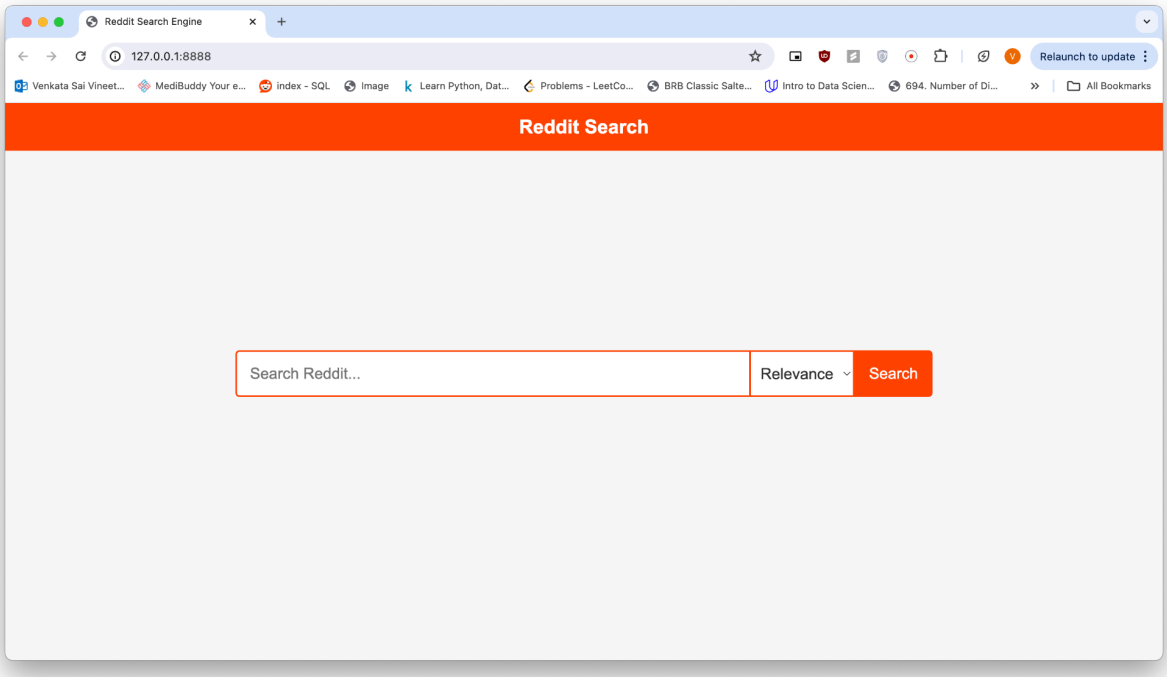
5. A web-application that can be deployed to a web server

Deploying a Flask application to a production web server requires using a WSGI (Web Server Gateway Interface) adapter as they are primarily a servlet container for Java applications. Ensure your Flask application is production-ready. This includes setting environment variables, finalizing configurations, and removing any development settings.

Once we ensure that the flask application is production ready we can use Jython, which allows running Python on the Java platform. We need to convert our Flask application to run on Jython by ensuring all your Python code and dependencies are compatible. We can then Package our Jython-based Flask app into a WAR file using Jython's tools. Copy the generated WAR file to the webapps directory of the web server installation. Restart the server to deploy the application.It will automatically deploy the WAR file upon startup and make your application accessible via the configured context path.

6. Screenshots showing the system in action.

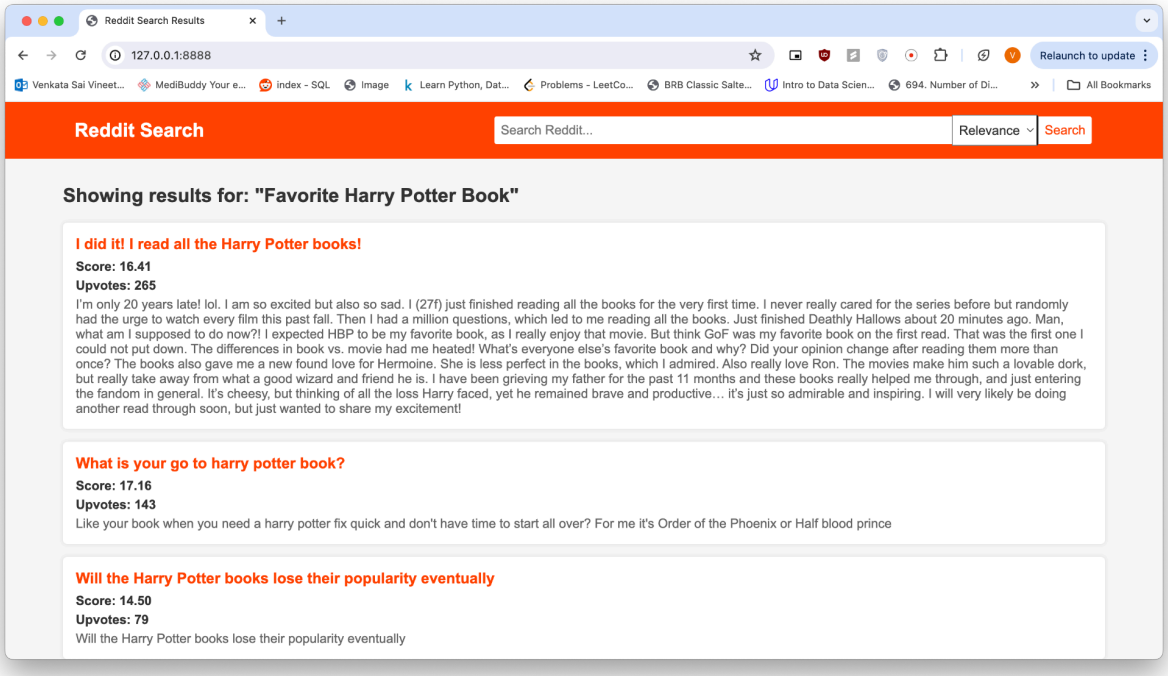
a. The homepage of our Flask web application, a Reddit search engine, contains a search bar where users can enter queries to find Reddit posts. Adjacent to the search bar, a drop-down menu is given which allows users to sort results by upvotes, relevance, or popularity.



b. The Upvotes option sorts Reddit posts based on the number of upvotes the post has, with the most upvoted posts appearing first. The Relevance option uses a custom ranking function which we have created to sort posts by how closely they match the search keywords. The Popularity option sorts posts by the number of comments they have, highlighting the most discussed content. This gives flexibility for the users to tailor their search results to their specific interests and needs.

c. The results page of our Reddit search engine displays the top 20 matching Reddit posts based on the user's query. Each post is presented with its title, which is a clickable link redirecting users to the original Reddit post. Alongside the title, the body of the post is shown to provide context. Additionally, each post includes a match score, derived from Lucene, to indicate its relevance to

the query, and the number of upvotes to highlight its popularity. This will ensure the users can quickly assess and navigate to the most relevant and popular posts related to their search.



Extra Credit

Made a sort mechanism which allows to sort the posts by number of upvotes, top comments and relevance which is a combination of match score, upvotes and comment count with a customized weights - $match\_score + 0.15 \times n_{upvotes} + 0.05 \times n_{comments}$. Relevance of each sorting mechanism is explained in the document in the ***Search Algorithm*** section.