Jaume Camps Romaguera - NIA 252021
Arol Garcia Rodríguez - NIA 252718
Albert Garrell Golobardes - NIA 254635

Group 101_5

IRWA 2024 Part 4: User Interface and Web Analytics

1. Introduction

The goal of Part 4 is to create an interactive web application for the search engine developed in previous parts and to integrate analytics for understanding user behavior. The web application allows users to interact with the search engine through a user-friendly interface and provides insights into search and usage patterns via analytics dashboards.

The implementation builds upon the previous parts, leveraging the search algorithms and ranking models to offer a functional web-based search experience. The web application integrates with the search engine backend and presents valuable data through visual dashboards that monitor user behavior, document popularity, and session engagement.

In our repository GitHub Repository contains both code and documentation for this segment, tagged for tagged for IRWA-2024-part-4.

2. Key Objectives

- **Web Application Development:**
  - Search Functionality: Users can input queries and view ranked search results from the tweet corpus.
  - Interactive Search Results: The application displays tweets with relevant metadata like likes, retweets, comments, and hashtags.
  - Document Interaction: Users can click on any result to view more details and interact with the tweet content.
- **Integration of Analytics:**
  - Query and Click Logging: The application logs each user query and stores interactions such as clicks and dwell times in CSV files.
  - Session Data: The application tracks and logs session details such as IP addresses, user agents, and locations.
  - Browser Usage Analytics: The data includes statistics on which browsers and operating systems users are employing.
- **Visualization and Dashboard:**
  - User Engagement Dashboards: Administrators can monitor user engagement via visualizations of document popularity, clicks, and dwell time.
  - Real-Time Updates: Dashboards are updated dynamically to reflect ongoing user interactions and content popularity.

3. Integration of Web Application and Seacrh Engine

Architecture Overview

Our application integrates the Flask framework as the web interface with the search engine backend built in Python. The interaction between these components follows these steps:

- Search Query Submission
  - Users enter a query in the search bar, and the request is sent to the /search route (defined in web_app.py). The SearchEngine object processes the query and fetches relevant results using the search_in_corpus function (from search_engine.py).
- Displaying Search Results
  - Once the search engine processes the query, the results are passed back to the web application, where they are dynamically rendered on the results page (results.html). The application displays the top-ranked tweets, including relevant metadata such as the tweet's text, the username, and engagement statistics.
- User Click Tracking
  - When a user clicks on a result, the interaction (including the document ID, query, and dwell time) is logged via the /clicked_doc route (defined in web_app.py). This data is then stored in the click_logs.csv file by the AnalyticsData object.
- Dashboard Visualization
  - The /dashboard route (defined in web_app.py) aggregates analytics data from various CSV files and visualizes it using pie charts and tables. Data includes browser usage, operating system distributions, and document popularity based on clicks.

4. Analytics Implementation

The application stores and processes several types of analytics data to provide insights into user behavior, including search queries, click logs, HTTP request logs, and session data. All new data is appended to the csv files keeping previous data in order to be able to do some analytics on it.

Data Storage

All analytics data is stored in the analytics_data.py module and saved in a dedicated folder called data_storage/. The data is organized in CSV files:

- Search Queries (search_queries.csv)
  - Logs each search query, session ID, and the number of terms in the query.
  - Example schema:

event_id	query	session_id	num_terms	timestamp
0	problem in india 2020	644da5ec-1a01-4f54-99a9-db379aa94e21	4	2024-12-03 18:32:27
1	protests in india 2020	644da5ec-1a01-4f54-99a9-db379aa94e21	4	2024-12-03 19:22:15
2	protests in india 2021	644da5ec-1a01-4f54-99a9-db379aa94e21	4	2024-12-03 19:29:52
3	protests in india 2021	644da5ec-1a01-4f54-99a9-db379aa94e21	4	2024-12-03 19:30:26
4	protests in india 2021	644da5ec-1a01-4f54-99a9-db379aa94e21	4	2024-12-03 19:31:01
5	protests in india 2021	644da5ec-1a01-4f54-99a9-db379aa94e21	4	2024-12-03 19:31:07
6	farmers in india	e877167e-5a8a-43dc-a4b4-72d526d9afc6	3	2024-12-04 00:40:48

- Click Logs (click_logs.csv)
  - Records click events, including the associated query, document ID, and dwell time.
  - The dwell_time is computed in the following way: we initialize a timer when the user clicks on the ‘Full tweet’ button and we stop it when the users clicks on the ‘Close’ button.
  - Example schema:

event_id	query	doc_id	dwell_time
0	protests in india 2020	doc_5490	2
1	protests in india 2020	doc_5490	2
2	protests in india 2020	doc_5485	2
3	protests in india 2020	doc_5485	3
4	protests in india 2020	doc_5485	3
5	protests in india 2020	doc_16501	5
6	support farmers please	doc_838	10
7	support farmers please	doc_11176	23
8	support farmers please	doc_4674	1

- HTTP Logs (http_logs.csv)
  - Captures HTTP request details such as IP address, request method, and response status.
  - Example schema:

event_id	ip	http_method	requested_url	http_version	response_status	timestamp
0	127.0.0.1	GET	http://localhost:8088/	HTTP/1.1	200	2024-11-30
1	127.0.0.1	GET	http://localhost:8088/static/styles/custom.css	HTTP/1.1	200	2024-11-30
2	127.0.0.1	GET	http://localhost:8088/static/styles/bootstrap.min.css	HTTP/1.1	200	2024-11-30
3	127.0.0.1	GET	http://localhost:8088/static/logo.png	HTTP/1.1	200	2024-11-30
4	127.0.0.1	POST	http://localhost:8088/search	HTTP/1.1	200	2024-11-30
5	127.0.0.1	GET	http://localhost:8088/static/styles/custom.css	HTTP/1.1	304	2024-11-30
6	127.0.0.1	GET	http://localhost:8088/static/styles/bootstrap.min.css	HTTP/1.1	304	2024-11-30
7	127.0.0.1	GET	http://localhost:8088/static/logo.png	HTTP/1.1	304	2024-11-30
8	127.0.0.1	GET	http://localhost:8088/	HTTP/1.1	200	2024-12-02
9	127.0.0.1	GET	http://localhost:8088/static/styles/custom.css	HTTP/1.1	200	2024-12-02
10	127.0.0.1	GET	http://localhost:8088/static/logo.png	HTTP/1.1	200	2024-12-02
11	127.0.0.1	GET	http://localhost:8088/static/styles/bootstrap.min.css	HTTP/1.1	200	2024-12-02

- Session Data (session_data.csv)
    - Tracks user session information, including IP, OS, browser, and location.
    - Example schema:

event_id	session_id	IP	OS	Browser	Country	City	timestamp_init
0	09cc50d3-511d-44c5-9dcc-7f8b831ae491	127.0.0.1	Mac OS	Safari	Spain	Vacarisses	2024-11-30 12:06:07
1	09cc50d3-511d-44c5-9dcc-7f8b831ae491	127.0.0.1	Mac OS	Safari	Spain	Vacarisses	2024-11-30 12:06:48
2	09cc50d3-511d-44c5-9dcc-7f8b831ae491	127.0.0.1	Mac OS	Safari	Spain	Vacarisses	2024-11-30 12:12:12
3	09cc50d3-511d-44c5-9dcc-7f8b831ae491	127.0.0.1	Mac OS	Safari	Spain	Vacarisses	2024-11-30 12:13:54
4	644da5ec-1a01-4f54-99a9-db379aa94e21	127.0.0.1	Mac OS	Safari	Spain	Vacarisses	2024-12-02 11:34:10
5	644da5ec-1a01-4f54-99a9-db379aa94e21	127.0.0.1	Mac OS	Safari	Spain	Vacarisses	2024-12-02 12:01:23
6	644da5ec-1a01-4f54-99a9-db379aa94e21	127.0.0.1	Mac OS	Safari	Spain	Vacarisses	2024-12-03 15:24:537
7	644da5ec-1a01-4f54-99a9-db379aa94e21	127.0.0.1	Mac OS	Safari	Spain	Vacarisses	2024-12-03 16:26:02
8	644da5ec-1a01-4f54-99a9-db379aa94e21	127.0.0.1	Mac OS	Safari	Spain	Vacarisses	2024-12-03 16:26:53
9	151bf204-a5ff-4cab-a60f-14f0b92b88bd	10.80.136.213	Mac OS	Safari	Spain	Vacarisses	2024-12-03 16:29:13

5. Example Execution

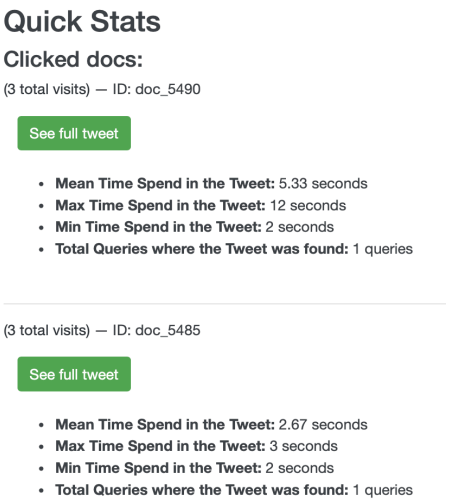
Here is an example of how the web application works in practice:

1. **Home Page**: The user lands on the homepage where they can enter a query in the search bar.
2. **Search Results**: After submitting the query, the search engine ranks the relevant tweets and displays them in the results page.
3. **Dashboard**: The admin can view the dashboard to analyze document popularity, browser usage, and other metrics.
4. **Stats**: Here the admin can view stats for document popularity, average dwell times, and the number of times a document was clicked.

6.Web Application Features

In all pages, we have added the feature to be able to navigate to all other pages.

- **Homepage (GET /)**
  - The homepage includes a search bar where users can enter their query and view results from the tweet corpus.
- **Search Functionality (POST /search)**
  - Users can enter a query in the search bar.
  - Results are displayed in order of relevance based on the search ranking.
  - Users can choose how many ranked documents to be shown.
- **Document Details (POST /clicked_doc)**
  - When a user clicks on a document, they can see the full tweet, engagement statistics (likes, comments, etc.), and a link to the original tweet.
- **Stats (GET /stats)**
  - The Statistics page shows a breakdown of document popularity, average dwell times, and the number of times a document was clicked.



- **Dashboard (GET /dashboard)**
  - The Dashboard provides insights into user behavior. We have included the following plots:
    - Ranking of visited docs. (Line Chart)
    - Browser distribution (Pie Chart)
    - OS distribution (Pie Chart)
    - City distribution (Bar Plot)
    - User interaction by hour (Line Chart)
    - Most Searched Queries (Bubble Chart)
    - Dwell-time average per document (Bar Plot)

- **Sentiment Analysis (GET /sentiment)**

- The Sentiment Analysis feature allows users to input text and get a sentiment score using the VADER sentiment analysis tool. We only have integrated the given feature in our project.

7. Conclusion

Part 4 of this project successfully integrates a Flask-based web application with the search engine developed in earlier stages. The application allows users to submit search queries, view ranked results, interact with documents, and access detailed analytics through a comprehensive dashboard. The integration of analytics enables real-time monitoring of user behavior, which can inform improvements to the search engine and user interface.

This part of the project demonstrates how web technologies and analytics can enhance the functionality and user experience of a search engine, providing valuable insights into user engagement and content popularity.