PROJECT REPORT
IRWA - PART 4 - G102-2

Clàudia Morales - u199906
Roger Sola - u199780
Queralt Zamora - u199903
G102-2

Universitat Pompeu Fabra Barcelona

1. Introduction

This document is the Part 4 Project Report, prepared by team G102-2. Here, it is described all the necessary information about the elaboration of the code for this last part, including the assumptions, methods, and algorithms employed. Each subsection presents the reasoning behind the decisions made and is classified into two main parts and a conclusion.

The search engine is designed to return relevant tweets from a corpus of documents, processing each query using both traditional and customized scoring algorithms. A key component of this part of the project is the optimization of query-document relevance, where we experimented with term frequency-inverse document frequency (TF-IDF) and cosine similarity for ranking, enhanced by the introduction of user behavior analytics to refine result rankings. These methods, combined with our pre-processing techniques, contribute to the effectiveness of the search results.

The repository for the deliveries has been shared with the labs teacher Francielle Do Nascimento, but since it is a public repository, the link to get in is the following:

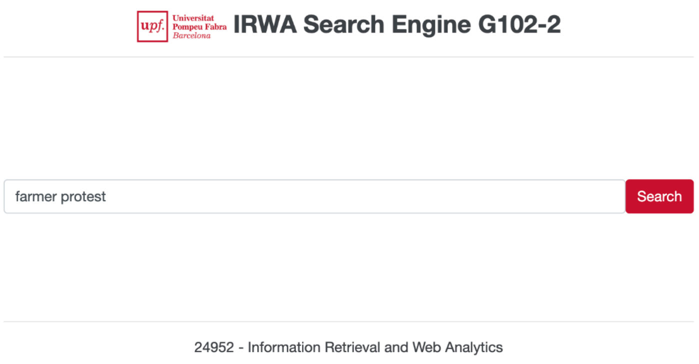
https://github.com/QuerZamora/IRWA_G102_2

2. User Interface (UI)

The User Interface (UI) is a crucial part of the project, which involves the creation of a search engine that retrieves relevant tweets based on user queries. The UI provides an interactive interface where users can input queries, view search results, and interact with detailed document information. Below is the detailed breakdown of the UI components.

2.1 Search Page

The central element of the search page is the search box. It allows users to input their query and initiate the search. The page is designed to be clean and intuitive, ensuring that users can easily interact with the search functionality.


IRWA Search Engine G102-2
farmer protest
Search
24952 - Information Retrieval and Web Analytics

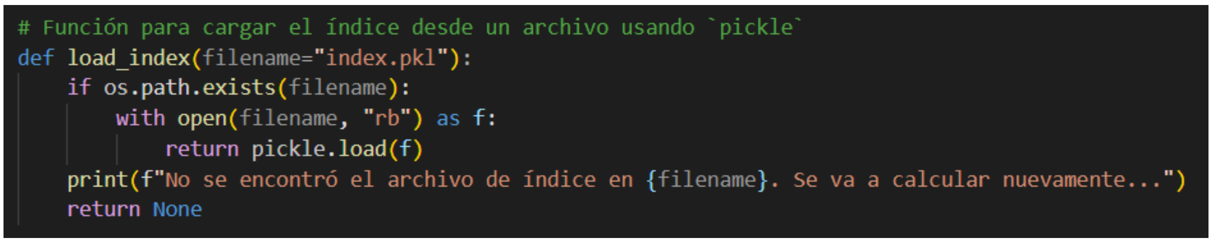
2.2 Search Action

When the user clicks the search button, the query entered in the search box is sent to the application's backend for processing. This interaction is handled through the Flask path, where the query is received and processed before passing it to the search engine to classify the documents.

2.3 Search Function in the Engine

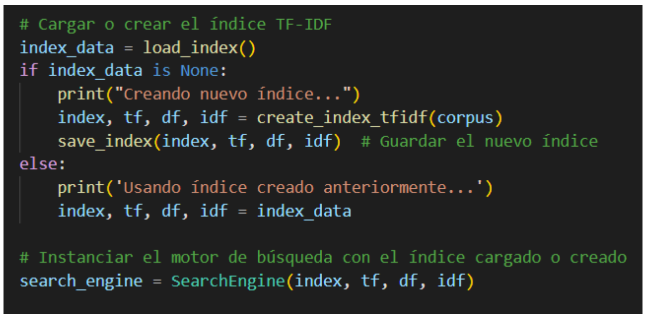
The search function is responsible for receiving the search query and other relevant parameters, such as the corpus of documents, and passing it to the search algorithms. The query string is processed by the search engine to retrieve the most relevant tweets based on the ranking mechanism used. This is done from an index.pkl that we have loaded and saved in a variable previously to speed up the process and that the search is done in a short time.

Clàudia Morales - u199906
Roger Sola - u199780
Queralt Zamora - u199903
G102-2

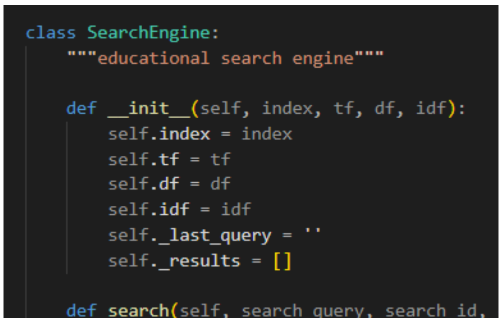


```python
# Función para cargar el índice desde un archivo usando `pickle`
def load_index(filename="index.pkl"):
    if os.path.exists(filename):
        with open(filename, "rb") as f:
            return pickle.load(f)
    print(f"No se encontró el archivo de índice en {filename}. Se va a calcular nuevamente...")
    return None
```

To work properly, we modified the Search Engine class to store index, tf, df, and idf. Thus, when loading the index, we initialize the search engine with these values as we can see in the following image:



```python
# Cargar o crear el índice TF-IDF
index_data = load_index()
if index_data is None:
    print("Creando nuevo índice...")
    index, tf, df, idf = create_index_tfidf(corpus)
    save_index(index, tf, df, idf)  # Guardar el nuevo índice
else:
    print('Usando índice creado anteriormente...')
    index, tf, df, idf = index_data

# Instanciar el motor de búsqueda con el índice cargado o creado
search_engine = SearchEngine(index, tf, df, idf)
```

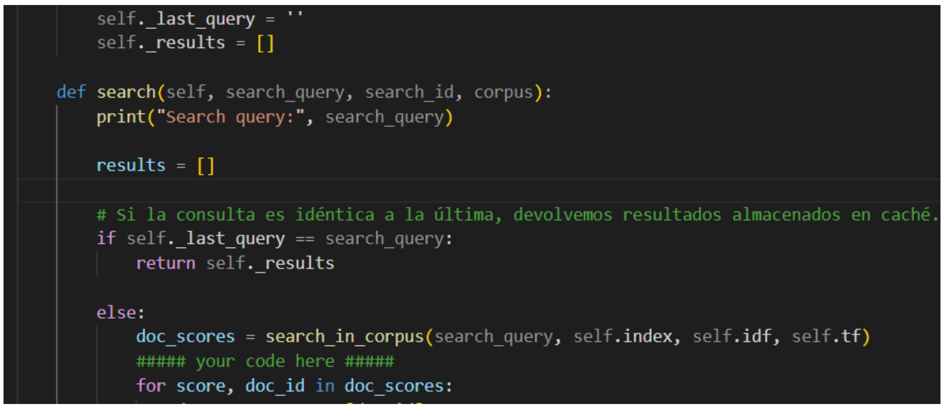


```python
class SearchEngine:
    """educational search engine"""

    def __init__(self, index, tf, df, idf):
        self.index = index
        self.tf = tf
        self.df = df
        self.idf = idf
        self._last_query = ''
        self._results = []

    def search(self, search_query, search_id,
```

2.4 Search Algorithms

To optimize search performance, the search algorithm uses various ranking methods, such as TF-IDF with cosine similarity, to rank tweets based on relevance. TF-IDF helps capture the importance of terms within the tweet and throughout the corpus, ensuring that only the most relevant documents are returned. To do this, we relied on the algorithms we did earlier in parts 1, 2, and 3.

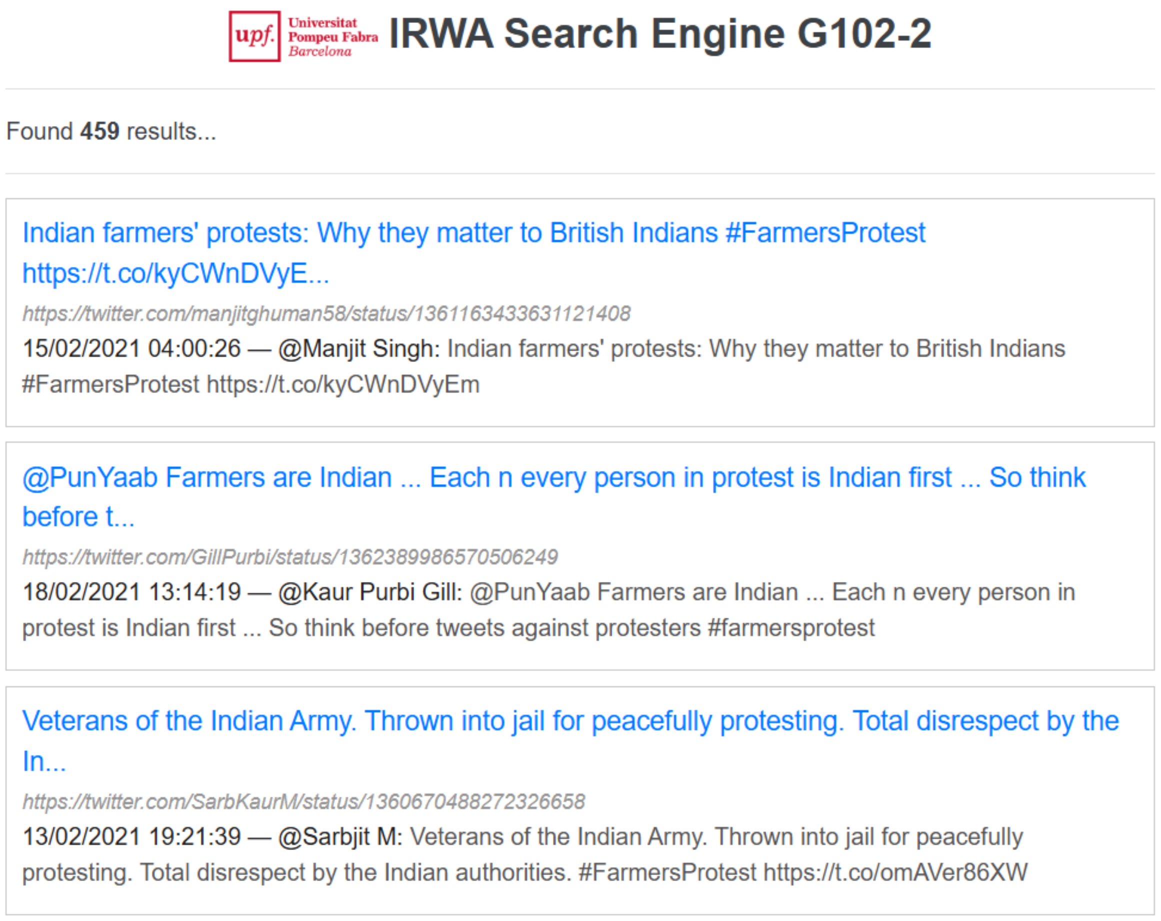
Also, to create a more efficient way to load the data, we store the data from the last query, so if the user wants to search for more information about that same query, the system will directly access the data. That is, if the user accesses the same query, the system will display the previous data, otherwise, the search engine will run.



```python
        self._last_query = ''
        self._results = []

    def search(self, search_query, search_id, corpus):
        print("Search query:", search_query)

        results = []

        # Si la consulta es idéntica a la última, devolvemos resultados almacenados en caché.
        if self._last_query == search_query:
            return self._results

        else:
            doc_scores = search_in_corpus(search_query, self.index, self.idf, self.tf)
            ##### your code here #####
            for score, doc_id in doc_scores:
```

2.5 Results Page

The results page displays the search results in a clear and structured format, showing key document information such as title, full tweet text, creation date and time and URL (linked to the tweet).



IRWA Search Engine G102-2

Found **459** results...

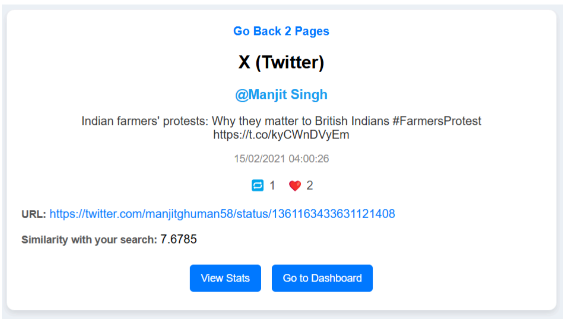
Indian farmers' protests: Why they matter to British Indians #FarmersProtest https://t.co/kyCWnDVyE...
https://twitter.com/manjitghuman58/status/1361163433631121408
15/02/2021 04:00:26 — @Manjit Singh: Indian farmers' protests: Why they matter to British Indians #FarmersProtest https://t.co/kyCWnDVyEm

@PunYaab Farmers are Indian ... Each n every person in protest is Indian first ... So think before t...
https://twitter.com/GillPurbi/status/1362389986570506249
18/02/2021 13:14:19 — @Kaur Purbi Gill: @PunYaab Farmers are Indian ... Each n every person in protest is Indian first ... So think before tweets against protesters #farmersprotest

Veterans of the Indian Army. Thrown into jail for peacefully protesting. Total disrespect by the In...
https://twitter.com/SarbKaurM/status/1360670488272326658
13/02/2021 19:21:39 — @Sarbjit M: Veterans of the Indian Army. Thrown into jail for peacefully protesting. Total disrespect by the Indian authorities. #FarmersProtest https://t.co/omAVer86XW

2.6 Document Details Page

The document details page displays the full tweet content along with its metadata, such as the creation date, the user who posted it, and the tweet's interaction statistics (likes, retweets). This page allows users to see the entire document in context, providing additional clarity for those interested in the specifics of a document.



X (Twitter)

@Manjit Singh

Indian farmers' protests: Why they matter to British Indians #FarmersProtest https://t.co/kyCWnDVyEm

15/02/2021 04:00:26

1 2

URL: https://twitter.com/manjitghuman58/status/1361163433631121408

Similarity with your search: 7.6785

Clàudia Morales - u199906
Roger Sola - u199780
Queralt Zamora - u199903
G102-2

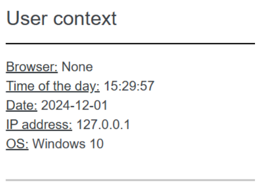
3. Web Analytics

3.1 Data Collection

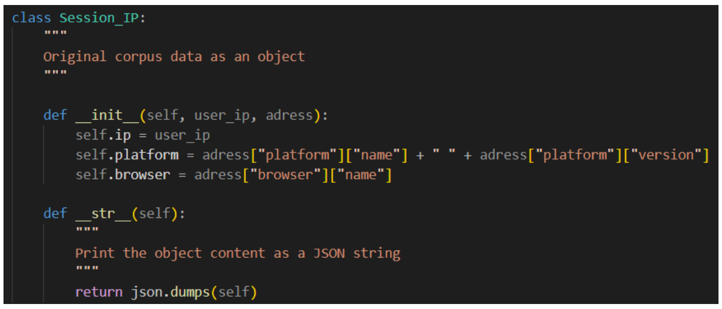
In this section, we track various aspects of user interaction with the search engine to gain insights into behavior and performance. The key data points include HTTP requests, search queries, document interactions, user context, and session data.

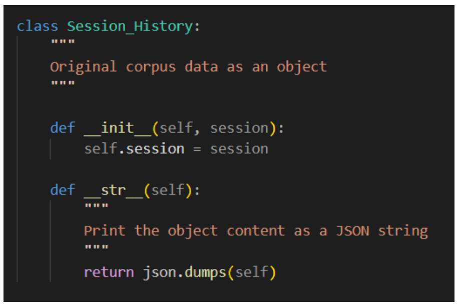
For HTTP:

We log HTTP requests, which include tracking the type of HTTP request (GET, POST), timestamp, user IP address, user-agent (browser, device), and the requested URL.


User context

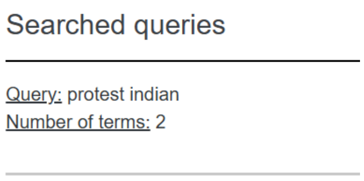
Browser: None
Time of the day: 15:29:57
Date: 2024-12-01
IP address: 127.0.0.1
OS: Windows 10


```python
class Session_IP:
    """
    Original corpus data as an object
    """

    def __init__(self, user_ip, adress):
        self.ip = user_ip
        self.platform = adress["platform"]["name"] + " " + adress["platform"]["version"]
        self.browser = adress["browser"]["name"]

    def __str__(self):
        """
        Print the object content as a JSON string
        """
        return json.dumps(self)
```


```python
class Session_History:
    """
    Original corpus data as an object
    """

    def __init__(self, session):
        self.session = session

    def __str__(self):
        """
        Print the object content as a JSON string
        """
        return json.dumps(self)
```

For Queries:

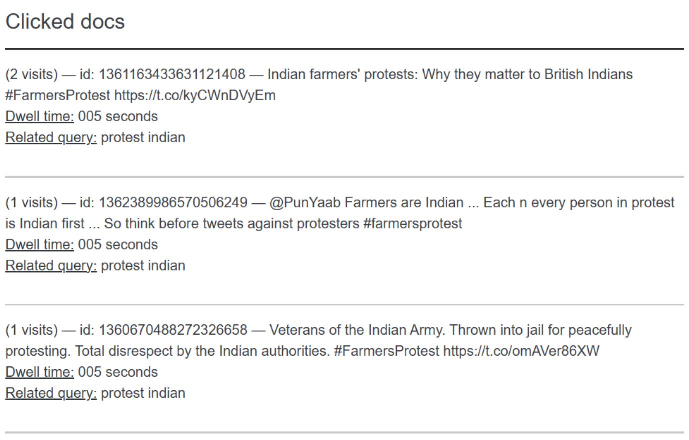
We track the queries submitted by users, including the number of terms in the query, the order of terms, and other related data (e.g., query length, specific terms used).


Searched queries

Query: protest indian
Number of terms: 2

For Results (Documents):

We track clicks on documents (tweets), record the ranking of clicked documents, and calculate the time between clicking on a result and returning to the search results page. We've done all of this with the ClickedDoc class and AnalyticsData.

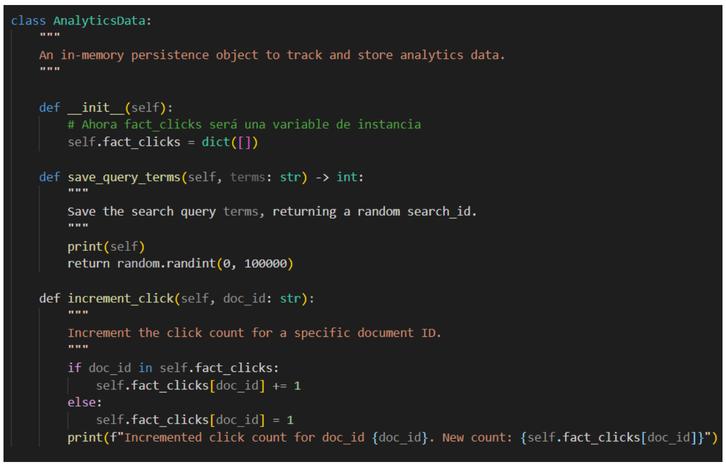

Clicked docs

(2 visits) — id: 1361163433631121408 — Indian farmers' protests: Why they matter to British Indians #FarmersProtest https://t.co/kyCWnDVyEm
Dwell time: 005 seconds
Related query: protest indian

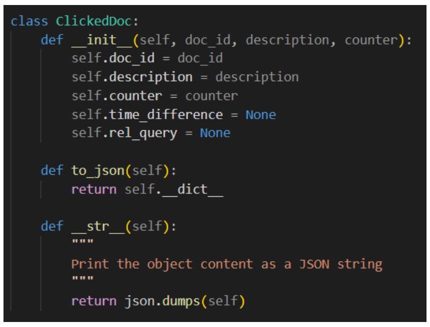
(1 visits) — id: 1362389986570506249 — @PunYaab Farmers are Indian ... Each n every person in protest is Indian first ... So think before tweets against protesters #farmersprotest
Dwell time: 005 seconds
Related query: protest indian

(1 visits) — id: 1360670488272326658 — Veterans of the Indian Army. Thrown into jail for peacefully protesting. Total disrespect by the Indian authorities. #FarmersProtest https://t.co/omAVer86XW
Dwell time: 005 seconds
Related query: protest indian

Clàudia Morales - u199906
Roger Sola - u199780
Queralt Zamora - u199903
G102-2



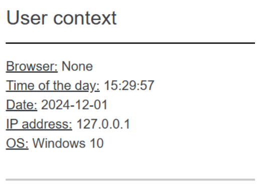
```python
class AnalyticsData:
    """
    An in-memory persistence object to track and store analytics data.
    """

    def __init__(self):
        # Ahora fact_clicks será una variable de instancia
        self.fact_clicks = dict([])

    def save_query_terms(self, terms: str) -> int:
        """
        Save the search query terms, returning a random search_id.
        """
        print(self)
        return random.randint(0, 100000)

    def increment_click(self, doc_id: str):
        """
        Increment the click count for a specific document ID.
        """
        if doc_id in self.fact_clicks:
            self.fact_clicks[doc_id] += 1
        else:
            self.fact_clicks[doc_id] = 1
        print(f"Incremented click count for doc_id {doc_id}. New count: {self.fact_clicks[doc_id]}")
```



```python
class ClickedDoc:
    def __init__(self, doc_id, description, counter):
        self.doc_id = doc_id
        self.description = description
        self.counter = counter
        self.time_difference = None
        self.rel_query = None

    def to_json(self):
        return self.__dict__

    def __str__(self):
        """
        Print the object content as a JSON string
        """
        return json.dumps(self)
```

For User Context:

We capture user context such as browser type, operating system, and device type. This is typically extracted from the User-Agent string in the HTTP request headers



User context

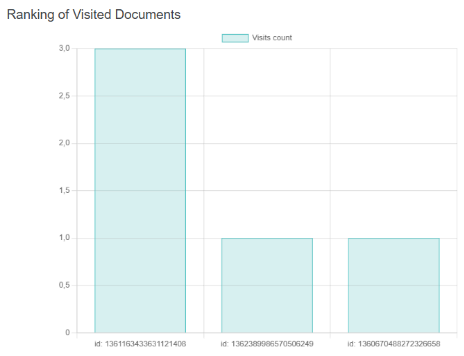
Browser: None
Time of the day: 15:29:57
Date: 2024-12-01
IP address: 127.0.0.1
OS: Windows 10

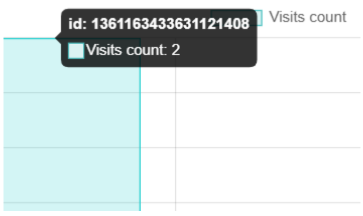
3.2 Data Storage

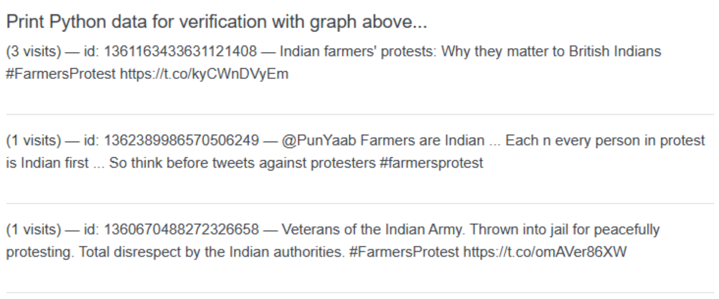
In short, as we have explained and shown above, to store the collected data, we have designed a simple but scalable data model. We have stored user context information, recorded document clicks, including query-document relationships, and recorded classification data, and logged all HTTP requests.

3.3 Analytics Dashboard

The analytics dashboard provides key metrics such as most visited documents, query frequency, and user interaction data. Display charts and reports using visual libraries. As we can see below, we have a bar chart that counts the documents visited and their id and further down we have the most specific information about the tweet and the user.







Print Python data for verification with graph above...

(3 visits) — id: 1361163433631121408 — Indian farmers' protests: Why they matter to British Indians #FarmersProtest https://t.co/kyCWnDVyEm

(1 visits) — id: 1362389986570506249 — @PunYaab Farmers are Indian ... Each n every person in protest is Indian first ... So think before tweets against protesters #farmersprotest

(1 visits) — id: 1360670488272326658 — Veterans of the Indian Army. Thrown into jail for peacefully protesting. Total disrespect by the Indian authorities. #FarmersProtest https://t.co/omAVer86XW

Clàudia Morales - u199906
Roger Sola - u199780
Queralt Zamora - u199903
G102-2

4. Conclusions

Part 4 of the project involved developing a web-based search engine with integrated analytics. The search engine retrieves and ranks relevant tweets based on user queries using traditional algorithms like TF-IDF. The user interface was designed for ease of interaction, featuring a central search box, a results page, and a detailed document page.

Web analytics were implemented to track user actions, including query input, document clicks, and session data. This data is stored in a structured database and visualized in reports to optimize the search experience.

Overall, this part of the project successfully combines search functionality with actionable analytics to improve user engagement and search performance.

Finally, with this last part 4, we are concluding the overall project for this subject. As a team, we wanted to express how much we have enjoyed seeing the project grow step by step throughout its different phases. It has been a rewarding experience to work together, and we especially want to thank the professors for their support whenever we needed it. Their guidance made it possible for us to enjoy the process while working hard on the project. This journey has been both educational and enjoyable, and we are proud of what we have achieved as a team, and ee also hope that our professors feel the same way.