

Part 4: User Interface & Web Analytics

In the following report we will go over the procedures and supporting material used for the fourth and last part of our IRWA's project. We will omit most details from the previous parts that composed our project.

Our main objective in this deliverable is to provide a web application to enter a search query, display the results obtained and usage statistics. The code uses the Flask framework, and the main displayed pages are search page, results page, document detail page, statistics page and dashboard page.

User interface

Search page: user writes a query and clicks Search button.

Results page: a list of documents related to the previous query are displayed and the user is able to click any of them. For the first part the algorithm that we have been using is TF- IDF, just like in the previous parts. In order to optimize our algorithm we have tried to retrieve the results faster and cleaner.





- **Faster:** we create the *index*, *tf* and *idf* just once (first time user is looking for a query) and store them in memory. Every other time the user searches a query, these fields are just read from json files which is much faster than calling *create_tf_idf* function.
- **Cleaner:** we create the index, tf and idf after preprocessing the documents. We have added a field to the Document object called *cleaned_description* containing the description processed with *build_terms* function (used in last practices). Thus, we are able to have both the real description for using it when printing the results, and the cleaned description, for using it in the *create_tf_idf* function. We have also used *build_terms* for preprocessing each requested query.

Document details page: Finally, we show the document detail page of the clicked document, that is a page to display the whole document's information (id, description, date, likes, retweets, hashtags and url).

Document Details

[Go Back](#) | [Go Back 2 pages](#) | [Go Back 3 pages](#) | [Go Back 4 pages](#) | [Stats](#) | [Dashboard](#)

Id: 1416433609091653633

Description: 
 COVID-19 vaccines COVID-19 vaccines in 10 countries in the rest of the  #VaccinEquity is 
to ending the pandemic, together! #WorldEmojiDay

Date: Sat Jul 17 16:24:23 +0000 2021

Likes: 3486

Retweets: 1517

Hashtags: ['VaccinEquity', 'WorldEmojiDay']

Url: <https://t.co/wVulKuROWG>

Web Analytics

The other part of the implementation is the analytics compilation of the search engine. First we have collected data for analytics and then we have created two more pages; statistics page and dashboard page.

1. For analytics we have collected the following data

Statistics for queries and words

- *fact_terms*: dictionary storing each term with the number of times it has appeared in query
- *fact_number_terms*: dictionary storing each query length
- *fact_query_times*: dictionary storing each query and the number of times it has been searched

Statistics for results (documents)

- *fact_clicks*: dictionary with the click counters: key = doc id | value = click counter
- *fact_rankings*: dictionary storing the top 10 most clicked docs with its number of clicks
- *fact_queries*: dictionary storing each document id with its list of queries related (meaning the queries typed before clicking this document)
- *fact_dwell_time*: dictionary storing the total time spent in each document

Statistics for user context

- *fact_browser*: dictionary storing browsers and the number of times users used each of them
- *fact_OS*: dictionary storing Operating Systems(OS) and the number of times users used each of them
- *fact_ip*: dictionary storing IPs and the number of times each one was used
- *fact_time*: dictionary storing the hours of the day and how many users where connected
- *fact_date*: dictionary storing dates (day/month/year) and how many users where connected each day

Statistics for sessions

- *fact_num_queries*: dictionary storing the number of searched queries for each session
- *fact_num_detail*: dictionary storing the number of visited documents for each session

2. Statistics page

In this page we display the information already collected with the following tables:

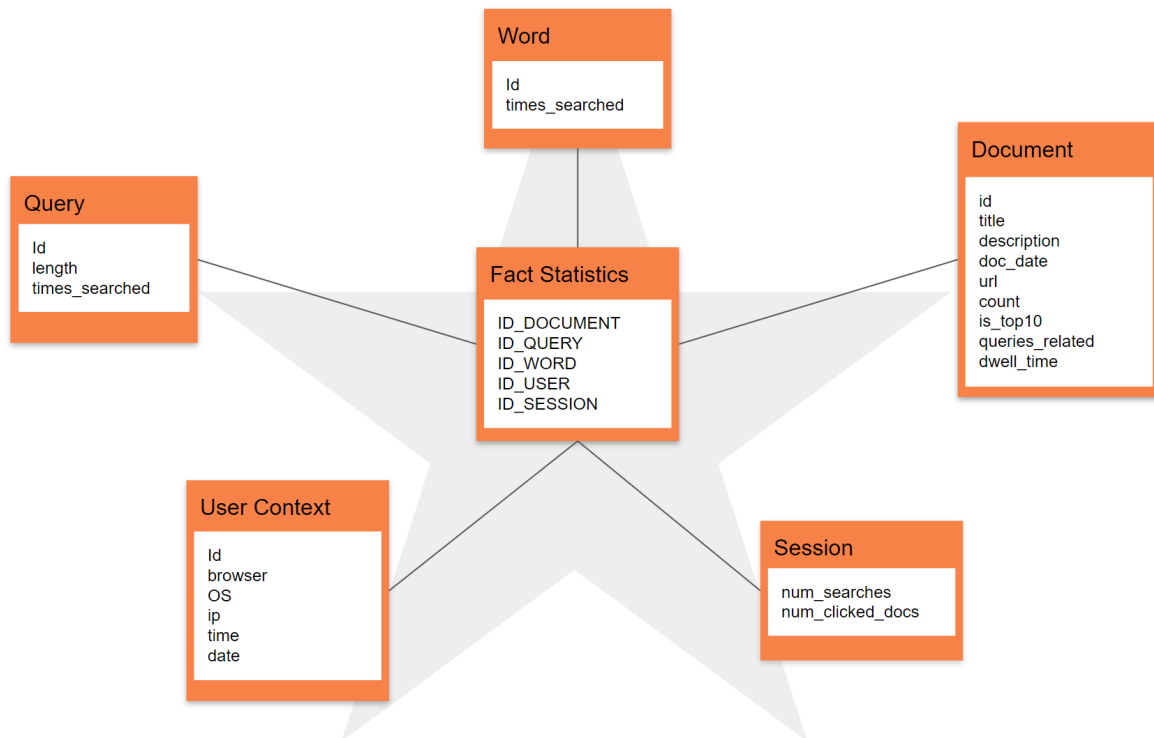
- **Clicked docs:** for each doc displays id, description, n° of visits, queries related and if is in top10 clicked
- **Requested queries:** for each query displays the query, its length and the number of times was searched
- **Sessions information:** displays browsers, OSs, IPs, times, dates

We use the StatsDocument object to print the information collected about the document. Note that we have added *is_top10*, which tells us if the document is one of the 10 most clicked documents, *queries_related*, which gives us a list of the queries used before clicking on the document, and *dwell_time*.

We have also created the StatsQuery object with all information explained in statistics for query and StatsSession object with all information explained in Statistics for user context and for sessions.

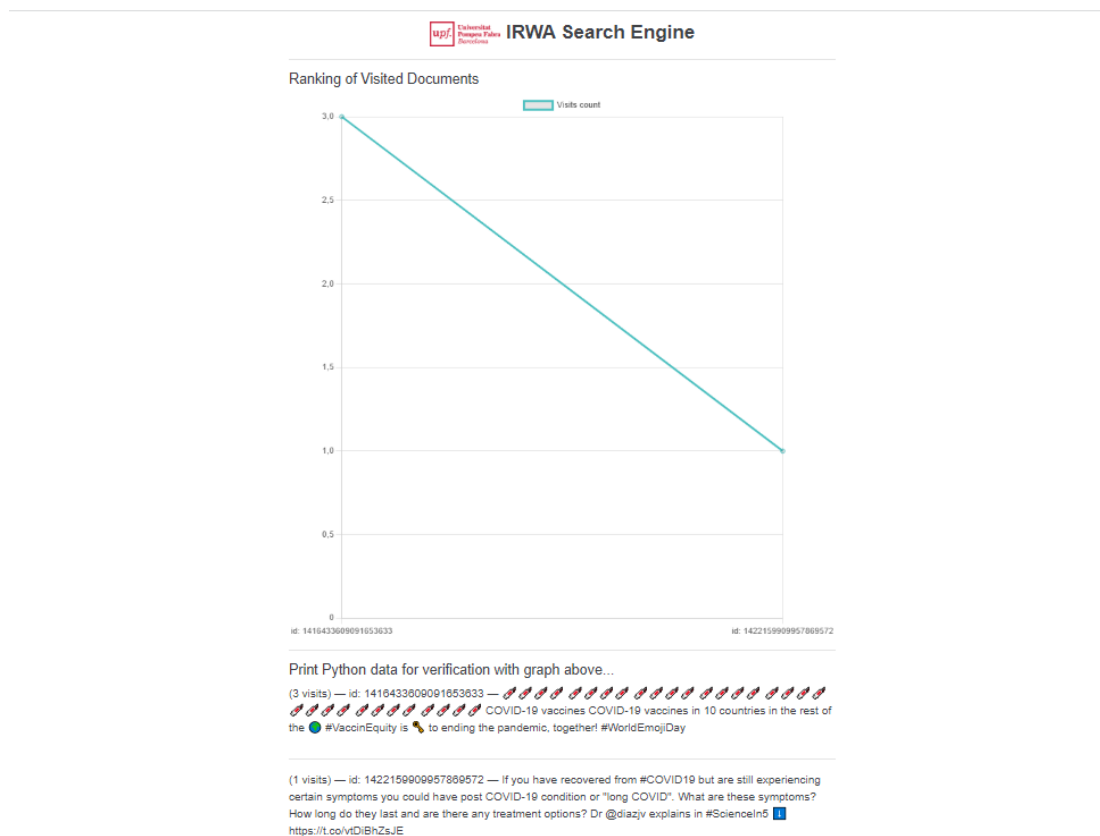
We have an example of how our statistics page looks below:

From the information we are displaying, we consider the next star schema:



3. Dashboard page

This page displays the usage statistics. Below we can see an example:



The following link will redirect you to our GitHub repository, specifically to the corresponding part. In this case part 4: <https://github.com/IRWA-2022/PART4>