

COVITTER ANALYSIS

SWE – 573 / PROJECT REPORT

MERVE MELIS MIRZA, 2019719132

TABLE OF CONTENTS

Introduction	2
Requirements	2
Functional Requirements.....	2
Non-functional Requirements.....	3
System Design	3
Mock-ups	3
System Architecture	4
Data Collection	5
Pre-Processing of Data.....	7
Data Analysis	8
Database Design	9
Scheduled Jobs	11
The Web Application	11
System Manual	12
User Manual	13
Login and Sign Up	13
“This Week” Page	15
“Custom Search” Page	20
Testing	23
Observations	26
Suggestions For Future Development	28
References	29

INTRODUCTION

Covitter Analysis is a COVID-19 analysis application on Twitter posts. It uses a stream of posts about COVID-19 collected at 10 minute intervals. On the application, analysis on sentiment, entity linking and co-occurrences are presented. The application can be accessed from <https://covitter.herokuapp.com/>. On this report, project requirements, system design, system and user manual and the test results will be presented. Also, in the final section, certain observations regarding the data and suggestions for future development are shared.

REQUIREMENTS

Functional and non-functional requirements were determined before the project started.

FUNCTIONAL REQUIREMENTS

The list of functional requirements is as follows:

- F1. Tweets related to COVID-19 should be analyzed.
- F2. User profiles should be kept.
 - F2.1 Users must login to system before accessing anything.
 - F2.2 Profiles should be password protected.
 - F2.3 A valid e-mail address must be provided.
- F3. User should be able to choose the time interval for analysis.
- F4. Application should stream data from Twitter API.
- F5. Collected data must be analyzed and results must be presented to user.
 - F5.1 Sentiment analysis is performed.
 - F5.2 Results should be positive/negative/neutral.
 - F5.3 Results should be visualized on the page.
- F6. Most common words in all tweets should be determined.
 - F6.1 Stop words must be excluded from the results.
 - F6.2 Results should be visualized on the page.
- F7. Co-occurrence graph should be created.
 - F7.1 Only entities that can be extracted from the posts should be considered.
 - F7.2 Results should be shown as a network on the page.
- F8. User information and profiles of the posters should not be disclosed on the app.

NON – FUNCTIONAL REQUIREMENTS

The list of non-functional requirements is as follows:

- NF1. The application must be a web application.
- NF2. The data source must be the Twitter API.
- NF3. GitHub must be used as the version control system.
- NF4. GitHub Wiki must be used for documentation.
- NF5. Python must be the programming language.
- NF6. Django must be used as the web framework.
- NF7. PostgreSQL must be used for relational database needs, if that type of database system is chosen.
- NF8. MongoDB must be used as the NoSQL database, for the unstructured / semi-structured data, if data would be kept in such format.
- NF9. The application should be containerized, using Docker.
- NF10. TagMe should be entity linking.

SYSTEM DESIGN

The system is designed to fulfill the requirements listed on the previous section. System design documents from several perspectives and the reasoning behind design decisions are presented in this section.

MOCK-UPS

The initial design was made using mock-ups of the application, how the main functionality would be provided. On Figure 1 and Figure 2, the initial design of user management screens are presented. They comply with the requirement F2 and its sub-requirements.

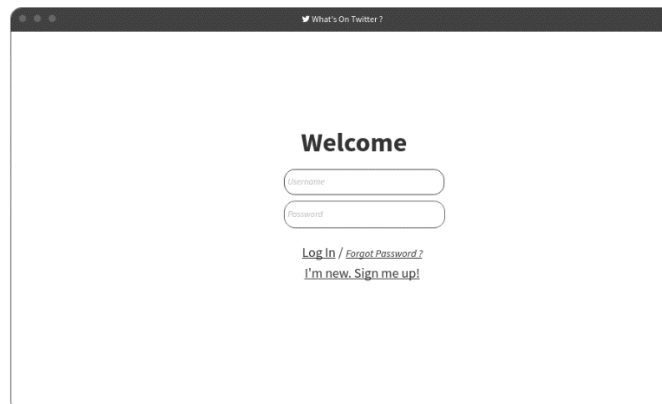


Figure 1: The mock-up of welcome page

Figure 2: The mock-up of sign up page

The mock-up of the contents of the application, the main page is shown on Figure 3. The contents are designed to comply with requirements F3, F5, F7 and F8. On the page date filter is provided for the user. Co-occurrence graph, frequently used words chart and sentiment analysis. There is also an extra feature for choosing the location the post has originated. At the end of the project not all of these components were realized but main idea could be reflected.

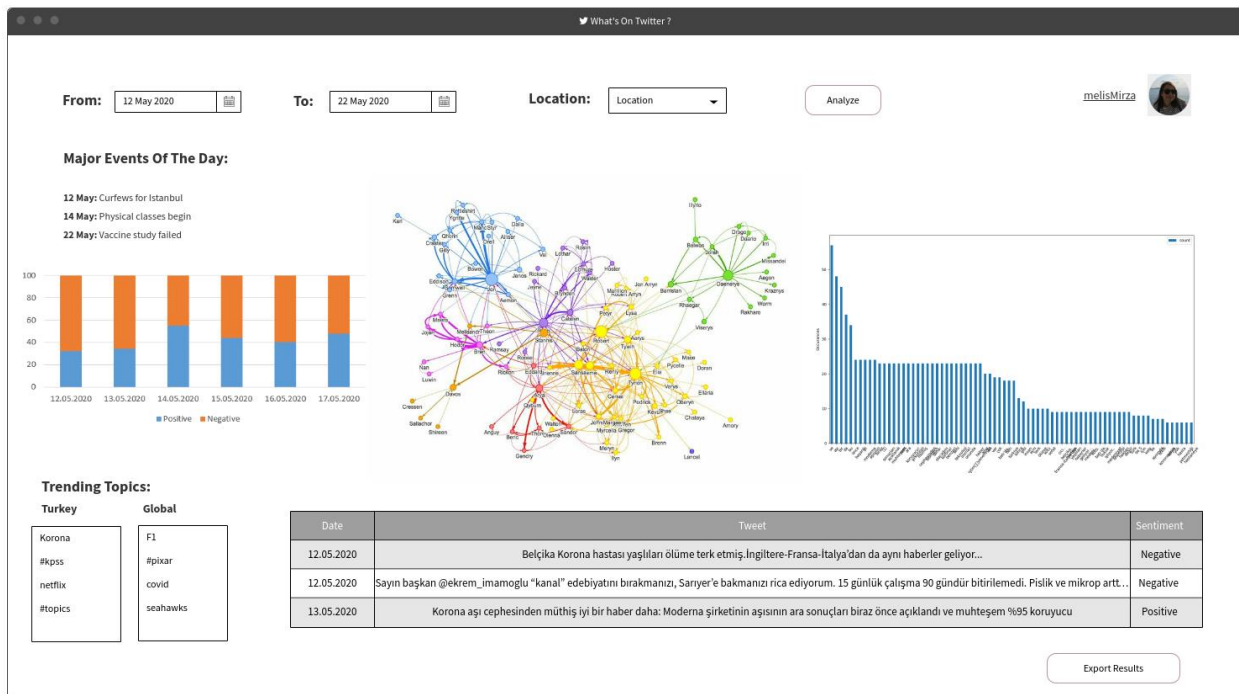


Figure 3: The mock-up of the contents of the main page of the application

SYSTEM ARCHITECTURE

There are requirements that would affect the design of system architecture, such as NF3, NF7, NF8 and NF9. Development is done on local machines and the developed code is pushed to the GitHub repository of the project (https://github.com/melisMirza/SWE573_project).

The main application servers and database are on a cloud platform. Heroku is chosen as this platform. There have been trials with Azure Platform however, there were insurmountable problems with deployment within the given completion time of the project, and also PostgreSQL server fee was quite high. Heroku was, however, suitable for a small size web application such as ours. Also, it had reasonable fees. Heroku also provided an automatic deployment to web. Overall system architecture is shown on Figure 3.

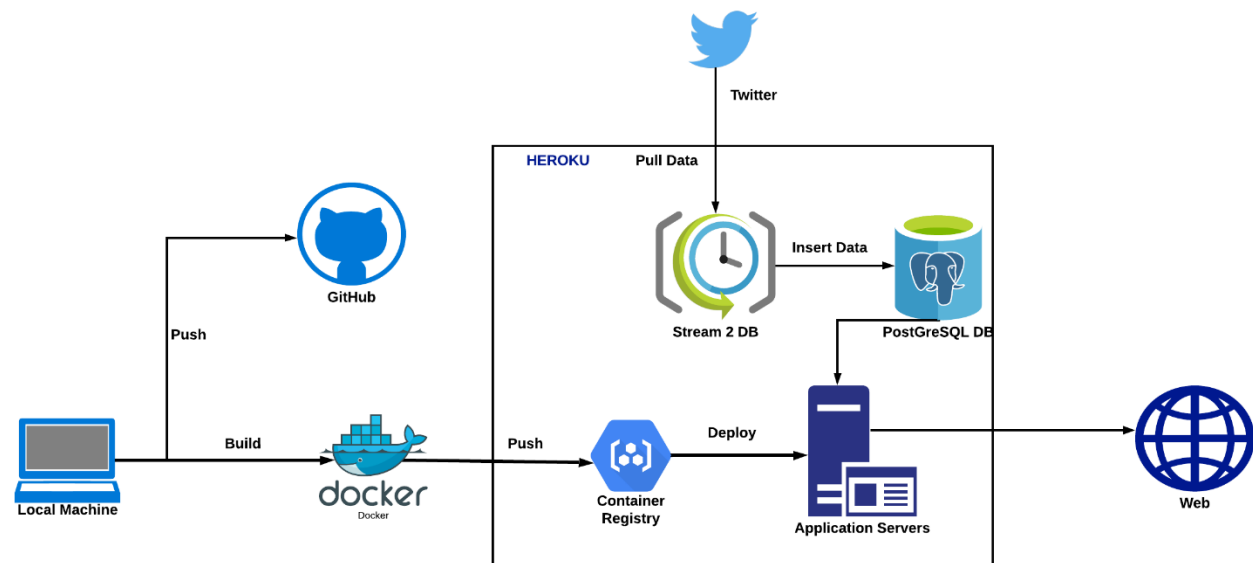


Figure 3: System architecture

Application is dockerized, using the guide on [1]. An image is build using Dockerfile. This image is pushed to Heroku's Container Registry and released using Heroku servers.

DATA COLLECTION

On the application, Twitter data is used. The data source is the Twitter API. To be able to use it, creating a developer account on Twitter. Using this account, it was possible to stream data continuously and look up the details of a post using its ID. F1, F4 and NF2 creates the constraints for data collection.

On Figure 4, a sample tweet is shown. This is not an original tweet but a retweet, that is re-posting a tweet without making any modifications. The user making the retweet can be seen on the upper end of the post, Nicola Wigmore, in this case. The owner of the original post is the user *@fhussain73*. Every user has a user name that starts with "@" and a screen name they can give themselves, for example Farzana Hussain. At the bottom the number of retweets, likes and replies are reported. For this project, special consideration is not given to replies. They are not treated only as individual tweets, disregarding their relation to each other (if ever streamed). In the content of the post, users can add hashtags (phrases written without spaces and starting with "#") to categorize their posts under a certain topic. Here, *#letsbeatcovitogether* and *#covidvaccine* are included hashtags. There are also user mentions where people can reference other Twitter users by their user names. In the sample *@MayorofLondon* is mentioned. For the project, both hashtags and user mentions is collected and analyzed.

Finally, it is possible to include media to the tweets. They are disregarded as well, as we do not have the means to analyze them within the project.



Figure 4: A sample tweet

In compliance with NF5, Python is the programming language. Python has a special package called Tweepy for Twitter API calls. This package is used for both stream and tweet lookup. The content provided by Twitter is very rich and has everything needed for our analysis. From every tweet, following attributes are collected:

- Post ID
- Post Content
- Post Date
- Username of Poster
- User Mentions (Screen name of the user instead of username)
- Hashtags
- Favourite Count
- Retweet Count

To guarantee the freshness of data scheduled job is designed. This job would collect 50 tweets every 10 minutes and a total of 7200 tweets would accumulate every day. The query filter in use is

```
{ track = ['corona','covid','coronavirus','covid19','covid-19'],  
  encoding = 'ascii',  
  languages = ['en']}
```

This would provide a reasonable amount to analyze and filter from. Daily collection is not increased further because data analysis and display depends mainly on the recency of the post and only a certain number among them are used. By keeping the daily amount to this level, displaying a longer time period was made possible. On Figure 5, the sequence diagram of this job can be seen. Data cleaning and analysis will be elaborated in the following subsections. The scheduled job is created on Heroku, not as a job within the application. This way the data layer is separated from the application, to keep the integrity and flow of data independent of the functioning of the application.

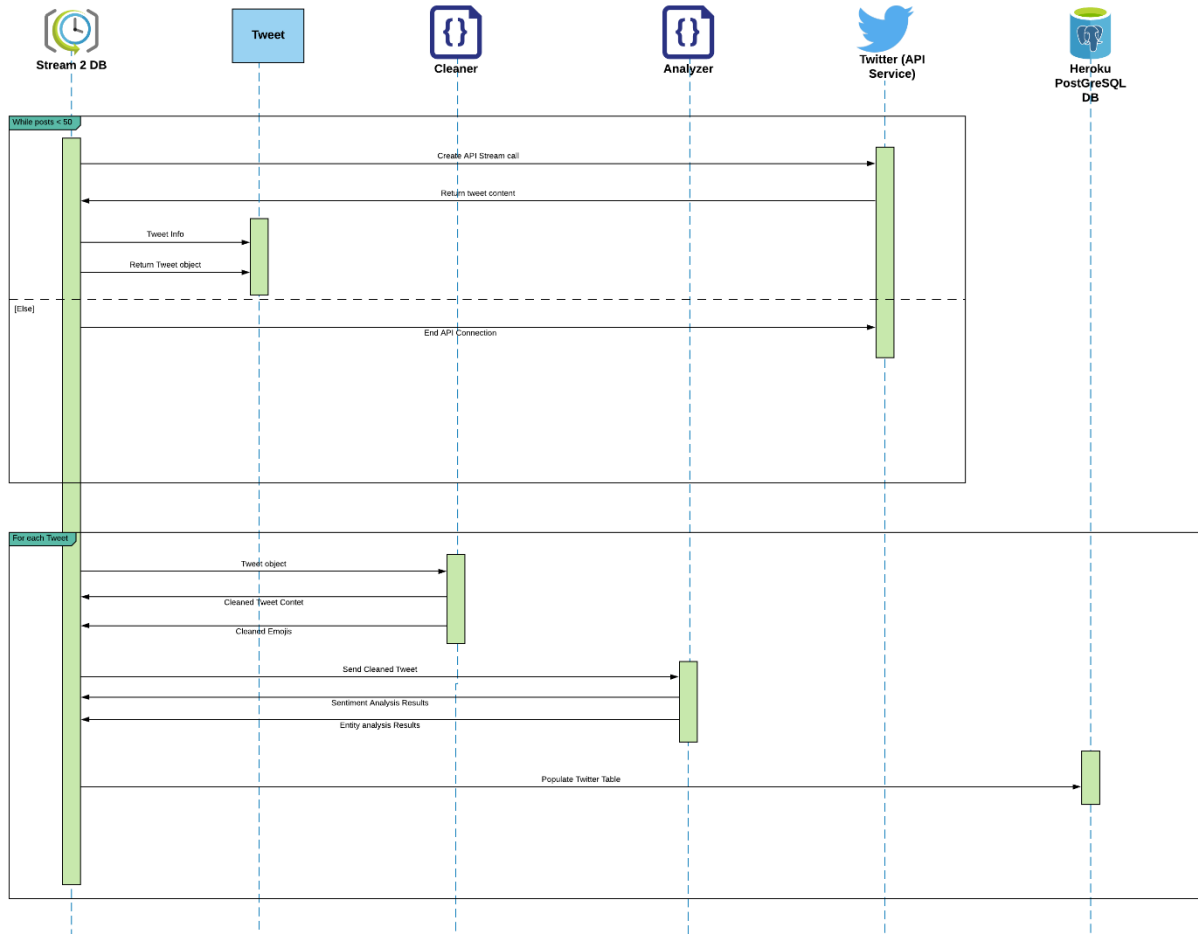


Figure 5: The sequence diagram of scheduled data collection job

In addition to this flow, there was need for posts earlier than project's start date. For that purpose, "snsraper" package of Python [2]. This package provides tweet IDs from a certain time. User can also filter certain content, such as COVID. Original plan was to employ this package for application's time interval search feature. Data was going to be streamed using this package and analysis and visualizations were all going to be executed live. However, after deployment to Heroku (which runs on AWS itself), it was realized that this packages IP's were banned by AWS. As a workaround, using the local machine for snsrape connection, 2000 tweet IDs per day were collected. Then, by Tweepy's lookup, the content of these posts are inserted to the database. The query for snsrape tweet ID collection is "covid since:{start_date} end:{end_date}". For that reason, it can be said that analysis for dates after mid-December would be healthier.

PRE-PROCESSING OF DATA

Before making any kind of analysis the data had to be cleaned [6]. The order and strategy followed is:

1. Standardize all tweets: Conversion to lower case and removal of new line or tab signifiers from the post content
2. Remove hashtags: The hashtags are stored and evaluated separately. To avoid re-evaluation, they are removed.

3. Remove mentions: Mentions to users are stored and evaluated separately. To prevent a user name from effecting the meaning of content, they are removed.
4. Remove URLs: At times some links are added to the tweet. Although these URLs may contain meaning both in abundance, where they link to and for bringing another perspective to the posts that contain them.
5. Replace Abbreviations: Abbreviations are frequently used in tweets. While analyzing they would not carry much meaning though. To identify the entities and analyze sentiment correctly, abbreviations are replaced with the phrases they imply. Slang and abbreviation list in [3] is used as the initial template and abbreviations specific to our case are added to it. For example, “us” is replaced with “United States” for to clear the meaning and to stop it from being removed at step 8.
6. Remove Punctuations
7. Convert Emojis: Emojis are very frequently used at tweets. They carry lots of meaning and sentiment for the analysis. To include them correctly, the emojis are replaced with their names. This way, emotions such as laughing, crying or anger are incorporated to the text.
8. Remove Stop Words: Stop words do not add much meaning to the content and they occur very frequently so they should be removed before analysis (Requirement F6.1). Stop words list in [4] is used as a template and it is customized further for specific cases of the application.
9. Make Lemmatization: By lemmatization, the roots of the words are determined so similarities between posts can be identified more accurately.

DATA ANALYSIS

There are 3 types analysis performed on the data.

Sentiment Analysis: There are several packages available for sentiment analysis on Python. Upon comparisons of these packages on sample tweets, “nltk”s “vader” package is used. This package returns a sentiment score between -1 and 1. 5 levels of sentiment results are set according to their scores as:

Very Negative: $-1 \leq s < -0.75$

Negative: $-0.75 \leq s < -0.25$

Neutral: $-0.25 \leq s < 0.25$

Positive: $0.25 \leq s < 0.75$

Very Positive: $0.75 \leq s \leq 1$

Entity Linking: Entities are identified in the posts using TagMe [5] package (Requirement NF10). Every identification returns a scoring. Entities with a score over 0.125 are taken into consideration. In overall, scores were generally low and identifications were not very correct. 0.125 provided a plausible amount of results with an acceptable accuracy. Entities appearing in each post are stored and evaluated accordingly.

Co-occurrence Graph: The co-occurrence networks are used to show relations and connections between entities. For the project, 3 co-occurrence graphs are created from 3 features of the tweet data; entities, hashtags and user mentions. For example, for entity co-occurrence, each entity is represented by a node in the graph. They are linked by an edge if they appear together in a post. Weight of the edges are calculated by the amount of time they appear together. In the end the graph obtained is undirected and weighted. Mention and hashtag graphs are created in the exact same manner.

Though they are very informative, it takes a very long time to create these graphs. It took about 10 minutes to create a network from 500 tweets. And a few more to make analysis on the graph. Given that the data size would be much greater for the application, the waiting time would not be acceptable for the user.

As a workaround to this problem, it was decided to create the graph from weekly data, store the adjacency matrix and centrality indices at the database and update this graph with a daily job. Also, because of the size of the graph, it would not have been very meaningful to view it on the page (unless it could be done interactively). Instead, analysis results of this graph are displayed on the application's weekly data page. For analysis of the graph, following indices are calculated. Python's "networkx" is used for all operations.

- **Strongest Connections:** Vertices that are connected by the most weighted edges. This would be the simplest indicator of co-occurrence.
- **Degree Centrality:** Degree centrality is the most intuitive measure of centrality. In this method, influence is measured by connectedness, that is the degree of a vertex.
- **Closeness Centrality:** Closeness centrality refers to the mean distance of a vertex to every other vertex in the network. It is measured by the geodesic path, that the shortest distance between vertices. The vertices with high closeness have easier access to information and high influence on other vertices.
- **Betweenness Centrality:** Betweenness deals with shortest paths as well. A vertex's betweenness is measured by the amount of geodesic paths among other vertices that pass by it. Such vertices are not necessarily well connected however a large amount of information passes on them. For example, in networks that are separated in clusters, there may be some vertices that lie between clusters or on the periphery. Such vertices have high betweenness and generally low connectedness. Removal of high betweenness vertices might severely disrupt communication in a network.
- **Eigenvector Centrality:** Eigenvector centrality considers not only the amount of the neighbors of a vertex but also the importance of these neighbors. The importance of a vertex is determined by the sum of the importance of all its neighbors. The eigenvector centrality might not be calculated sometimes because there is no convergence. In such cases, the component on the front-end is left empty.

DATABASE DESIGN

There were two options for databases, PostgreSQL or MongoDB. After deciding on the data collection method, it was more practical to go with a relational database (PostgreSQL) because the subset of the original data to be used was decided and most of the data on database was going to be analyzed (requirement NF7). There was no need to keep the original raw data in json format as well. The database is set on Heroku.

3 tables are used for all data needs. Their relationship is shown on Figure 6. "Twitter" is the main table that contains the Twitter post data. It is populated by the scheduled stream job and the batch past tweet transfer job. Detailed information on columns are given on Table 1.

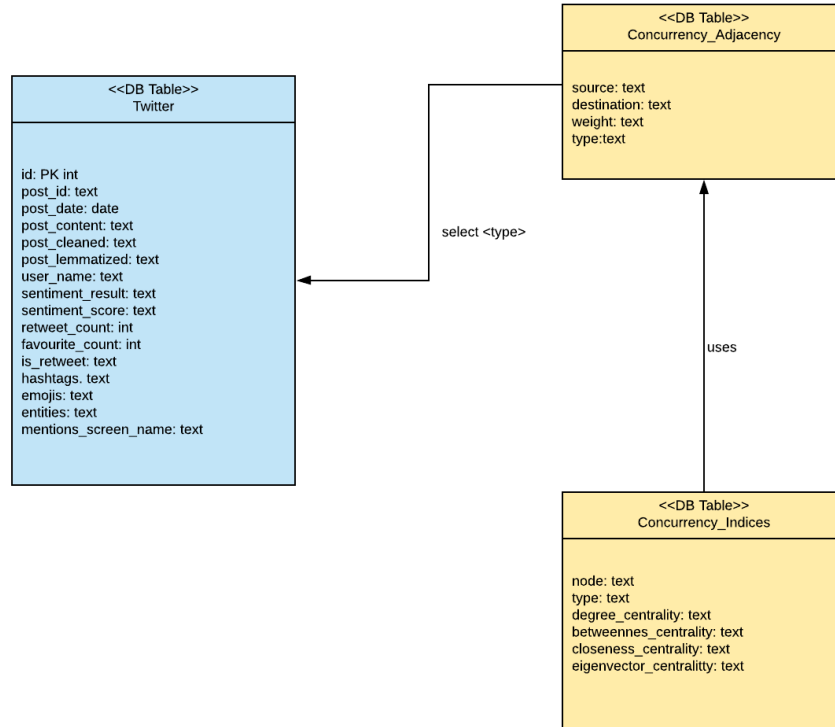


Figure 6: Relationship of database tables

Column	Type	Description
Id	Int	Primary key
Post_id	Text	ID of post on Twitter
Post_date	Date	Date that post is created
Post_content	Text	Original content of the post (before any processing)
Post_cleaned	Text	Cleaned version of the post content (output of pre-processing from steps 1-8)
Post_lemmatized	Text	Lemmatized version of post_cleaned (after pre-processing step 9)
User_name	Text	Username of the poster
Sentiment_score	Text	Numerical score obtained from sentiment analysis
Sentiment_result	Text	Sentiment category obtained, as shown on sentiment analysis section
Retweet_count	Int	Amount of retweets of the original post
Favourite_count	Int	Amount of likes original post received
Is_retweet	Text	If the post is a retweet (either “true” or “false”)
Hashtags	Text	Hashtags in the post. ‘#’ is remove. If post has more than 1 hashtag, they are combined in a string with token ‘ ’
Mentions_screen_name	Text	User mentions in the post. ‘@’ is remove. If post has more than 1 mention, they are combined in a string with token ‘ ’
Emojis	Text	Textual meaning of the emojis in the post. If post has more than 1 emoji, they are combined in a string with token ‘ ’
Entities	Text	Entities identified in the post as explained in entity linking section. If post has more than 1 entity, they are combined in a string with token ‘ ’

Table 1: “Twitter” table

The other 2 tables are used for the co-occurrence graph. Of them, “concurrency_adjacency” is for the adjacency matrix of the graph and “concurrency_indices” is for storing the centrality indices of the vertices. Data for 3 graphs; entities, mentions and tags, are all stored together on these tables. In the adjacency matrix, if the edge weight is 0, meaning no connection, that node pair is not stored in the table. For creation of the graph, “Twitter” table is used, by the scheduled job. Detailed information on columns are given on tables 2 and 3.

Column	Type	Description
Source	Text	One of the vertices that is connected by an edge (though named source there is no direction for the edge)
Destination	Text	Other vertex on the edge
Type	Text	Data source of the graph: entities, mentions or tags
Weight	Text	Weight of the edge between source and

Table 2: Concurrency_adjacency table

Column	Type	Description
Node	Text	Vertex of the graph
Type	Text	Data source of the graph: entities, mentions or tags
Degree_Centrality	Text	Degree centrality index
Betweenness_Centrality	Text	Betweenness centrality index
Closeness_Centrality	Text	Closeness centrality index
Eigenvector_Centrality	Text	Eigenvector centrality index

Table 3: Concurrency_indices table

SCHEDULED JOBS

There are 3 jobs scheduled, to keep the data fresh and accurate.

- **Data Stream:** This is the main job that streams data (50 posts) every 10 minutes. It retrieves data from Twitter API and populates “Twitter” table.
- **Update Interactions:** The retweet and favourite counts of the posts change constantly, especially for the very recent posts. This job runs continuously to update these values for the posts of the last 10 days.
- **Co-occurrence Network Update:** This is a daily job that creates the co-occurrence network of the week, updating “concurrency_adjacency” and “concurrency_indices” tables. During the update, there may be brief outages for co-occurrence analysis components on the front-end.

THE WEB APPLICATION

Python’s Django framework is used for programming. There are 2 apps that the actual application employs. One “User Management” is used for user activities such as keeping users, login and sign up actions. The views and templates for those pages are created and used from that app. It uses Django’s own “User” model and all actions are executed using its functions. The app that is dealing with the Twitter data and its analysis is “main”. It controls the backend, views and templates of “This Week” and “Custom Search” pages. For the front end, a free and publically available bootstrap theme by Creative Tim [7]. Every template used in the application is customized from this theme. The class diagram for the applications is given on Figure 7.

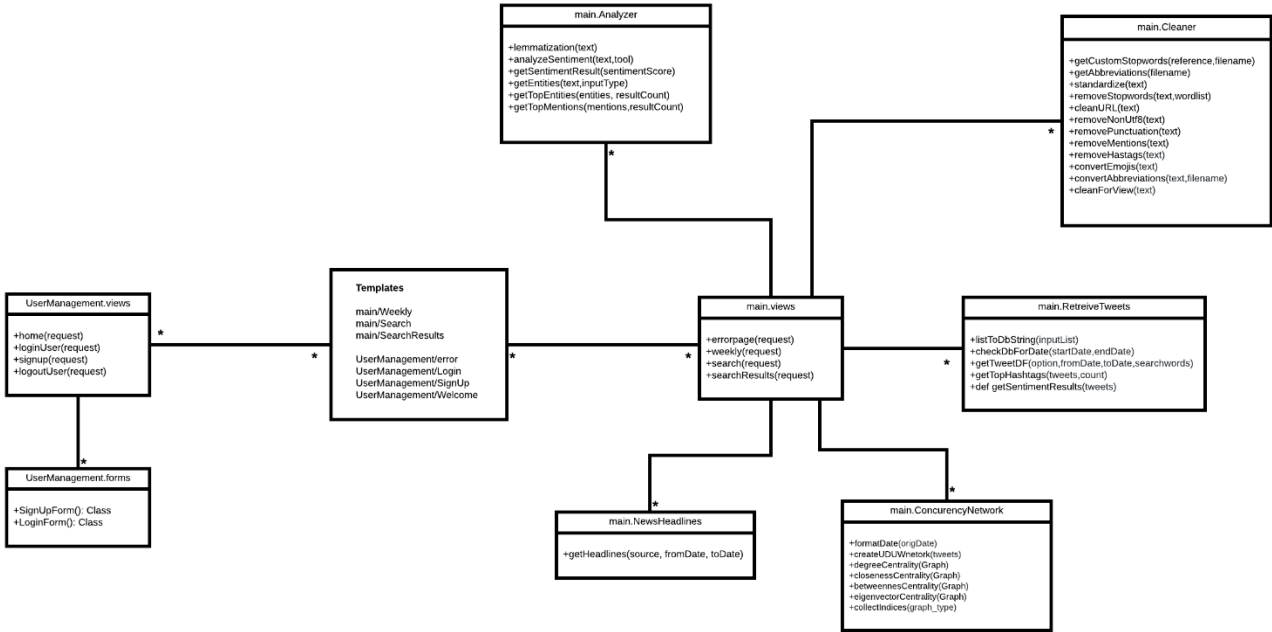


Figure 7: Class diagram of the application

SYSTEM MANUAL

The application is readily available on <https://covitter.herokuapp.com> and the source code can be cloned from https://github.com/melisMirza/SWE573_project. To use the system there are certain accounts one has to get. They are;

- Twitter developer account
- TagMe account
- News API account
- A database url that contains the data explained the “System Design” section.

An .env file has to be created by filling the values in the .env-sample file in the repository. It is placed at the root of the repository with (or in place of) the sample file. After obtaining all environment variables, there are two ways to run and customize the code.

Using pipenv

1. Install Python 3
2. Install pipenv by: `pip install pipenv`
3. Go to applications root directory: `cd SWE573_project`
4. Run `pipenv install`. This will install every dependency declared in the requirements.txt file.
5. After the virtual environment is created, go to it by; `pipenv shell`
6. Run the application by; `python manage.py runserver 0.0.0.0:8000` (for Windows) or `gunicorn CovitterAnalysis.wsgi:application --bind 0.0.0.0:8000` (for Mac)
7. Go to the `localhost:8000` to view the application.

Using Docker

As the application is dockerized, an image can be built and ran, by the following steps.

1. Build image; `docker build -t covitter -f Dockerfile.local .` (this is for local use, for deployment use Dockerfile instead)
2. Run; `docker run -it -p 8000:8000 --env-file=.env covitter` (`--env-file` is optional, add that if the application fails to read environment variables)
3. Go to *the localhost:8000* to view the application.

USER MANUAL

The application requires user registration. At the welcome page, Figure 8, a previously registered user may log in their account, if not they may sign up. If the user is already logged in, if it is cached by the browser, the welcome page is redirected to the “This Week” content page.

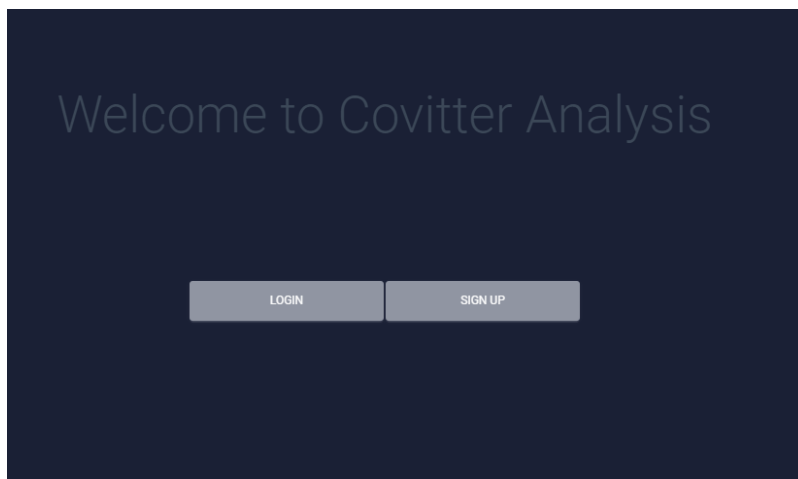


Figure 8: Welcome page

LOGIN AND SIGN UP

A new user can register to the application by providing the information on Figure 9; a unique user name, e-mail, a password of at least 8 characters and its confirmation, first name and last name. User’s input is checked by the system and in case of an unsatisfactory condition, an error message is display, such as “Passwords should be at least 8 characters.”. Once registration is completed successfully, the page is redirected to the login page.

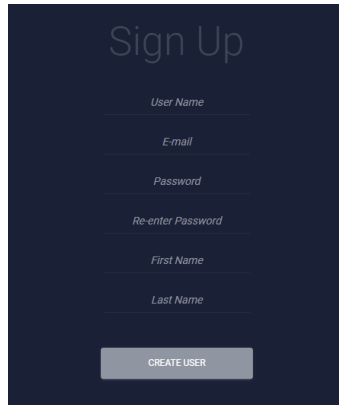
A dark-themed sign-up form with the title "Sign Up" at the top. Below the title are six input fields: "User Name", "Email", "Password", "Re-enter Password", "First Name", and "Last Name". At the bottom of the form is a button labeled "CREATE USER".

Figure 9: Sign up page

To access the system user should provide a valid user name and password to the page displayed on Figure 10. In cases of invalid inputs, "Username or password is incorrect." error message is displayed.

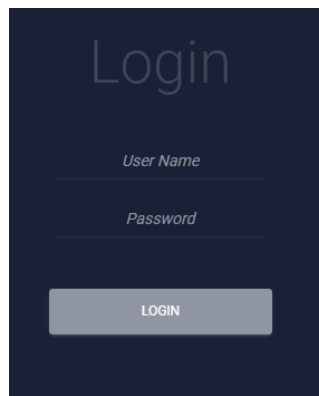
A dark-themed login page with the title "Login" at the top. Below the title are two input fields: "User Name" and "Password". At the bottom of the form is a button labeled "LOGIN".

Figure 10: Login page

There are 2 display options in the application, "This Week" and "Custom Search". The default opening page is "This Week" page. Both options are always available on the sidebar, Figure 11, user can switch between them to their liking. Logging out is done from the "Logout" button on the sidebar. Once logged out, the user is directed to the welcome page to either login or sign up.

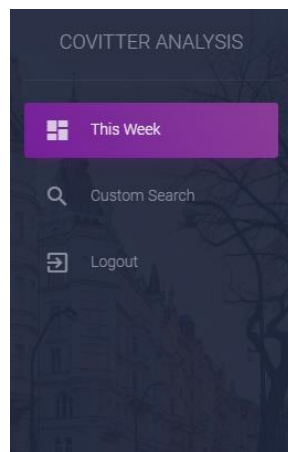
A sidebar menu titled "COVITTER ANALYSIS". It contains three items: "This Week" with a calendar icon, "Custom Search" with a magnifying glass icon, and "Logout" with a door icon. The "This Week" item is highlighted with a purple background.

Figure 11: Sidebar

“THIS WEEK” PAGE

“This Week” page contains the overall analysis of the content of the week. This page is not customizable and all components are pre-determined. On the top, there is a welcome message to the user. On the rest of the page, there are several displays, summarizing the analysis on the Twitter data of the week. Of all the collected data on the database, daily 500, in total 3500 posts are used on this page. The components on the page are as follows:

News Headlines: On this component, Figure 12, most popular news headlines of the week are presented. The purpose of this is to put the tweets and other analysis results to a perspective and to find some keywords, the user might want to search among the posts. 3 sources, CNN, Breitbart News and The Washington Post, were chosen, based on their political view, to portray a reasonable spectrum of the news. 10 headlines from each source is displayed and the user can switch between them using the tabs on the card header. The data for this component is obtained via an API call to the “News API” [8]. Therefore, the data is not stored but collected every time the page is called.

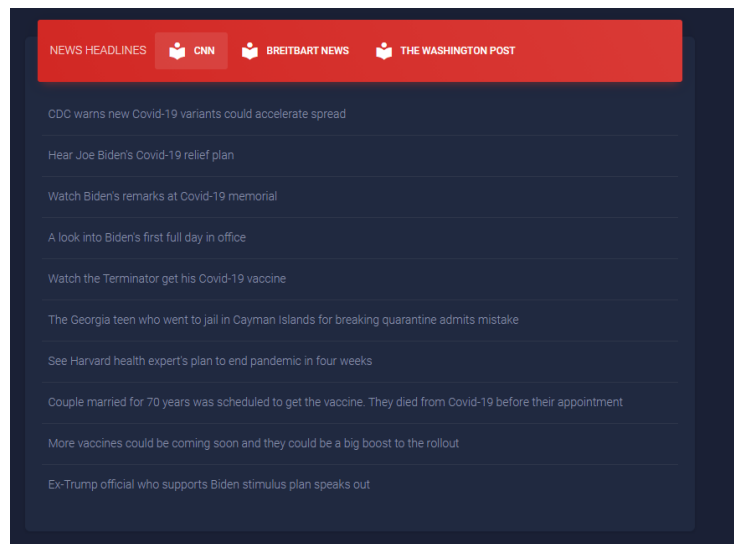


Figure 12: News headlines

Trends: Most frequently occurring, top 10 hashtags and the amount of their occurrences on 3500 posts are shown on this table, Figure 13. The Twitter API has a special endpoint for getting trends (at most a week old) however filtering through it is not possible. Therefore, the trends are calculated specially within the application. There was no cleaning procedure applied to the hashtags, such as omitting COVID related hashtags. All of the hashtags are taken into account. The data used for this table comes from application's database. Counting and sorting are done, when the page is called.

Trends		
Top 10 Trend		
	Hashtag	Occurance
1	covid19	181
2	covid	33
3	coronavirus	19
4	breaking	10
5	wearmask	8
6	b117	7
7	hankaaron	6
8	vaccine	4
9	covidmemorial	4
10	cdnpoll	4

Figure 13: Weekly trends

Entities: Entities are identified and stored at the database for each post content. Out of the 3500 posts, these entities are ranked and most frequently occurring 15 are displayed on the bar chart. The amount of total occurrence can both be seen on the chart and when hovered over the bar, as shown on Figure 14.

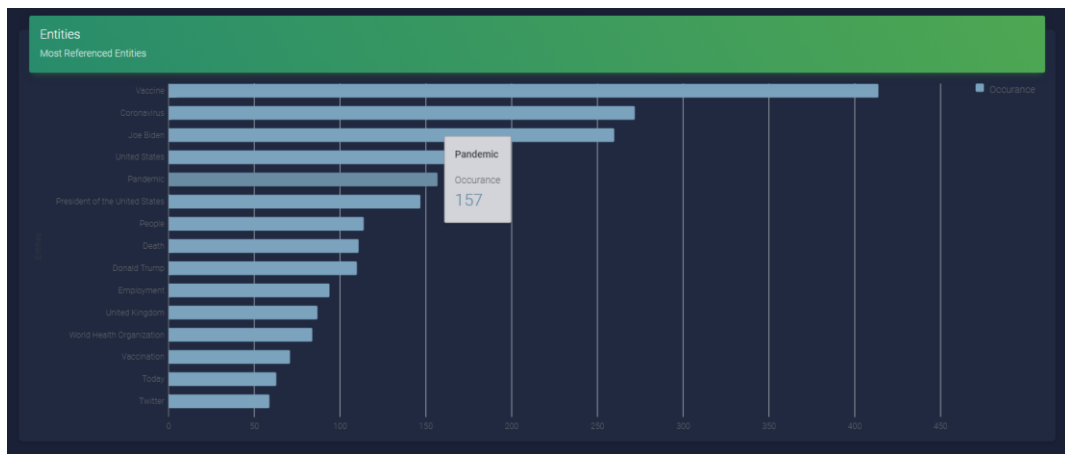


Figure 14: Most occurring entities

Co-occurency Network Analysis (Entities): The analysis results of co-occurency network, generated using the entities identified on the posts and their co-occurency, are shown on this component. There are 5 different analysis types; strongest connections, degree centrality, betweenness centrality, closeness centrality and eigenvector centrality.

On the “Strongest Connections” tab, entities that are linked with the top 10 highest weighted edges are displayed, Figure 15. Their weight shows the amount of posts they appear together.

CONCURRENCY GRAPH ANALYSIS (ENTITIES)		
STRONGEST CONNECTIONS		
EIGENVECTOR CENTRALITY		
Entity	Entity	Weight
President of the United States	Joe Biden	93
You	If	37
Old age	Vaccine	34
People	Joe Biden	34
Pandemic	Coronavirus	32
Coronavirus	Vaccine	29
Coronavirus	Joe Biden	28
President of the United States	Donald Trump	26
Ballet	Vaccine	26
Old age	Ballet	26

Figure 15: Strongest connections of co-occurrence graph analysis of entities

On the tabs for centrality results, top 10 vertices (entities) with the highest index are displayed with that index value. As additional information, the strongest connections these vertices have are given, with their edge weights. On Figure 16, degree centrality tab is shown. Content for the other centrality tabs are the same.

CONCURRENCY GRAPH ANALYSIS (ENTITIES)		
STRONGEST CONNECTIONS		
DEGREE CENTRALITY		
BETWEENNESS CENTRALITY		
CLOSENESS CENTRALITY		
EIGENVECTOR CENTRALITY		
Vertex	Index	Most Weighted Neighbours (weight)
Vaccine	0.152082	Joe Biden (11) World Health Organization (11) Virus (6) African Americans (6) Australia (6)
Coronavirus	0.124044	Vaccine (29) Joe Biden (28) Donald Trump (17) World Health Organization (13) Infection (10)
United States	0.090484	Joe Biden (18) Coronavirus (18) Pandemic (14) President of the United States (12) Vaccine (12)
Pandemic	0.085599	Coronavirus (32) Joe Biden (20) World Health Organization (8) Vice president (4) Time (4)
Joe Biden	0.074554	Vice President of the United States (12) World Health Organization (9) Climate change (7) Twitter (3) Economy (2)
Death	0.069244	Patient (13) Vaccine (9) Employment (8) United Kingdom (8) Cancer (8)
Employment	0.063297	Instacart (14) Dismissal (12) Trade union (11) Strike action (6) Person (6)
United Kingdom	0.054163	Coronavirus (16) Mortality rate (15) Brazil (13) Government of the United Kingdom (10) Joe Biden (9)
World Health Organization	0.053951	Black Lives Matter (2) Twitter (2) Light (1) Syrian people (1) Interim management (1)
President of the United States	0.053314	Joe Biden (93) Donald Trump (26) Coronavirus (21) Pandemic (15) Computer virus (15)

Figure 16: Degree centrality tab of co-occurrence graph analysis of entities

Sentiment Analysis: Sentiment analysis results are pre-calculated and stored at the database. When the page is called, they are just visualized on the chart. There are 5 levels for sentiments; very negative, negative, neutral, positive and very positive. Results on the chart are the percentages of each sentiment on daily basis. When hovered over the bar, details can be viewed on Figure 17.

There is a thing to consider while running the system on local machines. There is a 3-hour difference between the local time and the data time, so the results of the date should be analyzed accordingly.



Figure 17: Sentiment analysis results

Co-occurrence Network Analysis (Hashtags): The analysis results of co-occurrence network, generated using the hashtags on the posts and their co-occurrence, are shown on this component. There are 5 different analysis types; strongest connections, degree centrality, betweenness centrality, closeness centrality and eigenvector centrality.

On the “Strongest Connections” tab, entities that are linked with the top 10 highest weighted edges are displayed, Figure 18. Their weight shows the amount of posts they appear together. The tabs for centrality indices are the same as the ones for entities, as on Figure 16.

CONCURRENCY GRAPH ANALYSIS (HASHTAGS)		
STRONGEST CONNECTIONS		
EIGENVECTOR CENTRALITY		
Hashtag	Hashtag	Weight
vaccine	covid19	6
covid19	coronavirus	5
failedresponse	wearmask	4
failedresponse	covid	4
wearmask	covid	4
covid	biden	3
vaccine	coronavirus	3
covid19	b117	3
sosbrazil	dia23impesachmentja	2
covidvaccine	bahubali	2

Figure 18: Strongest connections of co-occurrence graph analysis of hashtags

User Mentions: Mentions to users are achieved by the “@username” notations on the posts. Before writing to database, user given screen names are read from the API and they are stored instead of the user names. On this component, most frequently mentioned 15 users are displayed. When hovered over the bar, details can be viewed, Figure 19.

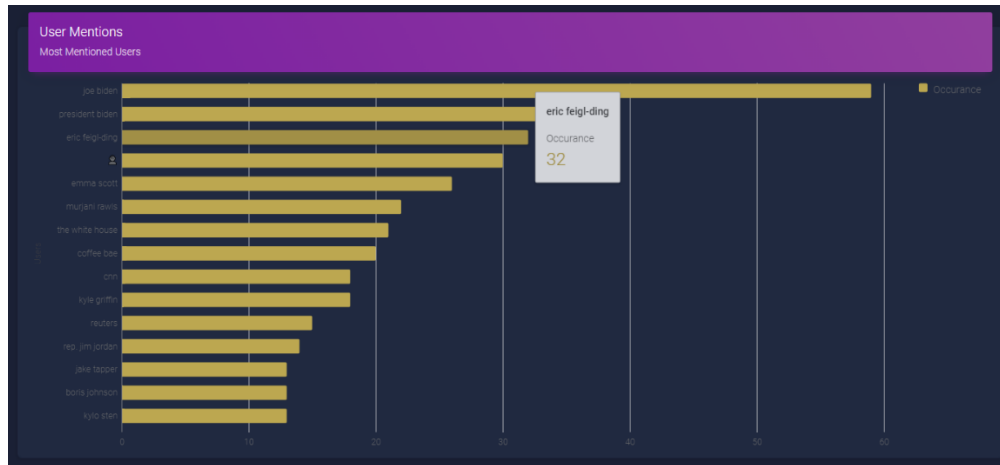


Figure 19: Top user mentions

Co-Occurancy Network Analysis (Mentions): The analysis results of co-occurrence network, generated using the user mentions on the posts and their co-occurrence, are shown on this component. There are 5 different analysis types; strongest connections, degree centrality, betweenness centrality, closeness centrality and eigenvector centrality.

On the “Strongest Connections” tab, entities that are linked with the top 10 highest weighted edges are displayed, Figure 20. Their weight shows the amount of posts they appear together. The tabs for centrality indices are the same as the ones for entities, as on Figure 16.

CONCURRENCE GRAPH ANALYSIS (MENTIONS)		
STRONGEST CONNECTIONS		
EIGENVECTOR CENTRALITY		
User	User	Weight
votevets	ted cruz	7
jill biden	whitman-walker	6
jill biden	cancer support community	6
tony shaffer	joe biden	6
vice president mike pence	second lady karen pence	6
vice president mike pence	don winslow	6
don winslow	second lady karen pence	6
cancer support community	whitman-walker	6
marjorie taylor greene us	joe biden	5
kamala harris	joe biden	5

Figure 20: Strongest connections of co-occurrence graph analysis of user mentions

Tweets: On this component, a sample of processed tweets are displayed. Instead of the daily 500 tweets, only 100 of them, per day, are selected randomly. For these tweets, post date, original post content, favourite and retweet counts and sentiment analysis results are displayed, Figure 21. There is also link to the actual tweets on Twitter that the user can be directed to, by clicking anywhere on the row of that tweet. This way the user can check the post further, for example for the media content in the post or the replies it received.

Right above the table, a search box is provided to search for phrases or numbers on any column or row of the table. The search function is reactive and responds quickly. The system searches for an exact match.

The most recent tweets are likely to be streamed in the last 10 minutes and therefore the retweet and favourite counts are likely to be 0. Further down the table, earlier tweets will have more accurate counts.

Tweets						
Content 700 The Search						
Search...						
	Post Date	Post	Favourited	Retweeted	Sentiment	
1	Jan. 26, 2021	@kwawkes Look 🙏🏿 Our Voices Have been Heard.. Please @JoyNewsOnTV when will this take effect especially at the airport. I beg try make it take effect Before I come Home lol. Thank you Ecowas Chair @NAkufoAddo #FullMotion https://t.co/nCq8P37ud3	0	0	Positive	
2	Jan. 26, 2021	@davidschneider One minute silence for each covid death and you would be standing for... UK: 2 months Australia: 15 hours New Zealand: 25 minutes	0	0	Negative	
3	Jan. 26, 2021	Can your boss force you to take the Covid-19 vaccine? Here's what you should know https://t.co/bad9C6q3xr	0	0	Neutral	
4	Jan. 26, 2021	@cspbame an informative piece on the #COVID19 vaccine	0	0	Neutral	

Figure 21: Tweets table

"CUSTOM SEARCH" PAGE

On the custom search page, the user can retrieve tweets from a certain time interval, from or to a certain date or tweets containing certain words, Figure 22. There are at least 2000 posts for everyday from 10th February 2020.

Start date:

mm/dd/yyyy

End date:

mm/dd/yyyy

Search...

Figure 22: Search options

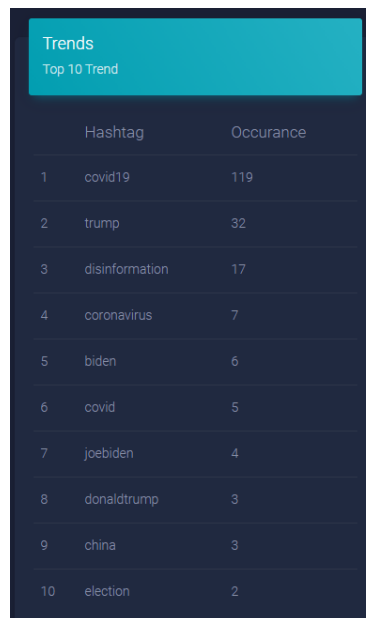
There are combinations of the search criteria:

- If all 3 inputs are provided, up to 1500 posts (at times there are not enough data that comply with the search criteria) are retrieved from the given time interval including the searched word. These posts are used for analysis. To make sure data from every day of the given period is considered, 1500 posts are divided equally among the days.

- If no date is provided, the word(s) is searched among the most recent posts and up to 1500 posts are retrieved. It passes a “like” SQL query to the database, so it searched for the exact matches (prefixes or postfixes are included in the results).
- If no search words are provided, 1500 posts are divided equally among the days and analyzed.
- If start date is left empty, most recent 1500 posts up to the given end date are retrieved. If a search word provided, it is added to the filter as well.
- If end date is left empty, 1500 posts starting from the given start date are retrieved. If a search word provided, it is added to the filter as well.

The contents of this page is similar to “This Week” page. As a main difference, co-occurrence graph is not created for this page and analysis components are omitted. The reason for that is the graph is not created upon call but as a fixed job for weekly data criteria. Rest of the components are as follows:

Trends: Most frequently occurring, top 10 hashtags and the amount of their occurrences on the posts are shown on this table, Figure 23. There was no cleaning procedure applied to the hashtags, such as omitting COVID related hashtags. All of the hashtags are taken into account. The data used for this table comes from application’s database. Counting and sorting are done, when the page is called.



Trends	
Top 10 Trend	
	Hashtag
	Occurance
1	covid19 119
2	trump 32
3	disinformation 17
4	coronavirus 7
5	biden 6
6	covid 5
7	joe Biden 4
8	donald trump 3
9	china 3
10	election 2

Figure 23: Trends of the filtered posts

News Headlines: On this component, just like the one “This Week” page, Figure 11, most popular news headlines of the selected time period are presented. 3 sources, CNN, Breitbart News and The Washington Post, were chosen, according to their political views, to portray a reasonable spectrum of the news. 10 headlines from each source is displayed and the user can switch between them using the tabs on the card header.

The data for this component is obtained via an API call to the “News API”. Therefore, the data is not stored but collected every time the page is called. The “News API” returns the news for the last month alone, so if the search period is earlier than that, only a message is displayed here, Figure 24.

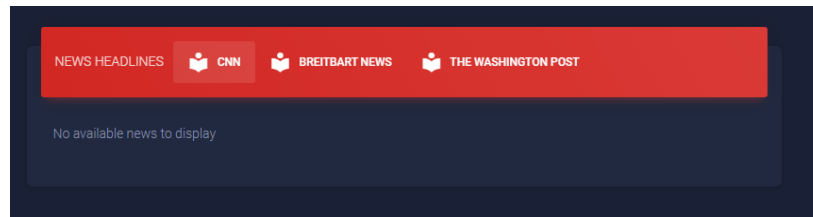


Figure 24: News headlines for a time period earlier than a month

Entities: Entities are identified and stored at the database for each post content. Out of the retrieved posts, these entities are ranked and most frequently occurring 15 are displayed on the bar chart. The amount of total occurrence can both be seen on the chart and when hovered over the bar, as shown on Figure 14.

Sentiment Analysis: Sentiment analysis results are pre-calculated and stored at the database. When the page is called, they are just visualized on the chart. There are 5 levels for sentiments; very negative, negative, neutral, positive and very positive. Results on the chart are the percentages of each sentiment on daily basis. When hovered over the bar, details can be viewed. Unlike “This Week” page, the number of days may differ according to the filtering of the search, Figure 25.

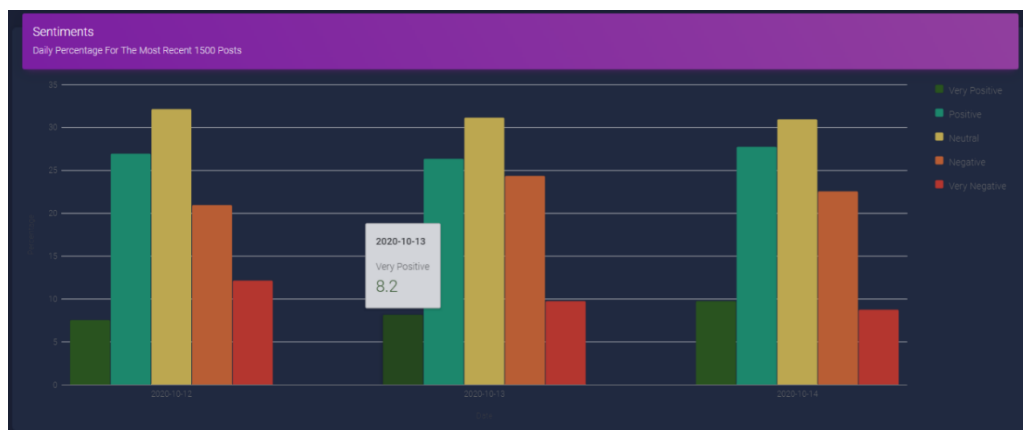


Figure 25: Sentiment analysis results for the filtered search

User Mentions: Mentions to users are achieved by the “@username” notations on the posts. Before writing to database, user given screen names are read from the API and they are stored instead of the user names. On this component, most frequently mentioned 15 users are displayed. When hovered over the bar, details can be viewed, Figure 19.

Tweets: On this component, a sample of processed tweets are displayed. All of the analyzed tweets are displayed on this table (at most 1500). For these tweets, post date, original post content, favourite and retweet counts and sentiment analysis results are displayed, Figure 21. There is also link of the tweets that the user can be directed to, by clicking anywhere on the row of that tweet.

Right above the table, a search box is provided to search for phrases or numbers on any column or row of the table. The search function is reactive and responds quickly. A sample search is shown on Figure 26.

Tweets									
Content 1500 The Search									
trump									
	Oct.								
10	14, 2020	@regcoral	He must do antiTrump videos. Those people around him are incredibly bad influenced. Covid infectious senators are participating in the Confirmation hearings without masks.				1	0	Negative
	Oct.								
12	14, 2020	@HouseGOP @GOPLeader @RepKevinBrady	Yes it's about Covid . Trump and Kevin McConnell have been responsible for the deaths of so many thousands of people. It's no less than criminal!				1	0	Negative
	Oct.								
16	14, 2020	@TeamTrump @Mike_Pence	Did he lie about COVID-19 to people there like he did in February 2019, while privately telling his wealthy donors it was much worse? Asking for 211,000 dead Americans and their families. See https://t.co/EVsvNB8rsd				0	0	Negative
	Oct.								
21	14, 2020	Trump's first day of COVID-19 was Day 200 for me. Here's what I think of his 'victory'	https://t.co/dAYhMSTwVvk via @USATODAY				0	0	Neutral
	Oct.								
22	14, 2020	@santiagomayer_	But yeah, Trump has had him at the White House when he tried to get these big companies to open everything up. Apparently he touted a vitamin(?) that will cure Covid... why did they break up because the pillow guy is obsessed with Trump!				1	0	Neutral

Figure 26: Searching a word on the Tweets table

TEST RESULTS

Below are the results of some of the functional test scenarios.

- Sign Up Page**

Sign up page is tested for reaction to incorrect inputs. On Figure 27, application's response can be seen for incorrect password length, not matching password confirmation and existing user. Application is acting correctly for these inputs.

The figure shows three screenshots of the 'Sign Up' page, each with a different error message at the bottom:

- Left Screenshot:** The 'Password' field contains 'admin2' and the 'Re-enter Password' field contains 'mel'. The error message at the bottom is 'Password should be at least 8 characters'.
- Middle Screenshot:** The 'Password' field contains 'admin2' and the 'Re-enter Password' field contains 'mel'. The error message at the bottom is 'Password confirmation does not match'.
- Right Screenshot:** The 'Password' field contains 'admin' and the 'Re-enter Password' field contains 'mel'. The error message at the bottom is 'User already exists'.

Figure 27: Error messages from the sign up page

However, there is a bug with the system for invalid e-mail address. E-mail is validated as accepted but desired message cannot be displayed. This stays as a known bug.

- **Login Page**

There is a single error message for the login page. If there is a problem with either the username or password, “username or password is incorrect” message is displayed, as in Figure 28.

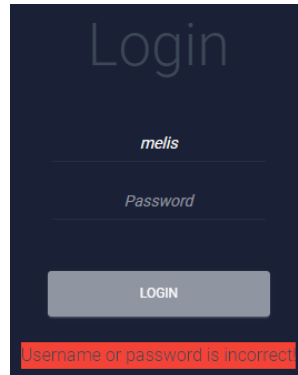


Figure 28: Error messages from the login page

- **Accessing Tweet from Tweets Table**

There is a direct link from the tweets table to the actual tweet for every row of data. By clicking the row on Figure 29, the tweet on Figure 30 is accessed as expected.

Jan. 373 25, 2021	COVID Vaccines: What do you think? We've been having lots of conversations about this over here. Excited for @VOCALNewYork's town hall this Wednesday with @DrOniBee and @DrKimSue	14	8	Positive
----------------------------	--	----	---	----------

Figure 29: A tweet from Tweets table



Figure 30: Actual tweet accessed from Tweets table

- **Rendering Page**

While testing the deployed page, there is a persistent problem with rendering bar charts. Heroku keeps applications on stand-by if they are not called for a certain time. “Unknown renderer type” error is displayed on Figure 31. This error cannot be fixed but when the page is refreshed, charts are rendered correctly.



Figure 31: Rendering Error

- **Searches with No Results**

In case of some searches, there are no results to display. In such cases, the page is presented as in Figure 32. There are not any errors however, all components are left empty.

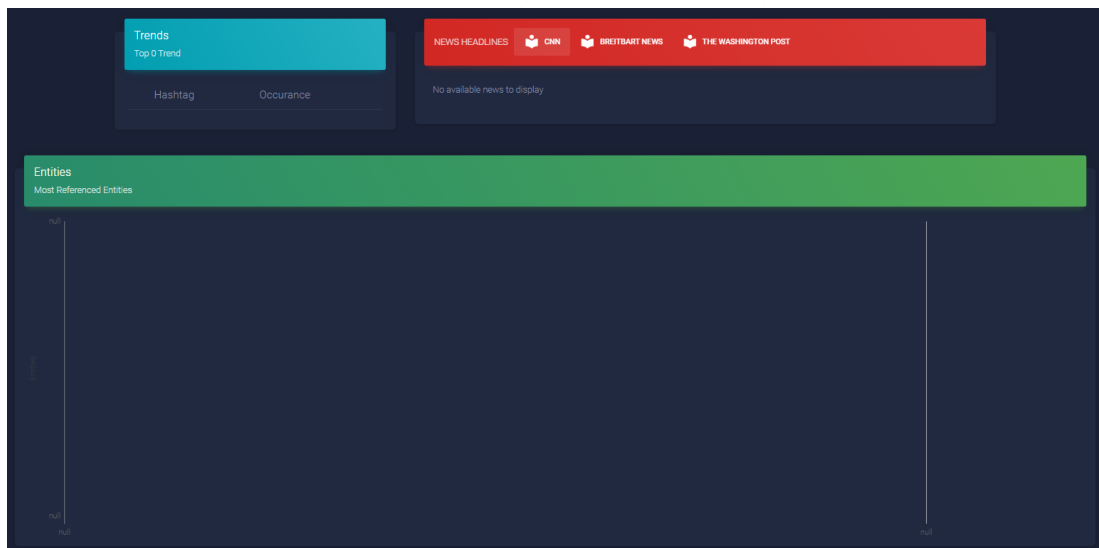


Figure 32: The case of no results

- **The Error Page**

When there is an unexpected error, the special error page is displayed. It is tested extensively; the user is always correctly redirected to the error page, Figure 33.

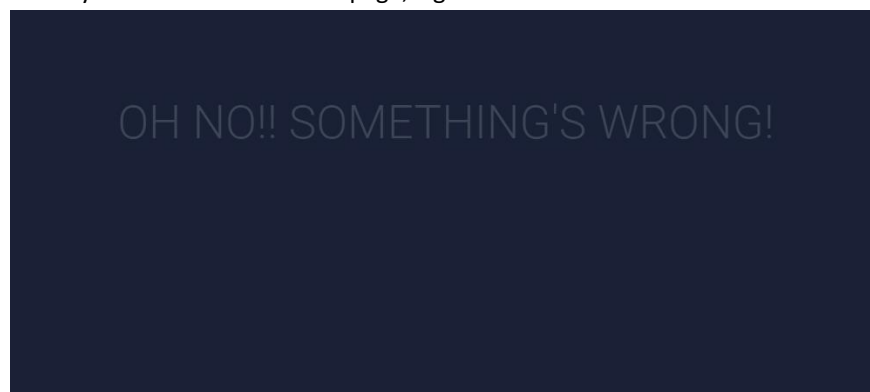


Figure 33: The error page

OBSERVATIONS

While making the analysis, there have been certain observations I have made. They are briefly listed here:

- Independent of the time interval of the posts, most frequently mentioned users and most frequently occurring entities are dominantly politicians. On Figure 34, two samples entities and mentions for the weeks starting from 20th January 2021 and from 13th September 2020.

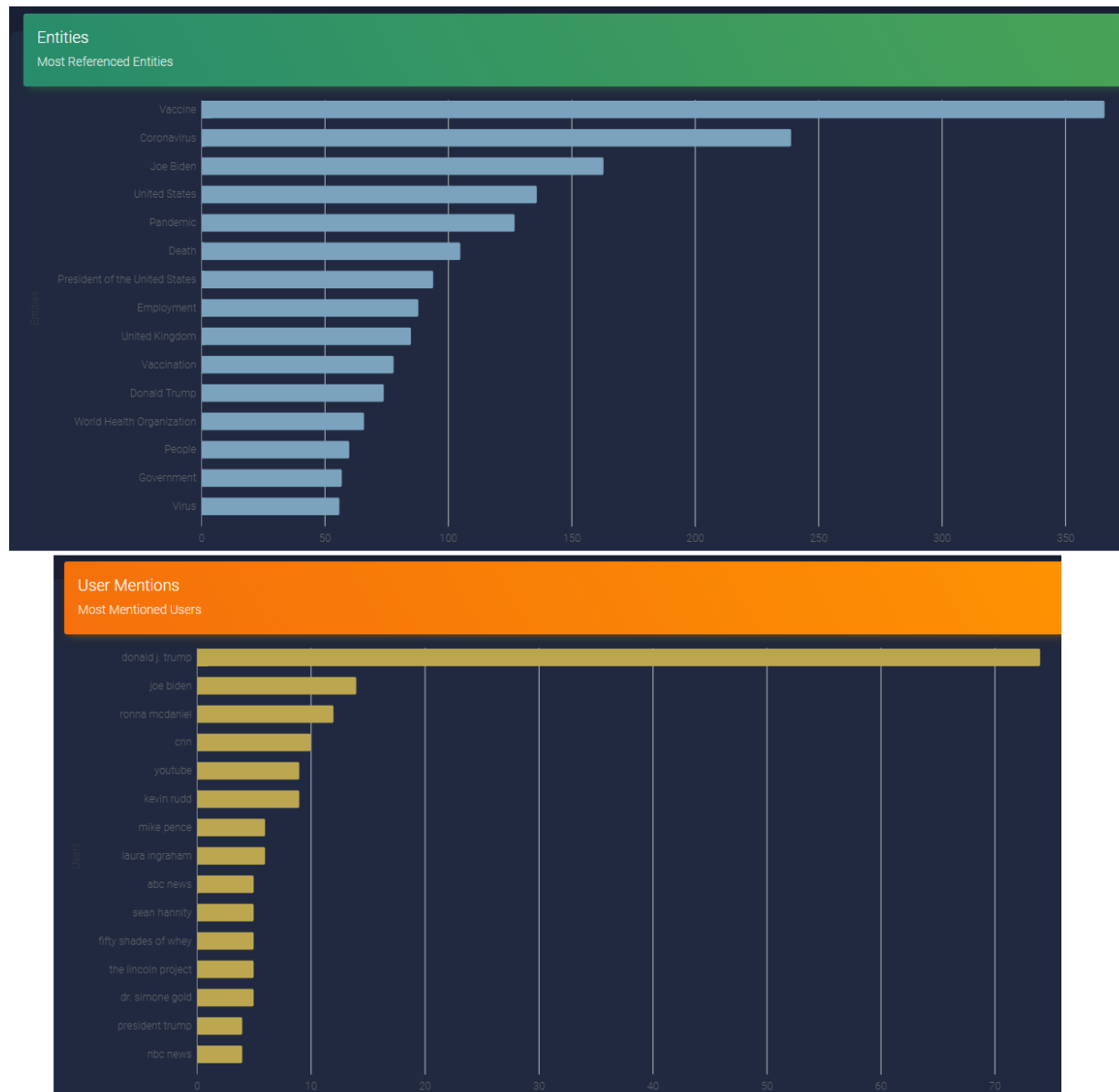


Figure 34: Entities and mentions containing references to politicians

- On the Covid subject, retweets were very frequent. For the data collected between 1st - 25th of January 2021, 69% of the overall tweets were retweets. There was a considerable amount that was news, provocative political and humor content among them.

- There are pornographic and promotional hashtags attached to the Covid related hashtags, as a strategy to gain visibility. On Figure 35, some pornographic content can be observed on the top of the degree centralities.

CONCURRENCY GRAPH ANALYSIS (HASHTAGS)		
STRONGEST CONNECTIONS		
DEGREE CENTRALITY		
BETWEENNESS CENTRALITY		
CLOSENESS CENTRALITY		
EIGENVECTOR CENTRALITY		
Vertex	Index	Most Weighted Neighbours (weight)
covid19	0.202454	warroompandemic (3) playschool (2) ontario (2) vaccine (2) b117 (2)
covid	0.061350	failedresponse (3) conservative (1) abbottsucks (1) deaththreats (1) paying (1)
hotwife	0.055215	hotmilf (1) dogging (1) horny (1) fuck (1) uptoswap (1)
coronavirus	0.052147	covid19 (3) iran (3) covid-19 (1) update (1) lockdown2021 (1)
horny	0.042945	nakedselfie (1) bwc (1) park (1) exibicionist (1) swingers (1)
covid-19	0.039877	corona (1) nsfwtwitter (1) bbw (1) mask (1) horny (1)
love	0.036810	hotmilf (1) dogging (1) fuck (1) beauty (1) hotwife (1)
hotmilf	0.036810	dogging (1) fuck (1) beauty (1) wife (1) public (1)
dogging	0.036810	fuck (1) beauty (1) wife (1) public (1) milf (1)
fuck	0.036810	beauty (1) wife (1) public (1) milf (1) carsex (1)

Figure 35: Pornographic hashtags with high degree centralities

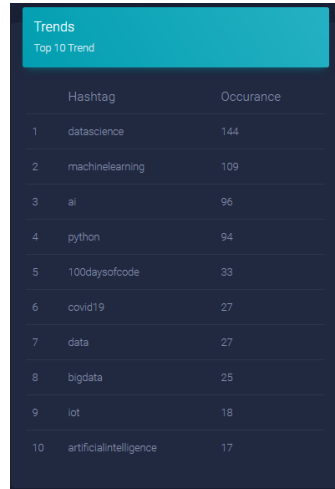
- Looking at the overall picture, it can be said that people are not likely to use multiple mentions or especially hashtags in a single post. This can be seen from the “Strongest connections” tab of both mentions and hashtags. For example, on Figure 36, out of 3500 posts even the most co-occurring pair occurs together only on 3 posts.

CONCURRENCY GRAPH ANALYSIS (HASHTAGS)		
STRONGEST CONNECTIONS		
DEGREE CENTRALITY		
BETWEENNESS CENTRALITY		
CLOSENESS CENTRALITY		
EIGENVECTOR CENTRALITY		
Hashtag	Hashtag	Weight
wearmask	failedresponse	3
python	datascience	3
ai	machinelearning	3
wearmask	covid	3
coronavirus	covid19	3
coronavirus	iran	3
ai	python	3
ai	datascience	3
ai	5g	3
python	machinelearning	3

Figure 36: Low number of co-occurrences of hashtags

- On Figure 36, hashtags on AI, Python, data science and machine learning can be observed. When “datascience” is searched using the application, top 10 hashtags (Figure 37) are all related topics. Examining the tweets further it can be seen that, they are mainly retweets and that while some of them

are news regarding the data analysis results most are posts promoting online courses on machine learning and data science.



Trends		
Top 10 Trend		
	Hashtag	Occurance
1	datascience	144
2	machinelearning	109
3	ai	96
4	python	94
5	100daysofcode	33
6	covid19	27
7	data	27
8	bigdata	25
9	iot	18
10	artificialintelligence	17

Figure 37: Hashtags retrieved from “datascience” search

- Since 23rd January 2021, a new humorous trends started. People are expressing their weariness of COVID with “once Covid is over is starting to sound a lot like ...” posts. Some examples are:
 - “once Covid is over” is starting to sound a lot like “when Vikings win the Super Bowl”.
 - “once Covid is over” is starting to sound a lot like “maybe he’ll change”.
 - “once Covid is over” is starting to sound a lot like “when Rihanna releases a new album”.
 - “once Covid is over” is starting to sound a lot like “when I win the lottery”.
- “Vaccine” has been the leading the top entities list since the companies announced successful results and the beginning of vaccination. “Death” as an entity has dropped down a couple of ranks (though still present) on that list although there are still very large numbers of deaths caused by Covid.

SUGGESTIONS FOR FUTURE DEVELOPMENT

There is some further work that can be done with the application. Below are some suggestions for improvements and new features.

Sentiment Analysis

- For sentiment analysis, nltk’s vader package was used. This package yielded plausible results but analysis can be much improved by using machine learning techniques for teaching sentiments. Then, the analysis will be more application specific too.
- In addition to sentiment analysis, emotion analysis can be added. For that it is almost certain that machine learning techniques must be used.

User Management

- User profile should be enriched. Features may be; storing earlier activity, scheduling analysis tasks that will be mailed to user later on.

- Paid membership may be an option. This way the user can access the news earlier than a month (since News API has fees for it) or use a greater number of posts for analysis.

Saving / Exporting Data

- Currently, only display of data is possible. Exporting the results in json or PDF formats can be made available to user.

Entities

- A link to relevant Wikipedia page could be added.
- Entity specific analysis may be presented as an option, such as applying the analysis only to posts containing that entity.

Co-occurrence Graph

- Currently the graph is not shown, only the results of the analysis are displayed. Front-end can be improved to make the graph interactive to the user, like in Gephy.

REFERENCES

1. Mitchel, J. *Deploying Django on Docker to Heroku with OpenCV*. <<https://www.codingforentrepreneurs.com/blog/deploy-django-on-docker-to-heroku-opencv>>
2. JustAnotherArchivist. *snsrape*. <<https://github.com/JustAnotherArchivist/snsrape>>
3. rishabhverma17. *sms_slang_translator*. <https://github.com/rishabhverma17/sms_slang_translator/blob/master/slang.txt>
4. neilkod. *gist:1319966*. <<https://gist.github.com/neilkod/1319966>>
5. Ferragina, P. *TagMe*. <<https://tagme.d4science.org/tagme/>>
6. Sudalairajkumar. *Getting started with Text Preprocessing*. <<https://www.kaggle.com/sudalairajkumar/getting-started-with-text-preprocessing>>
7. Creative Tim. *Material Dashboard Dark Edition*. <<https://www.creative-tim.com/product/material-dashboard-dark>>
8. *News API*. <<https://newsapi.org/>>
9. WebFX. *Emoji Cheat Sheet*. <<https://www.webfx.com/tools/emoji-cheat-sheet/>>
10. Fontanos, C. V. *Creating a Registration Page in Django*. <<https://carlofontanos.com/creating-a-registration-page-in-django/>>
11. Twitter. *API Reference*. <<http://docs.tweepy.org/en/latest/api.html>>
12. Django. *“Writing Your First App”*. <<https://docs.djangoproject.com/en/3.1/intro/tutorial01/>>