SENTIMENT ANALYSIS OF SOCIAL MEDIA WITH DATA VISUALISATION

BY

JOHN KELLY

SOFTWARE DEVELOPMENT

DESIGN DOCUMENT

5TH APRIL 2017

TABLE OF CONTENTS

ABSTRACT

The purpose of this document is to describe the architecture and the system design of this application. It will outline the core functionality of the project and wireframes of the UI will be presented.

1. INTRODUCTION

This document will outline the architecture required to deliver all the requirements stated in the functional specifications. It will be modified throughout the development of the application to reflect the latest state of it. This document will contain the use case diagrams, brief and detailed use cases, system sequence diagrams and a functional overview model. The purpose of this document is to familiarise the reader with the structure, components and responsibilities of the application.

Any terminology issues can be referred to the glossary section of the research document. This is where all the terms used to describe any part of this project will be defined.

2. USE CASE DIAGRAMS

This application contains several main components which will enable it retrieve tweets and analyse the sentiment in an efficient way.

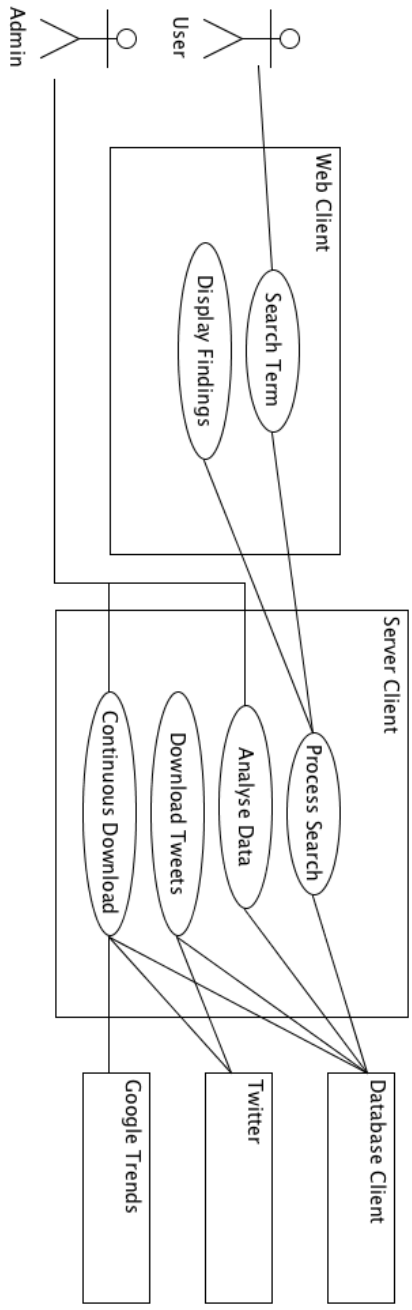

Figure 1. Use case diagram outlining the main functional components of the application.

3. BRIEF USE CASES

3.1 SEARCH TERM

Name:	Search Term
Actors:	User, Server
Description:	This use case begins when the web client finishes loading. The user is asked to enter a term which they would like to see the sentiment for. Then clicks the search button and the term is sent to the database server. The use case ends when the user clicks the search button and the term is sent through an API to the Server.

3.2 DISPLAY FINDINGS

Name:	Display Findings
Actors:	Server
Description:	This use case begins when the server returns the sentiment data for the searched term. This data is then used to render graphs expressing the sentiment in the tweets that were analysed. This use case ends when the web page is terminated.

3.3 PROCESS SEARCH

Name:	Process Search
Actors:	Web Client, Database Server
Description:	This use case begins when the web client requests data for a certain term. The server checks the database server for entries associated with the searched term. Then returns the received data from the database server. This use case ends when the web page is

3.4 ANALYSE DATA

Name:	Analyse Data
Actors:	Database Server, Server Admin
Description:	This use case begins when the server admin wants to start the analysis of the data. A request is sent to the database server to retrieve all the posts without a sentiment assigned to it. The data is then processed and a classification is assigned to it. The data is then sent to the database server and stored. This use case ends when the classification is stored.

3.5 DOWNLOAD TWEET

Name:	Download Tweet
Actors:	Database Server, Twitter
Description:	This use case begins when a term is received. The server uses Twitter's Search API to request tweets regarding the term. When the data is received from Twitter, the tweets are then stored in the database server. This use case ends when the data is successfully stored.

3.6 CONTINUOUS DOWNLOAD

Name:	Continuous Download
Actors:	Server Admin, Database Server, Twitter, Google Trends
Description:	This use case begins when the server admin starts the continuous download of tweets. The server requests the latest trends from Google trends and Twitter. Once the data is received, the server uses Twitter's Search API to request tweets about the trending topics. When the data is received from Twitter, the tweets are then stored in the database server. This use case ends when the tweets are successfully stored.

4. DETAILED USE CASES

4.1 SEARCH TERM

Name: Search Term

Actors: User, Server

Description: This use case begins when the web client finishes loading. The user is asked to enter a term which they would like to see the sentiment for. Then clicks the search button and the term is sent to the database server. The use case ends when the user clicks the search button and the term is sent through an API to the Server.

Main Success Story:

1. The user opens the web client for the first time.
2. The user is asked to input a term that they would like to see the sentiment for.
3. The user inputs the term.
4. The user presses the search button

Alternatives:

3a. The user can click one of the trends that appear on the homepage. This will submit the trend as the searched topic to the Web Client

4.2 DISPLAY FINDINGS

Name: Display Findings

Actors: Server

Description: This use case begins when the server returns the sentiment data for the searched term. This data is then used to render graphs expressing the sentiment in the tweets that were analysed. This use case ends when the web page is terminated.

Main Success Story:

1. Web client receives data from the server.
2. AngularJS uses D3 to render graphs.

4.3 PROCESS SEARCH

Name: Process Search

Actors: Web Client, Database Server

Description: This use case begins when the web client requests data for a certain term. The server checks the database server for entries associated with the searched term.

Main Success Story:

1. The web client requests data about a chosen term.
2. The server requests the data from the database server.
3. The returned data is formatted.
4. The data is returned to the web client.

Alternatives:

2a. If the term doesn't match any keywords in the database, it returns an empty list.

2b. If an empty list is returned, the download tweets use case will begin.

4.4 ANALYSE DATA

Name: Analyse Data

Actors: Database Server, Server Admin

Description: This use case begins when the server admin wants to start the analysis of the data. A request is sent to the database server to retrieve all the posts without a sentiment assigned to it. The data is then processed and a classification is assigned to it. The data is then sent to the database server and stored. This use case ends when the classification is stored.

Main Success Story:

1. The server admin wants to classify posts.
2. The server requests all the posts in the database server that don't have a sentiment associated with it.
3. Once the data is returned, the data is processed, this includes removing the stop words, convert to lower case and use n-grams.
4. The data are then run through three machine learning algorithms.
5. The sentiment and confidence of the algorithms are then decided.
6. The result is then sent to the database server and the data is updated to include the sentiment and confidence.

4.5 DOWNLOAD TWEETS

Name: Download Tweets

Actors: Database Server, Twitter

Description: This use case begins when a term is received. The server uses Twitter's Search API to request tweets regarding the term. When the data is received from Twitter, the tweets are then stored in the database server. This use case ends when the data is successfully stored.

Main Success Story:

1. The search term is received.
2. The server queries Twitter's Search API for tweets containing the searched term.
3. The data is returned to the server.
4. The id, text, timestamp, favorite counts, retweet counts and the coordinates are then extracted.
5. The URL is removed from the text.
6. The timestamp is then reformatted to Python's datatime format.
7. The search term is associated with the tweet
8. Repeat steps 4 to 7 for every tweet
9. The formatted tweets are then sent to the database server.

4.6 CONTINUOUS DOWNLOAD

Name: Continuous Download

Actors: Server Admin, Database Server, Twitter, Google Trends

Description: This use case begins when the server admin starts the continuous download of tweets. The server requests the latest trends from Google trends and Twitter. Once the data is received, the server uses Twitter's Search API to request tweets about the trending topics. When the data is received from Twitter, the tweets are then stored in the database server. This use case ends when the tweets are successfully stored.

Main Success Story:

1. The server admin starts the continuous download of tweets by running the download script.
2. The server requests the latest trends from Twitter and Google trends.
3. The trends from Ireland, England and America are extracted.
4. The server iterates through the list of trends
5. The server queries Twitter's Search API for tweets containing the searched term.
6. The data is returned to the server.
7. The id, text, timestamp, favorite counts, retweet counts and the coordinates are then extracted.
8. The URL is removed from the text.
9. The timestamp is then reformatted to Python's datatime format.
10. The search term is associated with the tweet
11. Repeat steps 5 to 10 for every tweet
12. The tweets are then sent to the database server to store the data.
13. Repeat steps 4 to 12 for all the trends

5. USER INTERFACE DESIGNS

The user interface (UI) will implement the simplistic but elegant approach to design. There will be two main screens. The first being the search screen which will allow the user search a term which will be sent to the server and processed. This screen will have a search box and a list of current trends. Figure 2, shows a wireframe of the proposed screen.

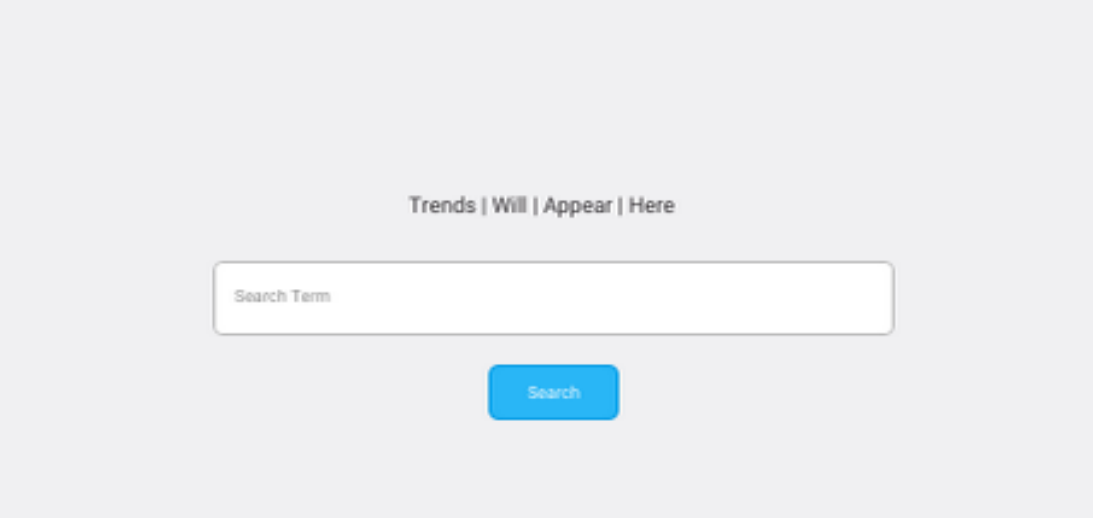

Figure 2. This is a wireframe illustrating the main search screen.

The next screen will display when the results have been sent back to the web client. This data will be represented in several graphs to show the sentiment the community of Twitter has about certain terms. This screen is proposed to implement the simplistic design as well. This screen will need to be self-explanatory to maintain the simplistic design. It's proposed to have minimal text on the screen, limited to a global legend so the user will be able to recognize the what the colour's in the graphs. This can be seen in figure 3.

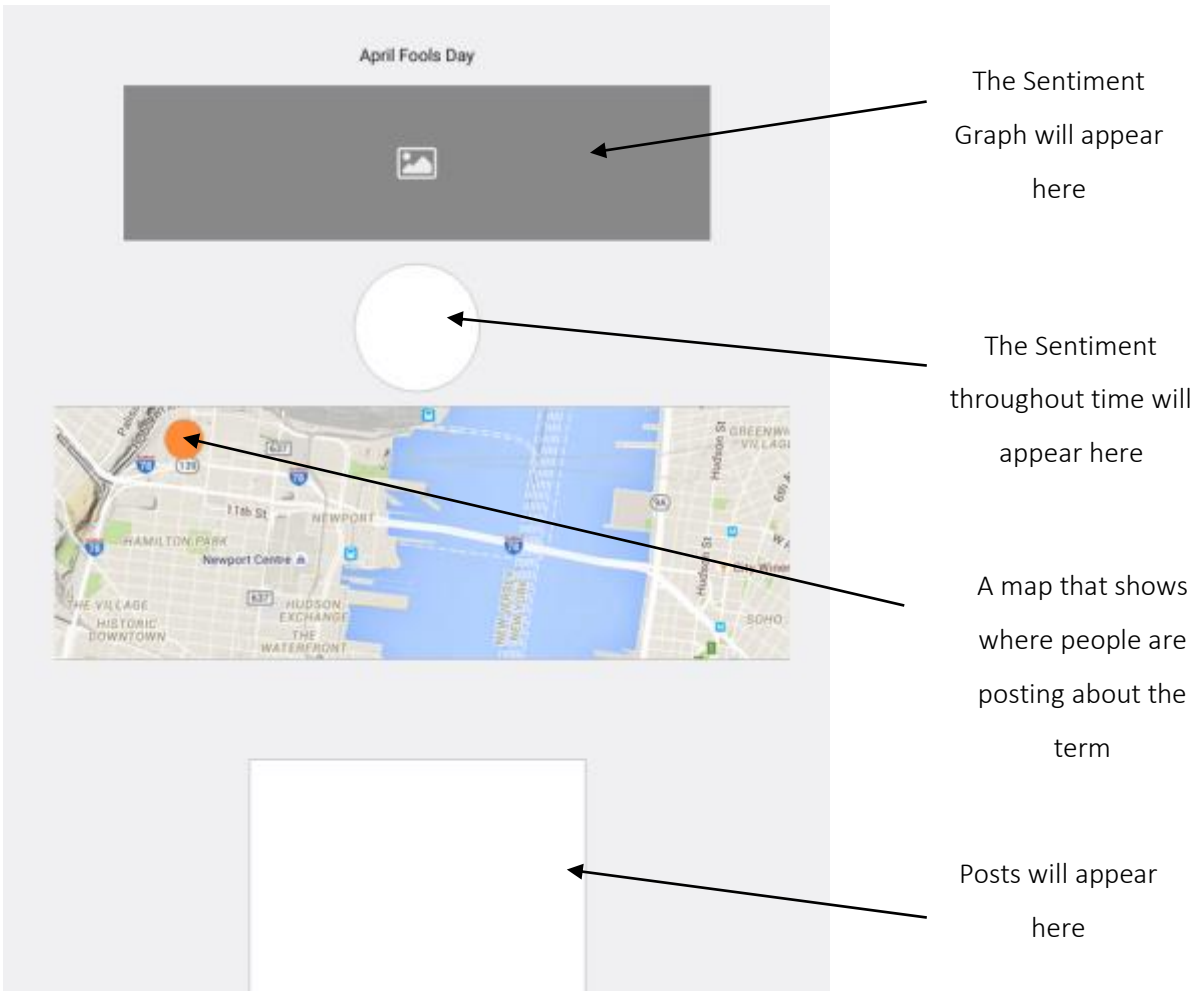

Figure 3. A wireframe illustrating the analysis result page

This screen contains several main aspects of the results. The header is the searched term, in this case the term being searched was “April Fools Day”. The next component is a graph. This graph will reveal the sentiment throughout time. The circle is a pie chart that contains shows the total split of the sentiment breakdown for the entire time that the tweets are shown for. The map is the next element which illustrates where people are discussing the searched term. The orange circle on the map marks people’s locations, this won’t be exact locations but a rough guide to give the user an idea of where people are tweeting from. The final element of the page is a box containing all the categorized tweets that the sentiment is shown for.

6. SYSTEM SEQUENCE DIAGRAM

6.1 SEARCH TERM

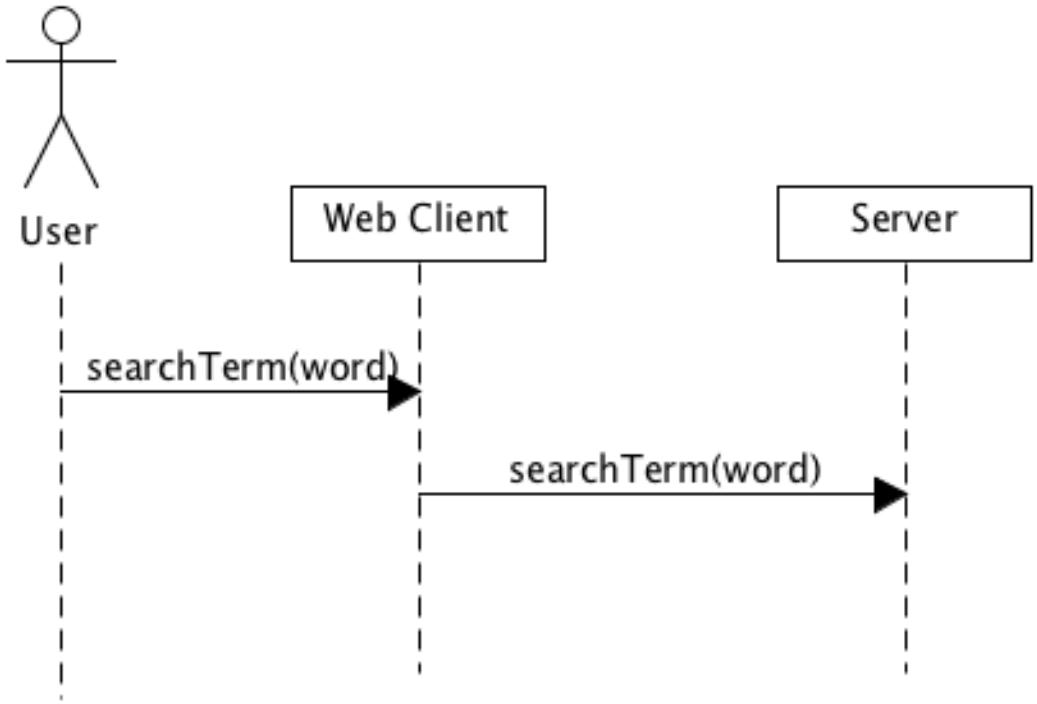

Figure 4. The System Sequence Diagram for how the user searches a term and retrieves tweets.

6.2 DISPLAY FINDINGS

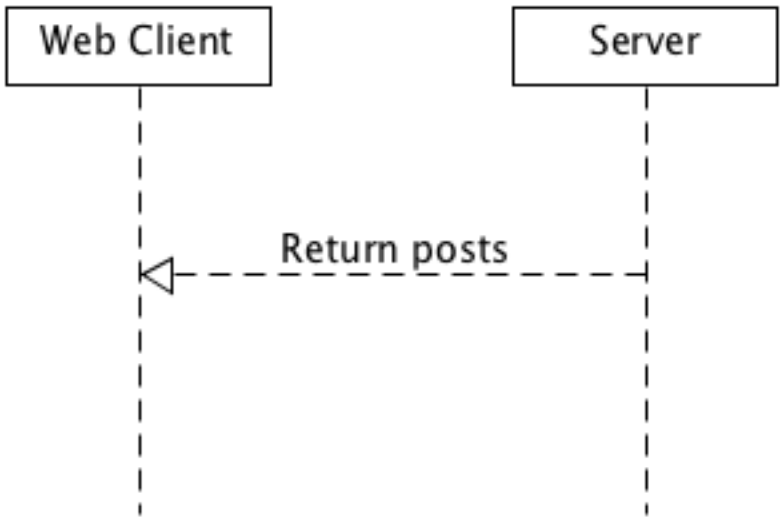

Figure 5. The display findings System Sequence Diagram.

6.3 PROCESS SEARCH

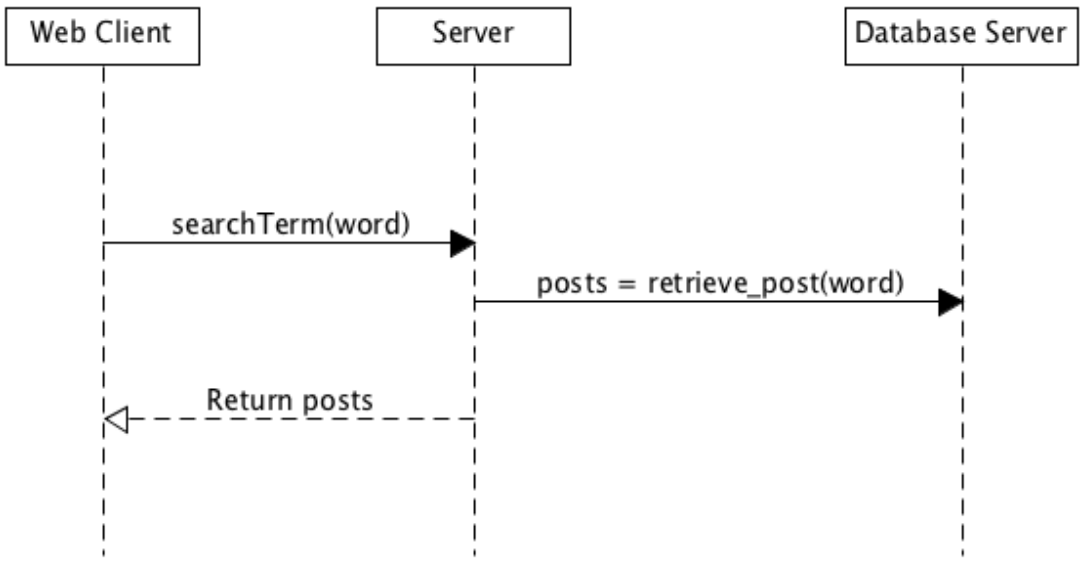

Figure 6. The system sequence diagram for how posts are retrieved from the database.

6.4 ANALYSE DATA

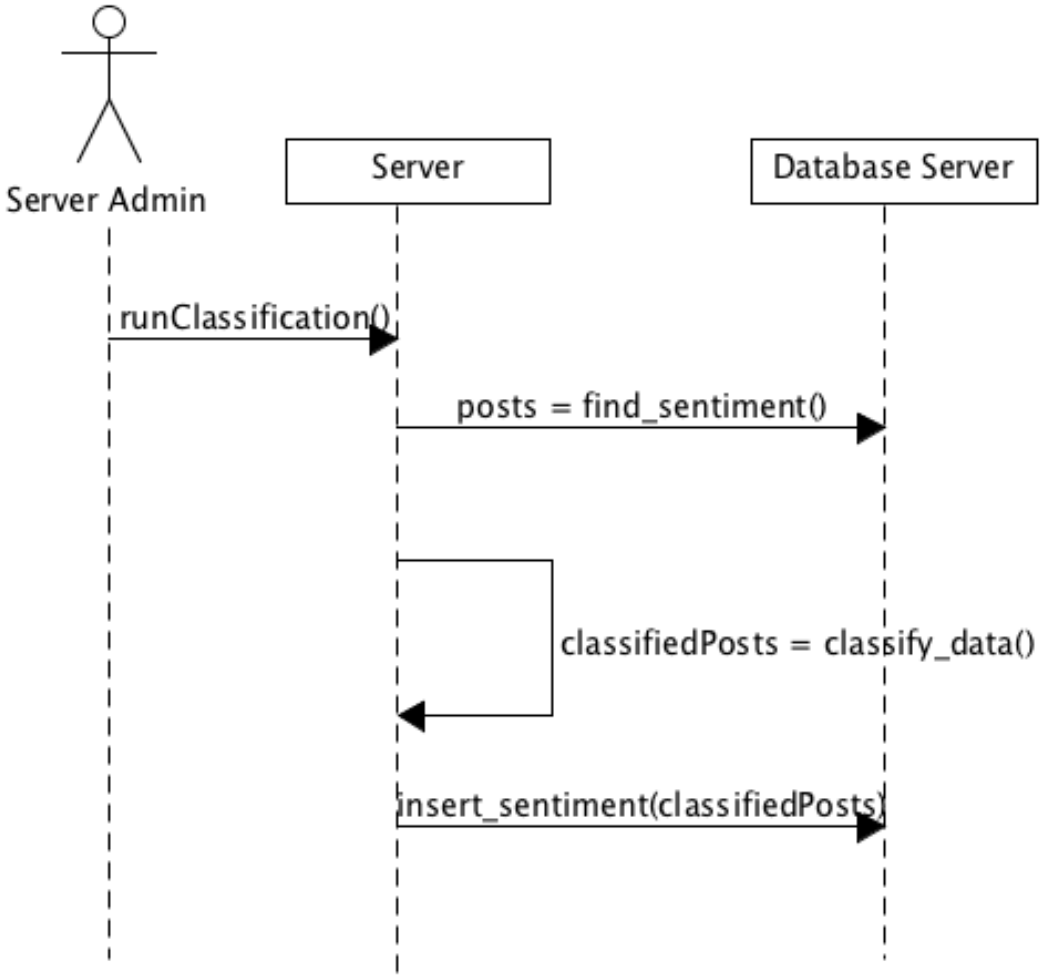

Figure 7. The system sequence diagram of how posts are analysed and a sentiment associated with a post.

6.5 DOWNLOAD TWEETS

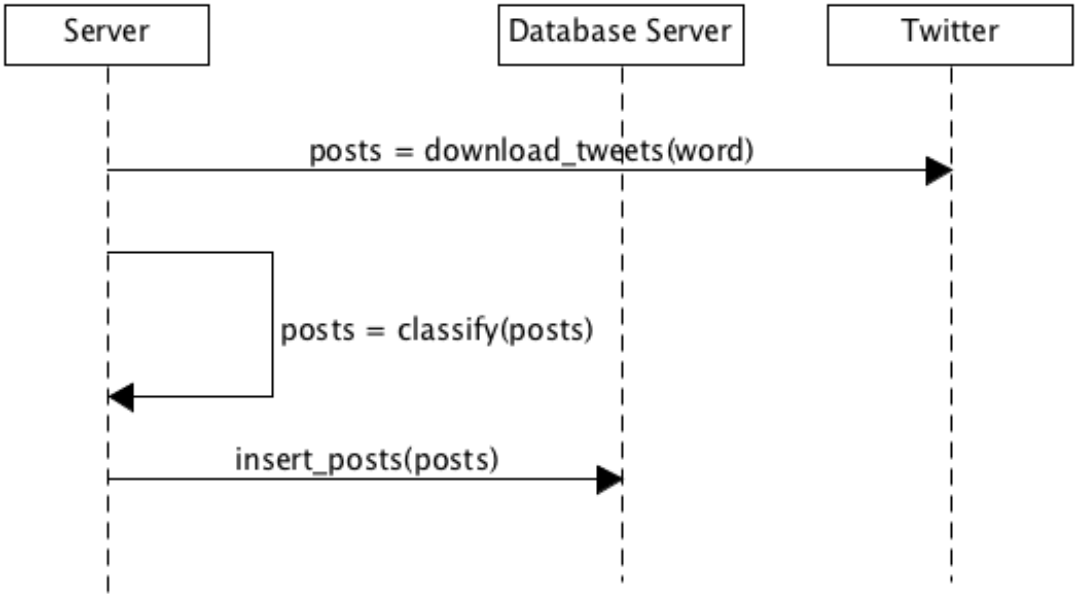

Figure 8. The system sequence diagram for the downloading of tweets.

6.6 CONTINUOUS DOWNLOAD

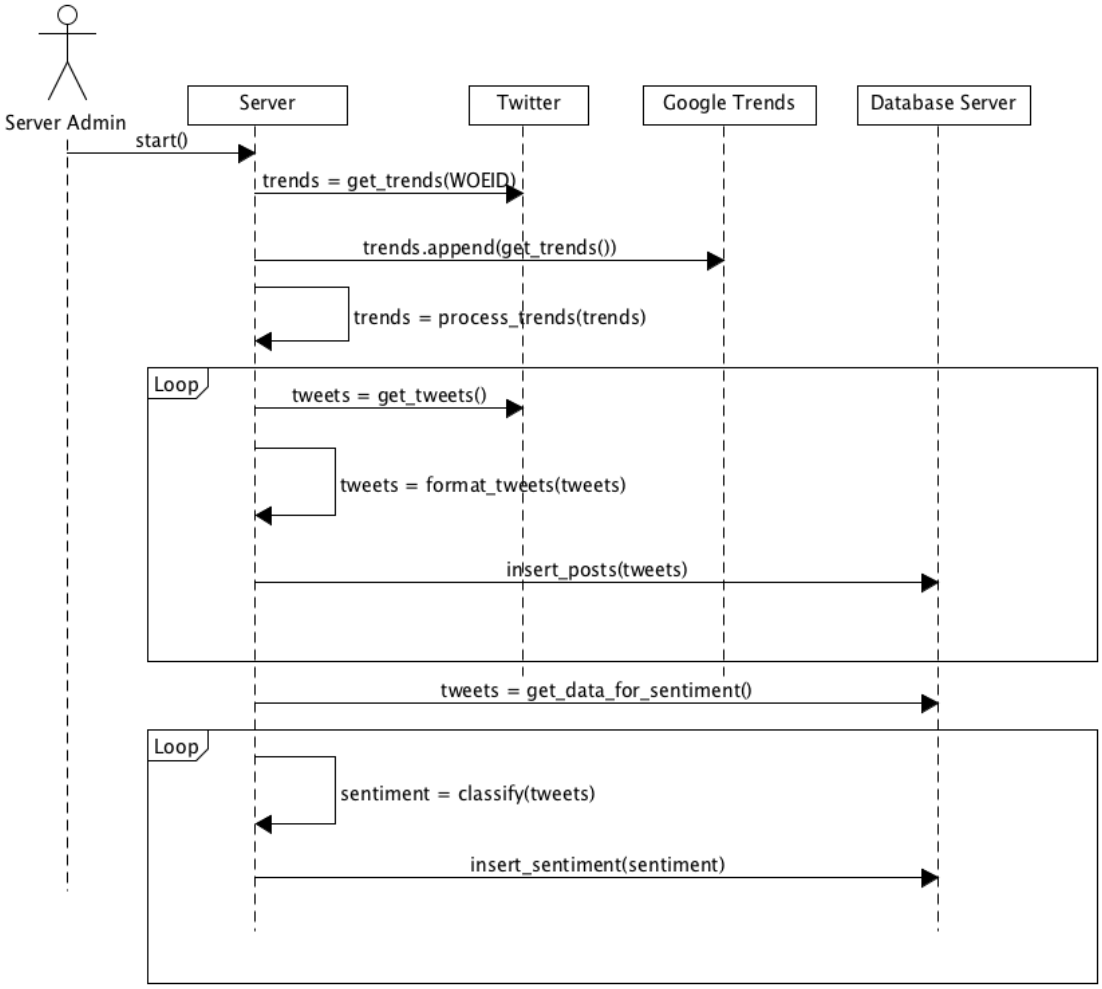

Figure 9. The System Sequence Diagram for how the server admin starts the continuous download of tweets.

7. GENERALISATION OF TWITTER SENTIMENT ANALYSIS

One of the main aspects of this project that separates it from its competitors is the generalisation of the code. This enables it to be used for much more than just a Twitter sentiment analysis tool. This section will outline the aspects that allow it to be a generalised machine learning analysis tool.

7.1 PLATFORM TO BE ANALYSED

The first aspect is the ability to use multiple platforms for the sentiment analysis. This allows for a more varied opinion that the user can base their analysis off. Figure 10 shows the files that should be needed to add or remove a platform.

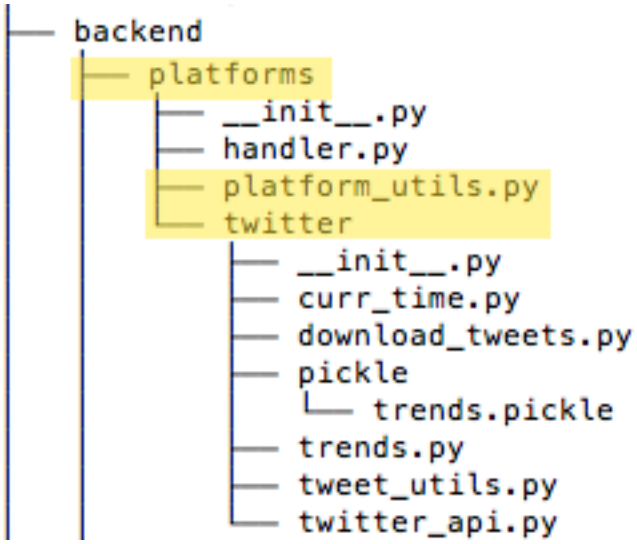

Figure 10. The files and folders that need to change to add or remove a platform.

The first element in figure 10 is a directory called '`platforms`'. The developer will need to add a folder including all the files necessary to process and download the posts off the new platform. The next element is a file called '`platform_utils.py`'. This file contains the name and download methods of the new platform. This includes the function needed for the continuous download use case and a single download if there aren't any entries in

the database that match the chosen keyword. As shown here, Twitter is the only platform implemented.

7.2 POSSIBLE CLASSIFIERS

The second aspect is the ability to change and edit the classifiers needed for the correct classification. When machine learning algorithm parameters are modified, it can potentially have a profound effect in the algorithms accuracy. For this reason, it was decided to allow the developer to change and modify the algorithms used with ease.

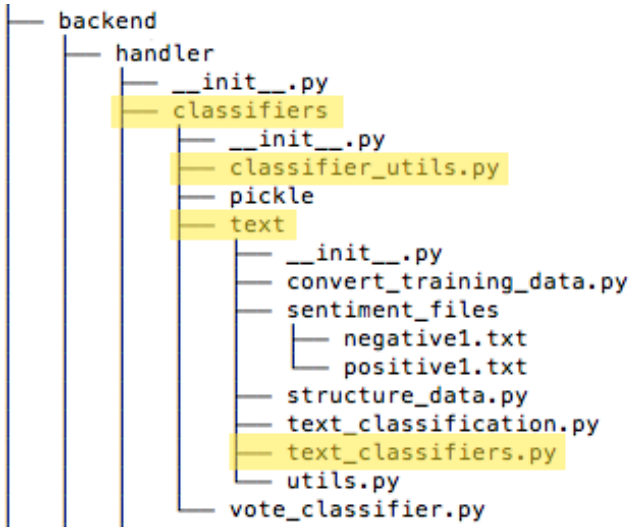

Figure 11. The files and folders that need to change to add or remove a platform.

The classifiers can easily be changed using the files and folders mentioned in figure 11. This shows within classifiers folder, there's a file called '`classifier_utils.py`'. This file contains all the functions needed to train, test and use cross validation. The classifiers can be added within the desired folder, in this case it's called '`text`'. The classifiers are placed into a Python dictionary where the classifier name is the key and the actual algorithm is the value. This enables the developer to have full access to enhance the capabilities of the performance of the algorithms. In this example, the classifiers are placed in the '`text_classifiers.py`' file which is then included in '`classifier_utils.py`' file.

7.3 ANALYTICS TYPE

The third aspect of this project which separates it from its competitors is the ability to add different types of analysis. This application is implemented using text analysis which is required for sentiment analysis. This type of analysis can also be used for social media monitoring, email routing and spam filters, for example. Although, text analysis can be very useful, this application allows the developers to change the machine learning algorithms which can allow this application to be used for many different types of analysis other than just text.

This means other types of analysis are available to the developer. For example, image analysis can be used for different types of filtering and security. In figure 11, a folder called '`text`' contains everything necessary to complete text classification. This allows the developer to add a new folder containing all the necessary elements needed to successfully complete another type of classification. Keeping all the necessary functions in a file called '`classifier_utils.py`' which can be used throughout the project.

As mentioned in the Classifiers section, the developer creates their own file which enables them to maintain the algorithms for that type of analysis. This enables the developer use the same code base for several different types of analysis. This means they can analyse text and images at the same time with very little extra code.

7.4 CLASSIFICATION

The users of this application might want to express different sentiment. For example, this application is implemented using the positive and negative sentiments, however, the users might prefer to add the neutral sentiment or add very positive and very negative sentiments to this application. For this reason, having the classifications adjustable is a vital aspect of this application.

Another aspect of this feature enables developers to submit different types of classifications. This feature is essential to allow the developer to create other tools like a spam filter. Having this feature greatly increases the reusability of the code base and allows the possibilities of the uses of this application to be increased.

7.5 APPLICATION PROGRAMMING INTERFACE

This application has an Application Programming Interface (API) already implemented for the sentiment analysis frontend. Depending on the decision made in the Classification section, the API will adjust to suit the new sentiments added, removed or renamed. This allows the current API to be used for a multitude of different classification projects.

8. FUNCTIONAL OVERVIEW MODEL

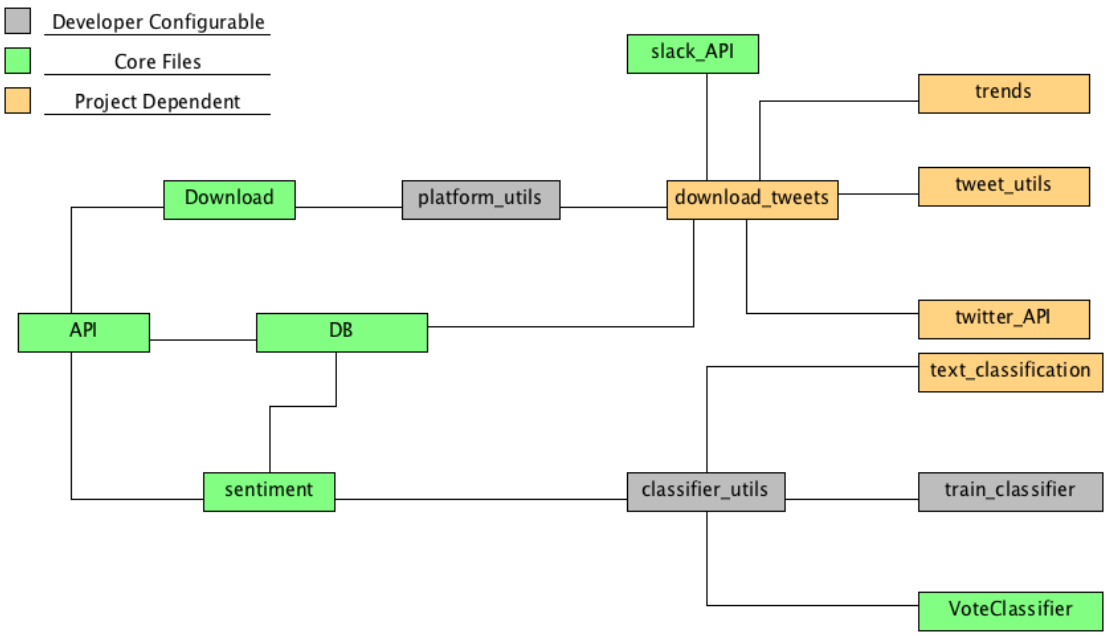

Figure 12. The relationship between the primary files and their purpose.

In figure 12, the files are classified into three main sections, Developer Configurable, Core Files and Project Dependent. This diagram illustrates the relationship between the different files and sections. The first type of file is the developer configurable files. These files contain the information that the developer must submit for their changes to be implemented into the application. An example of this is the methods of how the tweets are downloaded from Twitter.

The next section is Core Files. These are files that should not be changed unless necessary. They contain the core functionality of the application. The example of these files can be found in the API file where there is a standardised way of returning the data to the web client.

The final section is the Project Dependent files. These files are different depending on the project specifications. This project for example, required the use of Twitter. Due to this requirement, the download of tweets file exists. If another platform is required, then it will appear on the same level as download tweets.