**Experimental Setup**

**Implementation Details**

Back-end

**Front-end**

The Geospatial Sentiment Analysis is a web-based product, and the front-end is experimental setup by computer language html, CSS, and JavaScript. The Geospatial Sentiment Analysis system is available to run on any PC accompanying with well-setup back-end environment which installed Flask, Tweepy and Textblob. To access the webpage at local PC with html file to perform the sentiment analysis based on area.

The implementation details can be divided mainly to 5 units based on the functions, which are GUI, API calling, received data formatting, Map marking and tweets content displaying.

1. GUI

The elements (input bar, button, navigation bar, etc) of webpage is directly hypertext reference Bootstrap4 (the world’s most popular front-end component library) stylesheet. All the components are placed vertically to obtain a simply and organized layout.

For the displaying the map in website the script loads below API from the specific URL together with a callback function called initMap() to initialize the map.

    <script async defer  
    src="https://maps.googleapis.com/maps/api/js?key=YOUR\_API\_KEY&callback=initMap">  
    </script>

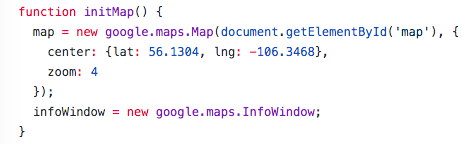


Fig. Function initMap()

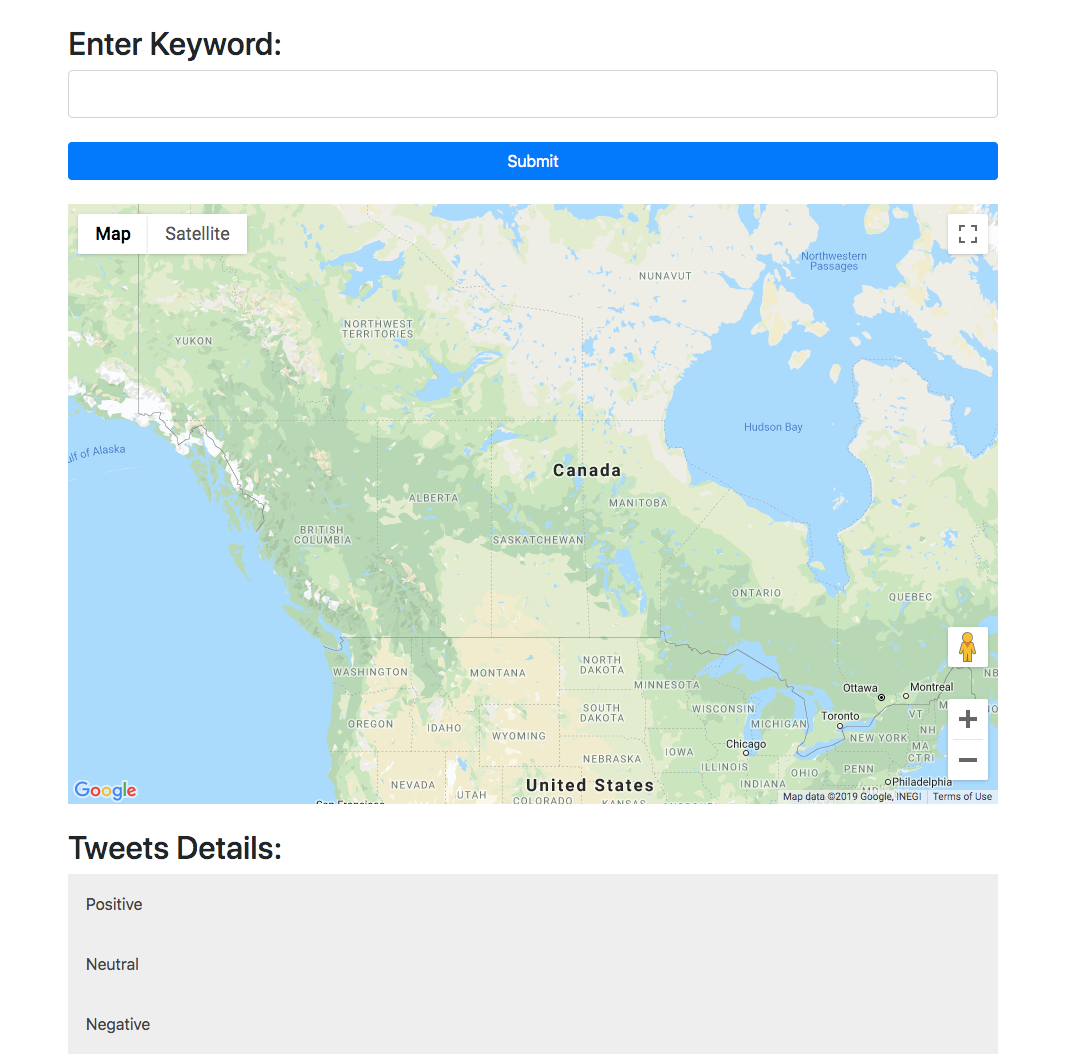


Fig. webpage layout

1. API calling

API calling is designed to retrieved JSON data from local back-end through URL by using Ajax GET request. There are 2 interfaces:

① /search?keywords=xxx

② /zoom?keywords=xxx&longitude=xxx&latitude=xxx

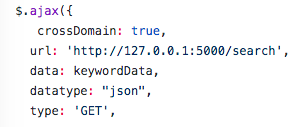


Fig. Ajax GET request

For the first interface (refer to Fig. ), the API calling program will provide keywordData with one parameter keyword, and for the second interface (refer to Fig.), the parameter will provide not only keyword, but also centre of map’s longitude and latitude to fulfil the zooming scenario and the default radius is 10 km.



Fig. Interface 1 parameter

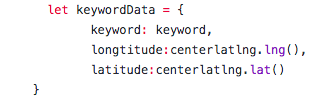


Fig. Interface 2 parameters

1. Received data formatting

The response data returns from API is in JSON format. In order to extract the value separately for further using, the data will be formatted to Array by using JSON.parse(). The formatted data refer to Fig.



Fig. Formatted data

1. Map marking

To implement marking in Google Map, google.maps.Marker() helps placing the custom makers in Google map.

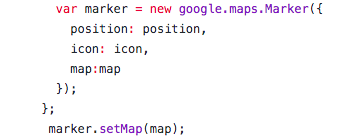


Fig. Google Marker

The system identifies the sentiment into three categories, which are positive, neutral and negative. The rule to decide which the data belongs to are as follows:

Positive: positive percentage 60%

Neutral: 40% < positive percentage < 60%

Negative: positive percentage 40%

The display with marker in Google map refers to fig.

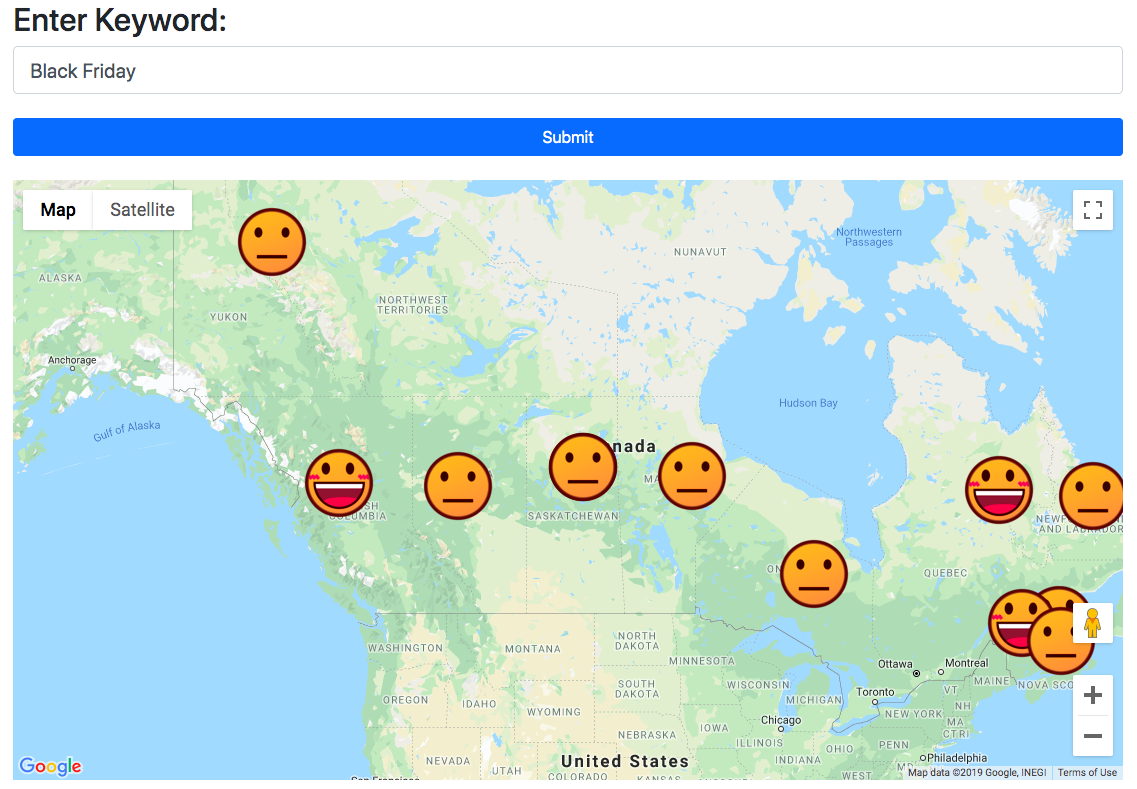


Fig. Map marking

1. Tweets content displaying

To show the detail tweets in webpage, extract the content from response data which already after JSON.parese(). Using these contents to link to component in the webpage using innerHTML refer to Fig , along with divided to 3 categories. The display refer to Fig.

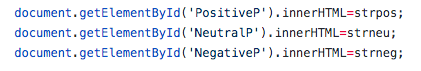


Fig. innerHTML

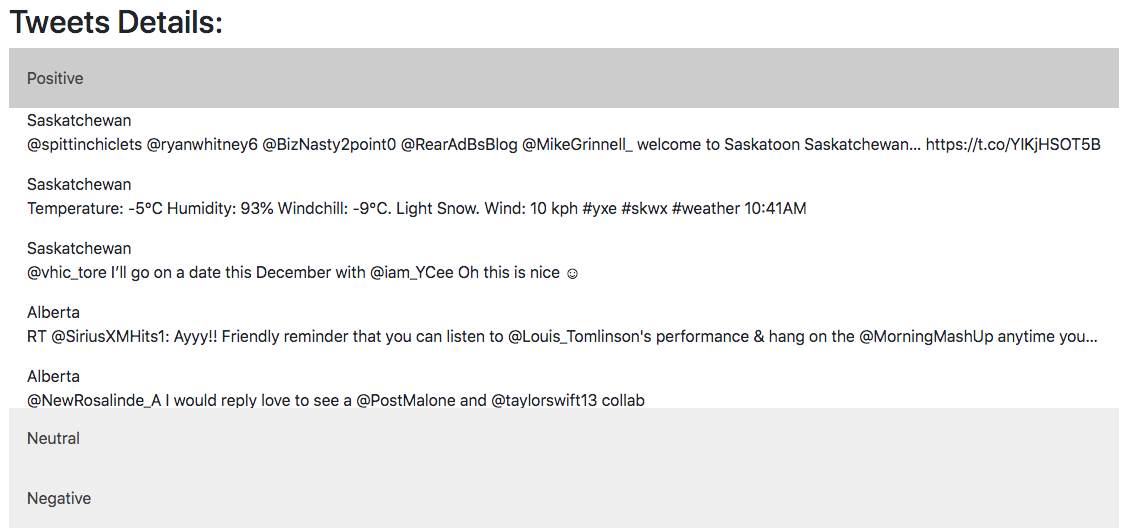


Fig. Tweets content

**Testing**

In the testing, it will divided into 2 parts (back-end and front-end) to run the testing which can be seem as TAT (Technical Acceptance Test). After confirming each part run well to do the completed service testing which is the UAT (User Acceptance Test), including marking on the map, showing tweets contents. For more detail test case please refer to appendix.

**Back-end Testing**

For the back-end, black box test is performed. There are two parts, one is data extraction, and the other is sentiment analysis.

**Data extraction part**

In the data extraction part, Tweepy is the API using to collect data from twitter platform. First, the back-end environment need to setup and verify the whether the Tweepy API can work in local computer. Since the API is provided by third-party, it assumes API should not have bugs which will affect the data extraction from Twitter. So doing the black box test is regarded as a normal procedure to verify whether the data can be gather from Twitter and whether the data generated exactly from Twitter or any other faults like garbled message, etc. In order to make sure the back-end environment can run smoothly in local environment to prepare for further execution.

**Sentiment analysis part**

For the sentiment analysis part, TextBlob API is choose to process the twitter data and provided analysis results. The first test must be installation in local computer to make sure it can be implemented in local computer. And then the verification of the accuracy of sentiment analysis should be taken into consideration. As this is the third party API, the black box testing will be performed to check the algorithm fulfil the expectation (analysis accuracy should not less than 70%) or not. For the basic and simple test, the sample tweets will use an input to verify the analysis algorithm meets the basic requirement of textual classification.

**Front-end Testing**

The testing in front-end will perform unit test. There are five functions in front-end, which are GUI displaying, API request, data formatting, map marking and tweets content displaying. By running the test separately to each function to make sure every part of web program works without any fault which would influence the process of the system. Since there is a dependency among these function, in order to decoupling the relationship, manually provide input data will be performed to guarantee each part work correctly before coupling together.

**End-to-End Testing (Service Test)**

The intention of service testing is to do the end-to-end testing to confirm the requirements implement successfully. The requirements of the application are

① Pointing the sentiment emojis (smile, neutral, sad) in specific location of the map

② Showing the Tweets content as references and categorizes the message

As there are two interfaces in this application like Fig. , a decision (branch) coverage will perform to cover all the path:

Test case 1: P1, P3, P4

Test case 2: P2, P3, P4

Marking in Google Map

v

Interface1: parameter: keyword

Interface2: parameter: keyword, location

Display Tweets content

P1 P2

P3

P4

Fig. Decision (Branch) Coverage

**Findings**

Naïve Bayes algorithm to perform the sentiment analysis is the historical and theoretical machine learning method. It pre-processes the textual message based on words occur frequencies to find the sentiment words. By the classifier training section with the samples which generated from the pre-process section to calculate the percentage of each sample sentiment outcome. After finishing the training, the test sample can input to the classifier to calculate the sentiment results.

**Challenges**

The challenges of this application are in two aspects, one is internal challenges and the other is external.

Internal challenges:

1. Testing coverage. It cannot be denied that the test case in this application is hard to design comprehensively, for example: stress test, which will not be included in this system as when a high frequency request to back-end to retrieve data, third-party API may detect as an robot and block the connection.

2. Front-end development. With zero experience of front-end development, the webpage will be designed as simple as possible to cover the function, instead of considering the user-friendly viewer.

External challenges:

1. Third-party API calling allowance. As Tweepy API need to apply allowance leading to a potential risk that the application will not be accepted.

2. Hard to find perfect sentiment analysis algorithm. As there are so many options provided online, such as using naïve Bayes to do the natural language processing, etc.

3. Competitors in sentiment analysis market. As the market pay attention to mine the potential commercial value, by doing this can enlarger the chances to take more market share in their target area. And it do have more powerful and useable online application to do this job. For example , the link <https://www.csc2.ncsu.edu/faculty/healey/tweet_viz/tweet_app/>

**Appendix**

Test case

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Item No. | Test case | Expected result | Test result | Pass/Fail |
| 1 | back-end Tweetpy | | | |
| 1.1 | Extract data from source Twitter | Able to get tweets from Twitter | Get information in Twitter | Pass |
| 1.2 | Extract data from source Twitter with specific keyword | Message related to keyword is extracted | Data related to keyword retrieved | Pass |
| 1.3 | Extract data from source Twitter with specific keyword and location | Message related to keyword and this location is extracted | Data related to keyword in specific area returned | Pass |
| 1.4 | Call Tweepy API 15/min | Return 15 times data | tweepy.error.TweepError: Twitter error response: status code = 403 (Blocked by Twitter) | Fail |
| 2 | back-end TextBlob | | | |
| 2.1 | Input sentiment sentences contain happy | The value of positive should over 0.5 | Positive > 0.7 | Pass |
| 2.2 | Input sentiment sentences contain nightmare | The value of positive should over less than 0.5 | Positive < 0.3 | Pass |
| 2.3 | Input sentiment sentence contain fine | The value of positive should no less than 0.4 and larger than 0.6 | Positive fits the expected range | Pass |
| 3 | back-end JSON data encoding | | | |
| 3.1 | Successfully format analysed result to JSON format | Data in Format JSON | Data in Format JSON | Pass |
| 4 | Front-end Web | | | |
| 4.1 | GUI display contain component: input bar, button, map, list | Input bar, button, map, list place vertically | Input bar, button, map, list place vertically | Pass |
| 4.2 | Front-end successfully call API with one parameter keyword | JSON data will return | JSON data return | Pass |
| 4.3 | Front-end successfully call API with one parameter keyword and location when map zoom | JSON data will return | JSON data return | Pass |
| 4.4 | No response when map zoom without any keywork | No response | No response | Pass |
| 4.5 | Format received JSON data into object | The value can be read separately | Each parameter for an object, can print through Array | Pass |
| 4.6 | Emoji display in the map with right emotion by checking averagePositive value to verify the state whether print the right emoji | Based on the averagePostive, each State mark with the right emoji | The emoji marks right after checking the parameter value (location, averagePositive) | Pass |
| 4.7 | Tweets display in the right categories in webpage with location name and text content | Based on value in tweets.positive to place the tweets content one by one in the categories it belongs to | Result display correctly | Pass |