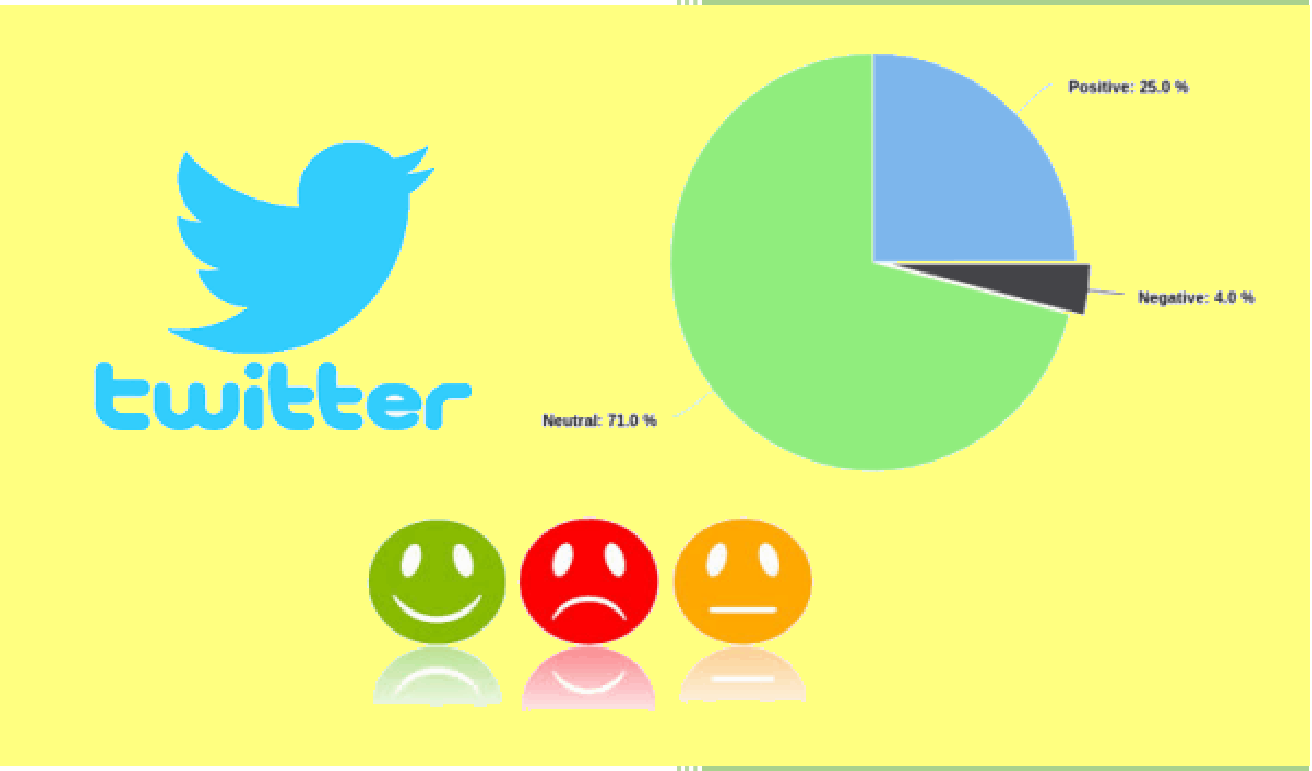
2021

Twitter Sentiment Analysis



CAB432 Assignment 2

Sy Tri Ha (Patrick) | N10273913

Minh Tuan Hoang (Shaun) | N10537848

1/11/2021

Contents

Introduction

Purpose & description

In era 4.0, Twitter and other social media platforms attract users to exchange and broadcast information, particularly every day up-to-date happenings. As a result, a large amount of data has been generated by users day-by-day, which can be utilized for multi-purposes. Each topic on Twitter is well-labelled with a relevant hashtag or specific topic, which aids in capturing all of the debate about that matter.

The application aims to extract those potential data around a chosen topic and analyze the sentiment. It performs the average sentiment score to examine trends toward the relevant topics. For clarification, a proportion sentiment chart is displayed. Moreover, the app also includes a direct button to the topic, which helps the user to start exploring.

Services used

Twitter Standard Search API (v.1.1)

Returns a collection of relevant Tweets matching a specified query. In our project, this will be used to extract 100 texts relevant to a topic that users want to discover.

Endpoint: `https://api.twitter.com/1.1/search/tweets.json`

Docs: https://developer.twitter.com/en/docs/twitter-api/v1/tweets/search/api-reference/get-search-tweets

Twitter Standard Trends API (v1.1)

An API to return the trending topics near a specific latitude, longitude location. In our application, it will be based on the unique WOEID (Where on Earth Identifier) which is given in the application to gain the topic list.

Endpoint: `https://api.twitter.com/1.1/trends/available.json`

Docs: https://developer.twitter.com/en/docs/twitter-api/v1/trends/locations-with-trending-topics/api-reference/get-trends-available

Twitter Standard Nearby Trends API (v1.1)

A button to sort out some closest trending topics with their location is included by using this API, which retrieves a full or nearby locations list of trending topics by locations.

Endpoint: `https://api.twitter.com/1.1/trends/closest.json`

Docs: https://developer.twitter.com/en/docs/twitter-api/v1/trends/locations-with-trending-topics/api-reference/get-trends-closest

Natural Sentiment Analysis

Natural is a NodeJS facility that allows using natural language in general. In terms of the application, the sentiment analysis function from this extension is utilised to calculate the sentiment score. By accumulating the polarity of each word and normalising with the length of the sentence, the algorithm evaluates the sentiment of a piece of text.

Docs: http://naturalnode.github.io/natural/sentiment_analysis.html

S3 bucket from AWS

Long-term storage for saving data that is extracted from Twitter API. All of the data that has been extracted will be saved here. For the second use of this technology, the application just needs to take *from the S3 bucket instead of fetch from Twitter API, which helps the extracting data process faster. Trends will play as the key for the sentiment result of it, they will be saved to be S3 Bucket.*

Elasticache for Redis

Amazon Elasticache is a cloud-based web service that makes it simple to create, administer, and scale a cache. It offers a scalable, high-performance, and cost-effective caching solution. Simultaneously, it reduces the difficulty of deploying and maintaining a distributed caching environment. Existing applications that use Redis can nearly immediately switch to Elasticache. One benefit of Elasticache is that it will provide a central Redis that hold all of the short-term data, which ensures the data of each instance is continuous and linking together. The Elasticache will be used in our project along with S3 Bucket for accessing data in a short time. The trends will be saved to this Elasticache.

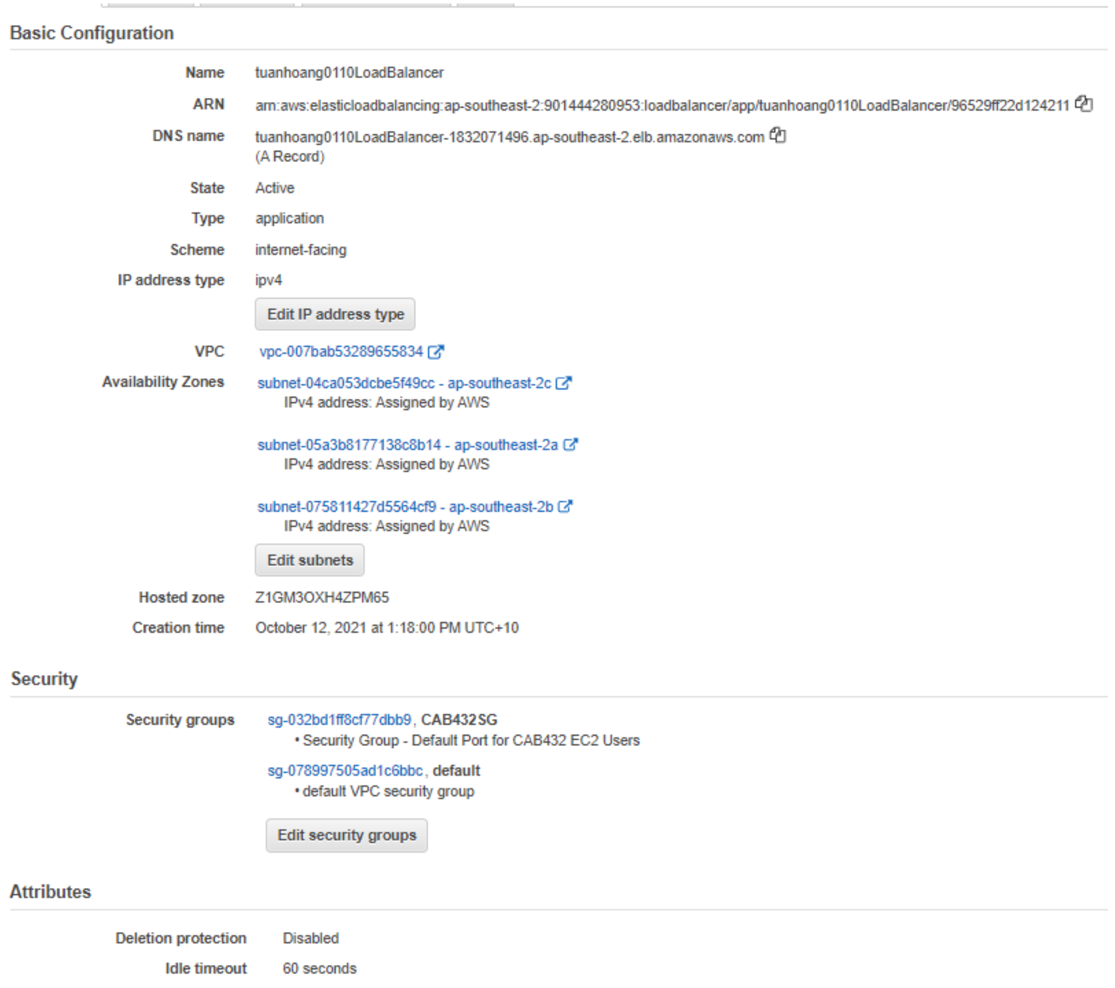
React

React is a user interface library written in JavaScript. It is the web application's view layer. With the outstanding feature that React can provide, it is utilized to develop the app's client-server.

Chart.js

In era 4.0, data may be used to tell compelling stories. Few libraries can help the developer produce remarkable results with relatively little effort to make it visualize in their application. Chart.js is one of those. It is a community-maintained open-source library that makes data visualisation simple in JavaScript. The library is applied to the project to display the percentage of sentiment within a chosen topic, which helps users filter the topic they want to discover further.

AWS Load Balancing



Basic Configuration

Name	tuanhoang0110LoadBalancer
ARN	arn:aws:elasticloadbalancing:ap-southeast-2:901444280953:loadbalancer/app/tuanhoang0110LoadBalancer/96529ff22d124211
DNS name	tuanhoang0110LoadBalancer-1832071496.ap-southeast-2.elb.amazonaws.com (A Record)
State	Active
Type	application
Scheme	internet-facing
IP address type	ipv4
VPC	vpc-007bab53289655834
Availability Zones	subnet-04ca053dcbe5f49cc - ap-southeast-2c IPv4 address: Assigned by AWS subnet-05a3b8177138c8b14 - ap-southeast-2a IPv4 address: Assigned by AWS subnet-075811427d5564cf9 - ap-southeast-2b IPv4 address: Assigned by AWS
Hosted zone	Z1GM3OXH4ZPM65
Creation time	October 12, 2021 at 1:18:00 PM UTC+10

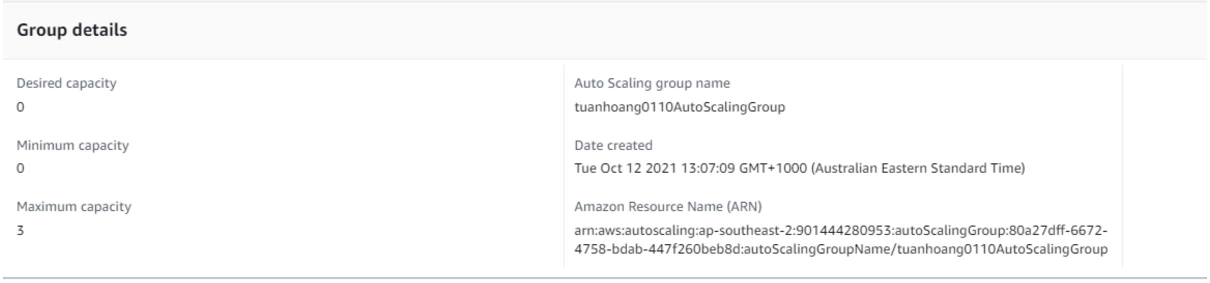
Security

Security groups	sg-032bd1ff8cf77dbb9, CAB432SG • Security Group - Default Port for CAB432 EC2 Users sg-078997505ad1c6bbc, default • default VPC security group

Attributes

Deletion protection	Disabled
Idle timeout	60 seconds

AWS Auto Scaling Groups



Group details

Desired capacity
0

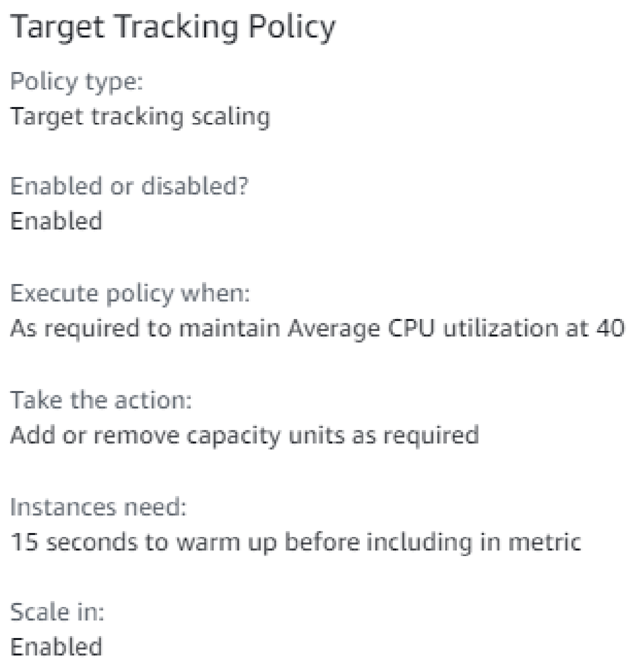
Minimum capacity
0

Maximum capacity
3

Auto Scaling group name
tuanhoang0110AutoScalingGroup

Date created
Tue Oct 12 2021 13:07:09 GMT+1000 (Australian Eastern Standard Time)

Amazon Resource Name (ARN)
arn:aws:autoscaling:ap-southeast-2:901444280953:autoScalingGroup:80a27dff-6672-4758-bdab-447f260beb8d:autoScalingGroupName/tuanhoang0110AutoScalingGroup



Target Tracking Policy

Policy type:
Target tracking scaling

Enabled or disabled?
Enabled

Execute policy when:
As required to maintain Average CPU utilization at 40

Take the action:
Add or remove capacity units as required

Instances need:
15 seconds to warm up before including in metric

Scale in:
Enabled

Use cases

US 1

As a	Twitter user
I want to	See the daily Twitter trending topic (or hashtag)
So that I can	Know which up-to-date information that I should follow or suggestions for topics that I might be interested in and want to find out.

US 2

As a	Twitter user and data curious person
I want to	Know the average sentiment score and viewing the sentiment proportion chart of it
So that I can	Filter the topic that I want to read and debate.

US 3

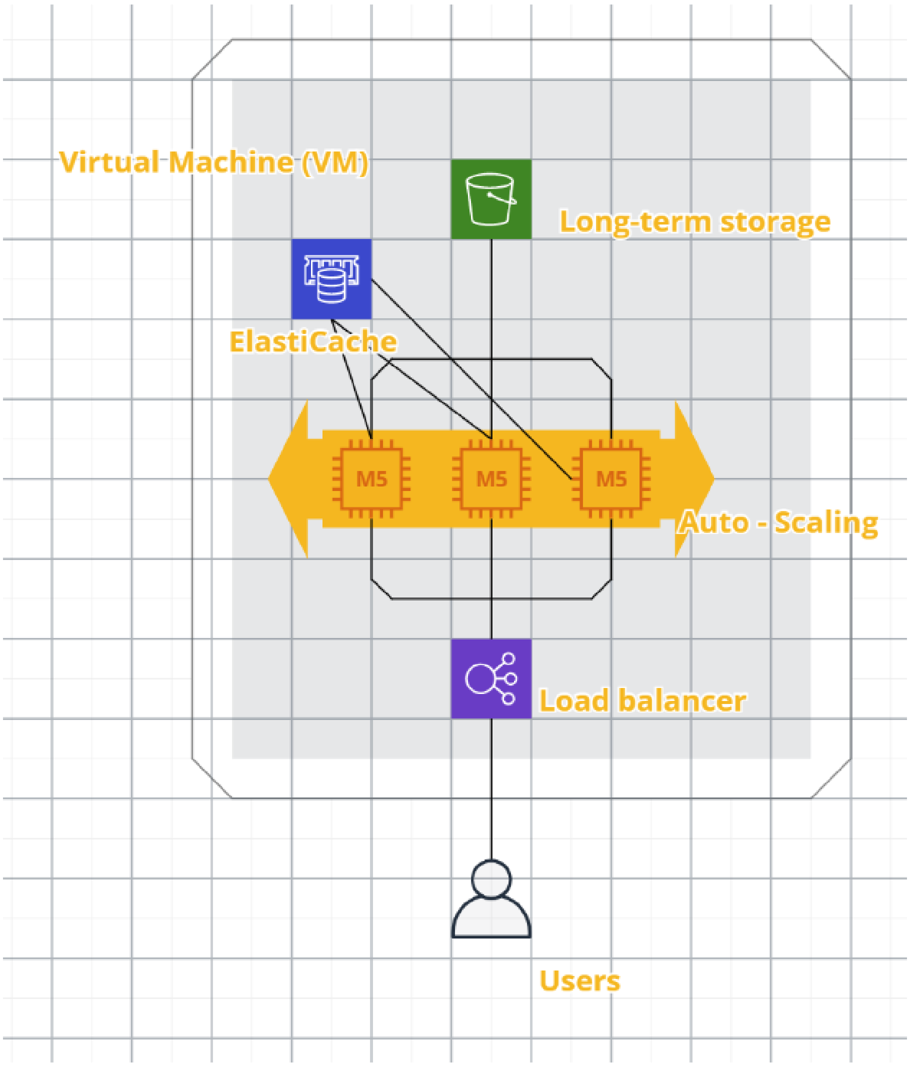
As a	Twitter user
I want to	Directly jump to my interesting topic
So that I can	Start to explore the topic.

Technical breakdown

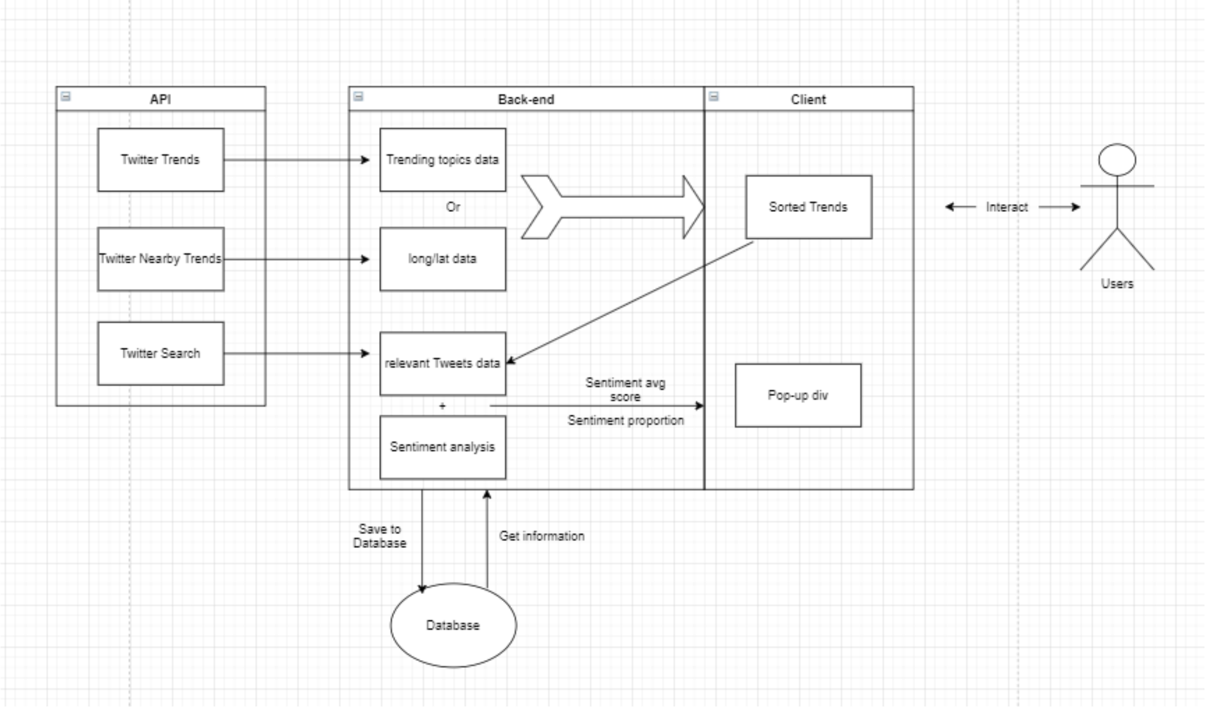
The Twitter Sentiment Analysis web application uses the **React-Js** library to build the web application view's layer. This application is being inspired by a YouTube tutorial which aims to use and display trends in Twitter in specific places using Twitter API. The layout of the project might be found similar to the tutorial video. This application applied **React State** and **Contexts** to manage the state of variables. **Axios** module is chosen as HTTP request handler because of its supports for promise API, JSON formatter, and older browsers. Besides, our Back-End server built on a basic **Express** framework will be responsible for calling requests to APIs (also use **Twitter npm** – an asynchronous client library for the Twitter REST) and serving prepared APIs to Front-End clients through custom routes. In this back-end server, **Elasticache** (short-term storage) and **S3 Bucket** (long-term storage) are applied to save the extracted data from the Twitter API, which aims for quickly processing data. To enable cross-site communication between the FE client and BE server, we added the **cors** module to the application. Finally, the app will be containerised using **Docker** and deployed via AWS to make it visible to users. The **Load Balancing** and **Auto Scaling Group** from AWS will also be applied to make sure the application is persistent and scalable.

Architecture

Architecture diagram



Sequence Diagram

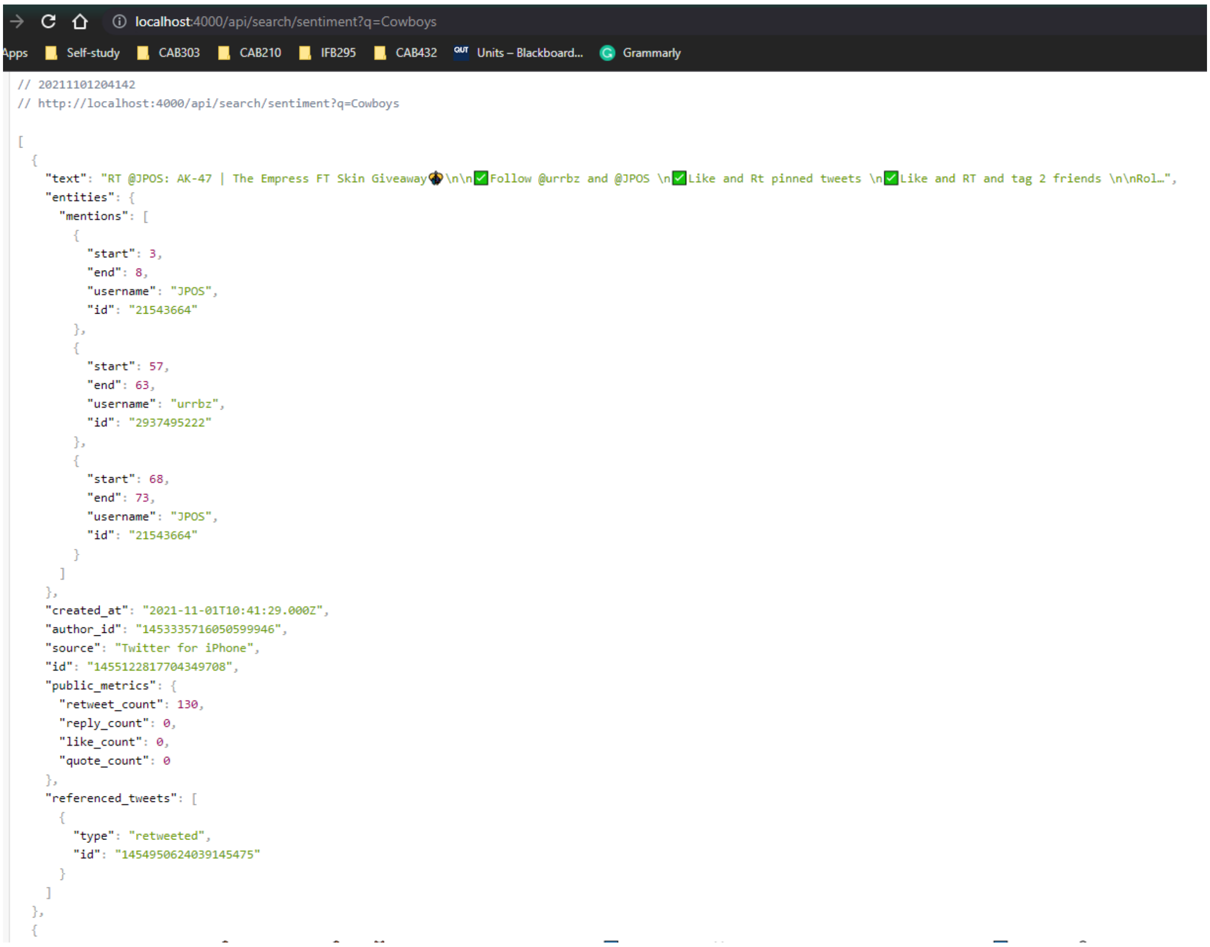


Client / server demarcation of responsibilities

Firstly, the main page with a list of trending hashtag or topic extracts from Twitter Trend API is displayed, followed by a dropdown that provides specific place options (based on WOEID) for users to filter. The list will change along with the choice in the dropdown. Next to the dropdown is a button used to filter the nearest trend around the user's location. The Twitter Nearest Trend API will require the longitude and latitude of the current user to compare and sort the list in the closest order. Each of the trends in the application makes to be a clickable button. When clicked, a data processing step is carried out in the backend. First, the trends/topic are being assigned as the query for Twitter Searching API, where information from 100 relevant posts is retrieved. The server then extracts texts and start to analyse them by using the sentiment analysis method from NLTK - Natural. The trends that is saved to Elasticache, will also play as the key for its sentiment result that are saved to the S3 Bucket for further quickly access. After the processing stage is done, the client will return a pop-up div: average score of sentiment, sentiment proportion chart, and a button that direct the user to that topic. (Refer to sequence diagram).

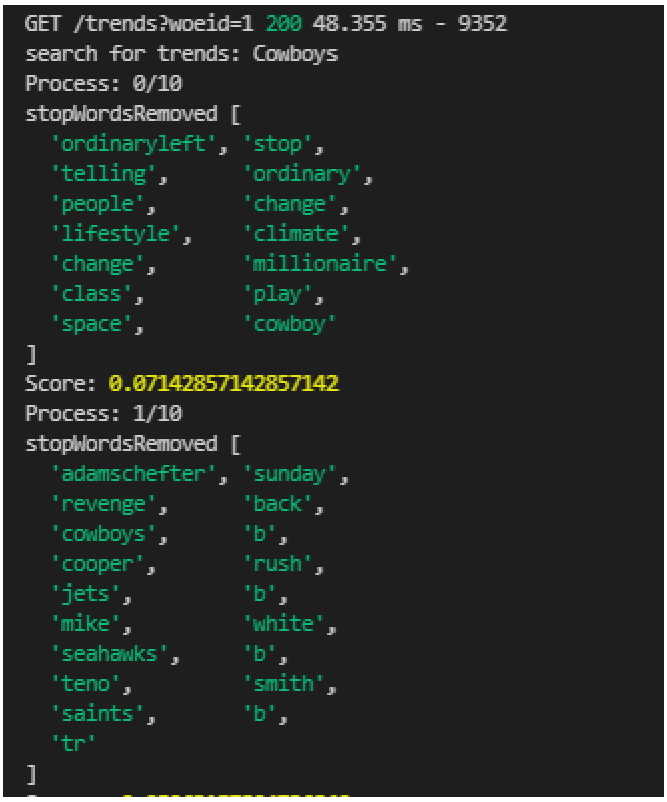
Response filtering / data object correlation

Raw data (The information is not correct due to the data is continuing update) :



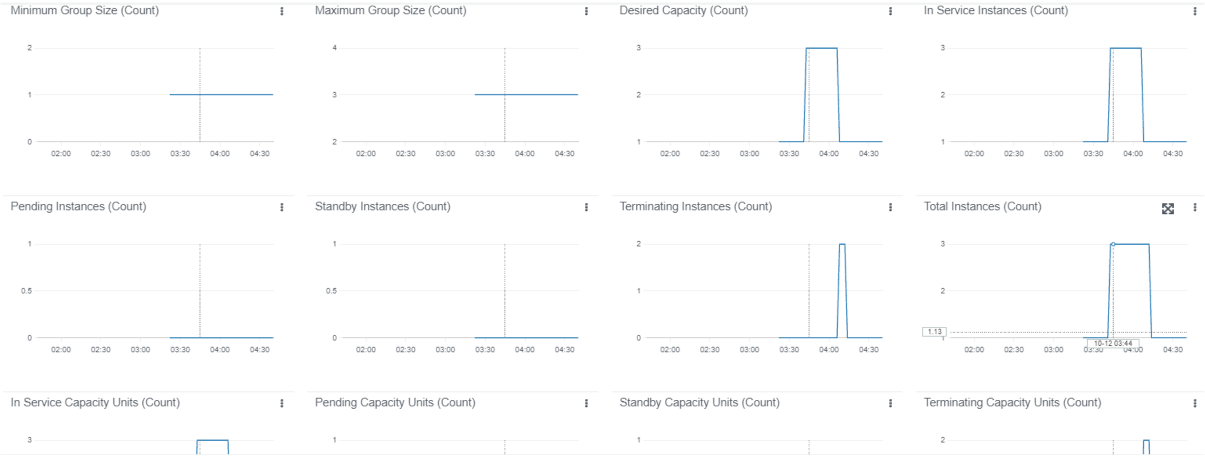
```
// 20211101204142
// http://localhost:4000/api/search/sentiment?q=Cowboys

[
  {
    "text": "RT @JPOS: AK-47 | The Empress FT Skin Giveaway🏆\n\n✅Follow @urrbz and @JPOS \n✅Like and Rt pinned tweets \n✅Like and RT and tag 2 friends \n\nRol…",
    "entities": {
      "mentions": [
        {
          "start": 3,
          "end": 8,
          "username": "JPOS",
          "id": "21543664"
        },
        {
          "start": 57,
          "end": 63,
          "username": "urrbz",
          "id": "2937495222"
        },
        {
          "start": 68,
          "end": 73,
          "username": "JPOS",
          "id": "21543664"
        }
      ]
    },
    "created_at": "2021-11-01T10:41:29.000Z",
    "author_id": "1453335716050599946",
    "source": "Twitter for iPhone",
    "id": "1455122817704349708",
    "public_metrics": {
      "retweet_count": 130,
      "reply_count": 0,
      "like_count": 0,
      "quote_count": 0
    },
    "referenced_tweets": [
      {
        "type": "retweeted",
        "id": "1454950624039145475"
      }
    ]
  },
  {
```

Filtered data:



```
GET /trends?woeid=1 200 48.355 ms - 9352
search for trends: Cowboys
Process: 0/10
stopWordsRemoved [
  'ordinaryleft', 'stop',
  'telling',      'ordinary',
  'people',       'change',
  'lifestyle',    'climate',
  'change',       'millionaire',
  'class',        'play',
  'space',        'cowboy'
]
Score: 0.07142857142857142
Process: 1/10
stopWordsRemoved [
  'adamschefter', 'sunday',
  'revenge',      'back',
  'cowboys',      'b',
  'cooper',       'rush',
  'jets',         'b',
  'mike',         'white',
  'seahawks',     'b',
  'teno',         'smith',
  'saints',       'b',
  'tr'
]
```

Scaling and Performance


Minimum Group Size (Count)
Maximum Group Size (Count)
Desired Capacity (Count)
In Service Instances (Count)
Pending Instances (Count)
Standby Instances (Count)
Terminating Instances (Count)
Total Instances (Count)
In Service Capacity Units (Count)
Pending Capacity Units (Count)
Standby Capacity Units (Count)
Terminating Capacity Units (Count)
02:00
02:30
03:00
03:30
04:00
04:30
1.13
10-12 03:44

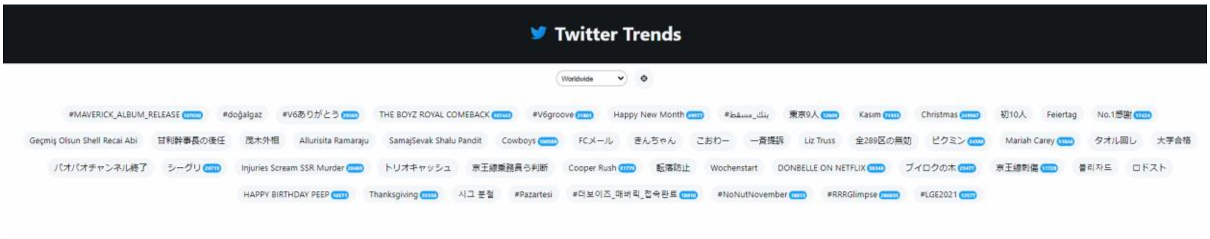
Test plan

Task	Expected Outcome	Result
Display the trend following the specific place	Result displayed in the main page	PASS
Change the specific places by clicking in dropdown	Trends change following	PASS
Integrating Elasticache Redis	Data uploaded to Elasticache Redis	PASS
Integrate S3 Bucket	Data uploaded to S3 Bucket	PASS
Extract data from search and do sentiment analysis	The Console print out the score and stopped words	PASS
Display the pop-up trend data	The relevant of the trend display (name, chart)	PASS
Extract Twitter API	Extract data and print to the console	PASS
Extract data from storage	The time of the loading for each sentiment analysis is decreased significantly	PASS

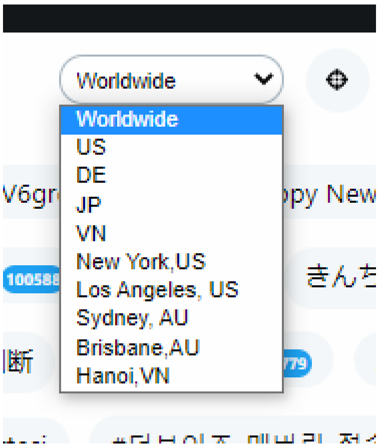
Difficulties / Exclusions / unresolved & persistent errors /

The Trend is said about toward trends. For example, a topic named Happy Halloween might have mostly sentiment result that is neutral and positive as only 100 texts are searched. It causes the exactness of the sentiment. Further development with a widened range of searches and in-depth in various aspects will likely help increase the result.

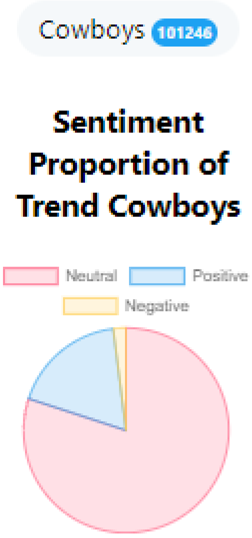
User guide



Access to the website, users will be able to see list of current in specific places (Worldwide trend by default). The User can change the place by selecting options given in the dropdown bar. Or filtering with their current location by click in the icon to the right. But there might be restriction whether the website is unsecure (Won't allow to use the users' current location -> Cannot use the function).



Users might click on the trend that they are interested in, and the website will do the sentiment analysis. The expected result showed below:



Finally, by clicking to the name of the trend, the website will navigate to the link of the topic on Twitter.

References

yoursTRULY (2021). *Twitter trends API fullstack application using Reactjs and Nodejs Express*. Youtube. https://www.youtube.com/watch?v=1KwbjY_5qYU

Appendices